

12th

eScience

IEEE INTERNATIONAL CONFERENCE

2016

October 23–27, 2016
Baltimore, Maryland, USA

[Conference Information](#)

[Table of Contents](#)

[Author Index](#)

[Search](#)



Proceedings of the 2016 IEEE 12th International Conference on e-Science. IEEE Catalog Number CFP1606A-USB. ISBN 978-1-5090-4272-2. Abstracting is permitted with credit to the source. Libraries are permitted to photocopy beyond the limit of U.S. copyright law for private use of patrons those articles in this volume that carry a code at the bottom of the first page, provided the per-copy fee indicated in the code is paid through Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923. For reprint or republication permission, email to IEEE Copyrights Manager at pubs-permissions@ieee.org. All rights reserved. Copyright ©2016 by IEEE. For technical support please contact Causal Productions (info@causalproductions.com).

Dear Colleagues,

It is our honor and pleasure to welcome you to 12th IEEE International Conference on eScience (eScience 2016), held in Baltimore, MD, USA, from October 23 to October 26th. The IEEE eScience conference remains the premier venue for foundational and applied research in computationally enabled discovery, and the quality of papers submitted this year reflects that strong reputation. Selecting the final program from among the excellent submissions was a challenge; it was a privilege to be able to become familiar with the community's work.

With the 2016 eScience conference we have attempted to broaden the reach of the community by calling for work in data-intensive science, new scientific discoveries made through the application of advanced cyberinfrastructure, and new results in the science of science --- studies assessing the impact and effectiveness of the process of science. We think these topics have made the program richer, and we hope you'll agree. We have at least one session on each of these areas, in addition to a strong core program in novel cyberinfrastructure for science.

We wish to thank our program committee for the thoughtful discussions during the review phase, our sponsors, and the eScience Steering Committee. Special thanks to Dan Katz for communicating the institutional knowledge about the conference to the organizing committee as a whole.

Welcome to Baltimore!

Alex Szalay and Randal Burns, General co-chairs
Bill Howe and Martin Kersten, Program co-chairs

Committees

General Co-Chairs:

Alex Szalay (JHU)

Randal Burns

Program Co-Chairs:

Bill Howe (UW)

Martin Kersten (CWI)

Workshop Chairs:

Tamas Budavari (JHU)

Howie Huang (GWU)

Communications Chair:

Bonnie Souter (JHU)

Local Accommodations Chair:

Margie Gier (JHU)

Dissemination Chair

Yanif Ahmad (JHU)

Proceedings Chair:

David A. C. Beck (UW)

Environmental Computing Workshop:

Dieter Kranzlmüller (LMU & LRZ)

Matti Heikkurinen (LMU)

Anton Frank (LRZ)

Steering Committee:

Daniel S. Katz (chair, UIUC)
David Abramson (UQ)
Ewa Deelman (ISI/USC)
Dave De Roure (UO)
Shantenu Jha (Rutgers)
Dieter Kranzlmüller (LMU)
Erwin Laurie (KTH)
Claudia Bauzer Medeiros (UC)
Kai Nan (CAS)
Paul Roe (QUT)
Paul Watson (UN)

Program Committee

Abani Patra – SUNY at Buffalo
Achim Streit – Karlsruhe Institute for Technology
Adam Barker – University of St Andrews
Alberto Krone-Martins – CENTRA/SIM
Alexandros Labrinidis – University of Pittsburgh
Andrea Clematis – IMATI-CNR
Andrew Wendelborn – University of Adelaide
Antonella Galizia – IMATI-CNR
Antonio Parodi – CIMA research foundation
Arun Konagurthu – Monash University
Bertram Ludaescher – University of Illinois
Bill Howe – University of Washington
Blesson Varghese – University of St Andrews
Bruno Schulze – LNCC
Chen Li – Monash University
Christine Borgman – UCLA
Christopher Re – Stanford University
Dan Gunter – Lawrence Berkeley National Laboratory
Daniel Halperin – University of Washington eScience Institute
Daniel Katz – University of Chicago & Argonne National Laboratory
David Walker – Cardiff University
David Maier – Portland State University
Dieter Kranzlmüller – Ludwig-Maximilians-Universität München
Domenico Talia – University of Calabria
Donatella Castelli – CNR-ISTI
Douglas Thain – University of Notre Dame
Edwin Valentijn – University of Groningen
Erwin Laure – KTH/PDC
Ewa Deelman – USC Information Sciences Institute
Federico Ruggieri – INFN/GARR
Foteini Alvanaki – CWI
Frank Seinstra – Netherlands eScience Center
Fushen Wang – Stony Brook University
Gang Chen – Institute of High Energy Physics
Gang Luo – University of Utah
Gilberto Pastorello – Lawrence Berkeley National Laboratory
Habibah Wahab – Universiti Sains Malaysia
Hai Jin – Huazhong University of Science and Technology

Hans-Joachim Bungartz – Institut fur Informatik
Heike Neuroth – Potsdam University of Applied Science
Ian Foster – Argonne National Laboratory
Ilkay Altintas – University of California, San Diego
Ioan Raicu – Illinois Institute of Technology
James Hogan – Queensland University of Technology
Jane Hunter – University of Queensland
Jeff Tan – IBM Research
Jiangning Song – Monash University
Jianwu Wang – San Diego Supercomputer Center
Johan Montagnat – CNRS
Jose Moreira – University of Aveiro
Jose Jr. – Universidade de Salo Paulo
Josh Vogelstein – JHU
Kai Nan – Computer Network Information Center
Kevin Ashley – Digital Curation Centre
Kevin Page – University of Oxford
Kris Bubendorfer – Victoria University of Wellington
Kristin Tufte – Portland State University
Kyle Chard – Computation Institute
Lee Giles – Pennsylvania State
Lennart Johnsson – University of Houston
Lukasz Miroslaw – Wroclaw University of Technology
Lutz Gross – School of Earth Sciences
Lynda Hardman – Centrum Wiskunde & Informatica
Madhusudhan Govindaraju – SUNY Binghamton
Malcolm Atkinson – School of Informatics
Manolis Koubarakis – University of Athens, Greece
Mark Hedges – Kings College London
Marta Mattoso – COPPE- Federal Univ. Rio de Janeiro
Martin Berzins – SCI Institute
Marty Humphrey – University of Virginia
Matti Heikkurinen – MNM-Team (Ludwig-Maximilians-Universität München)
Morris Riedel – Forschungszentrum Jlich
Nancy Wilkins-Diehr – San Diego Supercomputer Center (SDSC)
Nandamudi Vijaykumar – National Institute for Space Research (INPE)
Nick Jones – New Zealand eScience Infrastructure and University of Auckland
Nils Felde – MNM-Team (Ludwig-Maximilians-Universität München)
Omar Boucelma – LSIS
Paolo Missier – Newcastle University

Paul Bonnington – Monash University
Paul Roe – QUT
Paul Watson – Newcastle University
Paul Brown – Paradigm4
Peer Kroger – Ludwig-Maximilians-Universität München
Per Oster – CSC IT Center for Science Ltd
Peter Baumann – Jacobs University Bremen
Peter Kunszt – Support and Services for Science IT
Raul Sirvent – Barcelona Supercomputing Center
Riccardo Murri – Support and Services for Science IT
Rob Mei – Centrum Wiskunde & Informatica (CWI)
Romulo Goncalves – Netherlands eScience Center
Ron Perrott – Oxford University
Selver Softic – Graz University of Technology
Shantenu Jha – Rutgers
Simon Cox – University of Southampton
Stephane Gancarski – LIP6
Steven Newhouse – EMBL-EBI
Suresh Marru – Indiana University
Susumu Date – Osaka University
Sverker Holmgren – Division of Scientific Computing
Syed Abidi – Dalhousie University
Tamas Budavari – JHU
Tanu Malik – UChicago
Thomas Heinis – Imperial College
Tim Kraska – Brown University
Tom Hacker – Purdue University
Torben Pedersen – Aarhus University
Tori Risch – Uppsalla University
Vasa Curcin – King’s College
Wilco Hazeleger – Netherlands eScience Center
Wouter Los – University of Amsterdam
Xiaoru Yuan – Peking University
Yanif Ahmad – JHU
Yannis Ioannidis – University of Athens
Yogesh Simmhan – Indian Institute of Science (IISc)
Yoshio Tanaka – AIST
Yunquan Zhang – Chinese Academy of Sciences
Zhiming Zhao – University of Amsterdam

Sponsors



ALFRED P. SLOAN
FOUNDATION

2016 IEEE 12th International Conference on e-Science

e-Science 2016

Table of Contents

Welcome Message from the Chairs	i
Committees	ii
Steering Committee	iii
Program Committee	iv
Sponsors	vii
Author Index	450

Paper Session 1: Cloud Computing

Campus Compute Co-Operative (CCC): A Service Oriented Cloud Federation	1
<i>Andrew Grimshaw, Md Anindya Prodhan, Alexander Thomas, Craig Stewart, Richard Knepper</i>	
Albatross: An Efficient Cloud-Enabled Task Scheduling and Execution Framework Using Distributed Message Queues	11
<i>Iman Sadooghi, Geet Kumar, Ke Wang, Dongfang Zhao, Tonglin Li, Ioan Raicu</i>	
Prediction of Workflow Execution Time Using Provenance Traces: Practical Applications in Medical Data Processing	21
<i>Hugo Hiden, Simon Woodman, Paul Watson</i>	

Paper Session 2: Curation

Accelerating Data-Driven Discovery with Scientific Asset Management	31
<i>Robert E. Schuler, Carl Kesselman, Karl Czajkowski</i>	
Cooperative Human-Machine Data Extraction from Biological Collections	41
<i>Icaro Alzuru, Andréa Matsunaga, Maurício Tsugawa, José A.B. Fortes</i>	
Generating Knowledge Networks from Phenotypic Descriptions	51
<i>Fagner Leal Pantoja, Patrícia Cavoto, Julio Cesar dos Reis, André Santanchè</i>	

Paper Session 3: Reproducibility

PRUNE: A Preserving Run Environment for Reproducible Scientific Computing	61
<i>Peter Ivie, Douglas Thain</i>	
Converting Scripts into Reproducible Workflow Research Objects	71
<i>Lucas A.M.C. Carvalho, Khalid Belhajjame, Claudia Bauzer Medeiros</i>	
A Framework for Scientific Workflow Reproducibility in the Cloud	81
<i>Rawaa Qasha, Jacek Cala, Paul Watson</i>	
Conducting Reproducible Research with Umbrella: Tracking, Creating, and Preserving Execution Environments	91
<i>Haiyan Meng, Douglas Thain, Alexander Vyushkov, Matthias Wolf, Anna Woodard</i>	

Paper Session 4: Algorithms

Fast Window Aggregate on Array Database by Recursive Incremental Computation.....	101
<i>Li Jiang, Hideyuki Kawashima, Osamu Tatebe</i>	
Representation of Continuously Changing Data Over Time and Space: Modeling the Shape of Spatiotemporal Phenomena.....	111
<i>José Moreira, Paulo Dias, Pedro Amaral</i>	
An n -Gram Cache for Large-Scale Parallel Extraction of Multiword Relevant Expressions with LocalMaxs.....	120
<i>Carlos Goncalves, Joaquim F. Silva, Jose C. Cunha</i>	
DOT-K: A Distributed Online Top-K Elements Algorithm Using Extreme Value Statistics	130
<i>Nicholas Carey, Tamás Budavári, Yanif Ahmad, Alexander S. Szalay</i>	

Paper Session 5: Workflow

Budget Distribution Strategies for Scientific Workflow Scheduling in Commercial Clouds	137
<i>Vahid Arabnejad, Kris Bubendorfer, Bryan Ng</i>	
Datatrack: An R Package for Managing Data in a Multi-Stage Experimental Workflow — Data Versioning and Provenance Considerations in Interactive Scripting.....	147
<i>Philip Eichinski, Paul Roe</i>	
Starting Workflow Tasks Before They're Ready	155
<i>Wladislaw Gusew, Björn Scheuermann</i>	

Paper Session 6: Potpourri and Science of Science

Automatic Glomerulus Extraction in Whole Slide Images Towards Computer Aided Diagnosis	165
<i>Yan Zhao, Edgar F. Black, Luigi Marini, Kenton McHenry, Norma Kenyon, Rachana Patil, Andre Balla, Amelia Bartholomew</i>	
Improving Data Provenance Reconstruction via a Multi-Level Funneling Approach.....	175
<i>Subha Vasudevan, William Pfeffer, Delmar Davis, Hazeline Asuncion</i>	
Plans and Performances: Parallels in the Production of Science and Music	185
<i>David De Roure, Graham Klyne, Kevin R. Page, John Pybus, David M. Weigl, Matthew Wilcoxson, Pip Willcox</i>	

Paper Session 7: New Platforms

MOHA: Many-Task Computing Meets the Big Data Platform.....	193
<i>Jik-Soo Kim, Cao Nguyen, Soonwook Hwang</i>	
Globus Auth: A Research Identity and Access Management Platform	203
<i>Steven Tuecke, Rachana Ananthakrishnan, Kyle Chard, Mattias Lidman, Brendan McCollam, Stephen Rosen, Ian Foster</i>	
Management, Analysis, and Visualization of Experimental and Observational Data — The Convergence of Data and Computing	213
<i>E. Wes Bethel, Martin Greenwald, Kerstin Kleese van Dam, Manish Parashar, Stefan M. Wild, H. Steven Wiley</i>	
Developing a Citizen Science Web Portal for Manual and Automated Ecological Image Detection.....	223
<i>Marshall Mattingly III, Andrew Barnas, Susan Ellis-Felege, Robert Newman, David Iles, Travis Desell</i>	

Paper Session 8: Machine Learning Applications

Applying Data Mining Methods for the Analysis of Stable Isotope Data in Bioarchaeology	233
<i>Markus Mauder, Eirini Ntoutsis, Peer Kröger, Christoph Mayr, Gisela Grupe, Anita Toncalo, Stefan Hödlz</i>	
Data-Oriented Neuron Classification from Their Parts	243
<i>Evelyn Perez Cervantes, Cesar Henrique Comin, Roberto Marcondes Cesar Junior, Luciano da Fontoura Costa</i>	
Detecting Wildlife in Uncontrolled Outdoor Video Using Convolutional Neural Networks.....	251
<i>Connor Bowley, Alicia Andes, Susan Ellis-Felege, Travis Desell</i>	
Using LSTM Recurrent Neural Networks to Predict Excess Vibration Events in Aircraft Engines	260
<i>AbdElRahman ElSaid, Brandon Wild, James Higgins, Travis Desell</i>	

Invited Short Papers: Hot Topics 1

SimP: Secure Interoperable Multi-Granular Provenance Framework	270
<i>Amani Abu Jabal, Elisa Bertino</i>	
A Comprehensive Scenario Agnostic Data LifeCycle Model for an Efficient Data Complexity Management	276
<i>Amir Sinaeepourfard, Jordi Garcia, Xavier Masip-Bruin, Eva Marín-Tordera</i>	
Semantic Accountable Matchmaking for E-Science Resource Sharing.....	282
<i>Zeqian Meng, John Brooke, Rizos Sakellariou</i>	
Apache Airavata Security Manager: Authentication and Authorization Implementations for a Multi-Tenant eScience Framework	287
<i>Supun Nakandala, Hasini Gunasinghe, Suresh Marru, Marlon Pierce</i>	

Paper Session 9: Social Analytics and Population Simulation

Privacy-Protected Social Media User Trajectories Calibration	293
<i>Shuo Wang, Richard Sinnott, Surya Nepal</i>	
A Machine Learning Analysis of Twitter Sentiment to the Sandy Hook Shootings.....	303
<i>Nan Wang, Blessen Varghese, Peter D. Donnelly</i>	
A Hybrid Approach to Population Construction for Agricultural Agent-Based Simulation	313
<i>Peng Chen, Tom Evans, Michael Frisby, Eduardo Izquierdo, Beth Plale</i>	
A Parallel Microsimulation Package for Modelling Cancer Screening Policies	323
<i>Andreas Karlsson, Niten Olofsson, Erwin Laure, Mark Clements</i>	

Invited Short Papers: Hot Topics 2

OntoSoft: A Distributed Semantic Registry for Scientific Software	331
<i>Yolanda Gil, Daniel Garijo, Saurabh Mishra, Varun Ratnakar</i>	
A Secure Data Enclave and Analytics Platform for Social Scientists	337
<i>Yadu N. Babuji, Kyle Chard, Aaron Gerow, Eamon Duede</i>	
Reproducibility in Computer Vision: Towards Open Publication of Image Analysis Experiments as Semantic Workflows.....	343
<i>Ricky J. Sethi, Yolanda Gil</i>	
Crossing Analytics Systems: A Case for Integrated Provenance in Data Lakes	349
<i>Isuru Suriarachchi, Beth Plale</i>	
Interactive Provenance Summaries for Reproducible Science.....	355
<i>Xiang Li, Xiaoyang Xu, Tanu Malik</i>	

Paper Session 10: High-Performance Computing

ExTASY: Scalable and Flexible Coupling of MD Simulations and Advanced Sampling Techniques	361
<i>Vivekanandan Balasubramanian, Iain Bethune, Ardit Shkurti, Elena Breitmoser, Eugen Hruska, Cecilia Clementi, Charles Laughton, Shantenu Jha</i>	
Leveraging Burst Buffer Coordination to Prevent I/O Interference	371
<i>Anthony Kougkas, Matthieu Dorier, Rob Latham, Rob Ross, Xian-He Sun</i>	
A Fast Algorithm for Neutrally-Buoyant Lagrangian Particles in Numerical Ocean Modeling	381
<i>Renske Gelderloos, Alexander S. Szalay, Thomas W.N. Haine, Gerard Lemson</i>	
Data Management and Simulation Support Accelerating Carbon Capture Through Computing.....	389
<i>You-Wei Cheah, Joshua Boverhof, Abdelrahman Elbashandy, Deb Agarwal, Jim Leek, Thomas Epperly, John Eslick, David Miller</i>	

ECW 2016 — Environmental Computing Workshop

Session 1: Tools and Platforms

OMUSE: Oceanographic Multipurpose Software Environment.....	399
<i>Inti Pelupessy, Ben van Werkhoven, Arjen van Elteren, Jan Viebahn, Adam Candy, Simon Portegies Zwart, Henk Dijkstra</i>	
Automating Environmental Computing Applications with Scientific Workflows.....	400
<i>Rafael Ferreira da Silva, Ewa Deelman, Rosa Filgueira, Karan Vahi, Mats Rynge, Rajiv Mayani, Benjamin Mayer</i>	
Utilising Amazon Web Services to Provide an On Demand Urgent Computing Facility for climateprediction.net.....	407
<i>Peter Uhe, Friederike E.L. Otto, Md Mamanur Rashid, David C.H. Wallom</i>	

Session 2: Data

Automatic Fire Perimeter Determination Using MODIS Hotspots Information	414
<i>N. Chiaraviglio, T. Artés, R. Bocca, J. López, A. Gentile, J. San Miguel Ayanz, A. Cortés, T. Margalef</i>	
A Columnar Architecture for Modern Risk Management Systems	424
<i>Romulo Goncalves, Sisi Zlatanova, Kostis Kyriakos, Pirouz Nourian, Foteini Alvanaki, Willem van Hage</i>	
The eWaterCycle Project.....	430
<i>Niels Drost, Rolf Hut, Maarten van Meersbergen, Edwin Sutanudjaja, Marc Bierkens, Nick van de Giesen</i>	
The IQmulus High Volume Fusion and Analysis Platform for Geospatial Point Clouds, Featuring a Marine Bathymetry Use Case	431
<i>Quillon Harpham</i>	

Session 3: Case Studies

Manufacturing of Weather Forecasting Simulations on High Performance Infrastructures.....	432
<i>V. Šipková, L. Hluchý, M. Dobrucký, J. Bartok, B.M. Nguyen</i>	
The Climate Change for Flood and Debris Mitigation After Typhoon Morakot 2009 in Taiwan.....	440
<i>Harold Yih-Chi Tan</i>	
Performance Evaluation of Environmental Applications Using TELEMAC-MASCARET on Virtual Platforms.....	441
<i>Minh Thanh Chung, Manh-Thin Nguyen, Nhu-Y Nguyen-Huynh, Thong Nguyen, Nam Thoai</i>	
Environmental Computing — Applications, Tools and Data Solutions	449
<i>Matti Heikkurinen, Dieter Kranzlmüller</i>	

Campus Compute Co-operative (CCC): A Service Oriented Cloud Federation

Andrew Grimshaw, Md Anindya Prodhan, and Alexander Thomas

Department of Computer Science

University of Virginia

Charlottesville, Virginia

Email: {grimshaw,mtp5cx,apt9jf}@virginia.edu

Craig Stewart and Richard Knepper

Indiana University Bloomington

Bloomington, Indiana

Email: {stewart,rknepper}@indiana.edu

Abstract—Universities struggle to provide both the quantity and diversity of compute resources that their researchers need when their researchers need them. Purchasing resources to meet peak demand for all resource types is cost prohibitive for all but a few institutions. Renting capacity on commercial clouds is seen as an alternative to owning. Commercial clouds though expect to be paid. The Campus Compute Cooperative (CCC) provides an alternative to purchasing capacity from commercial providers that provides increased value to member institutions at reduced cost. Member institutions trade their resources with one another to meet both local peak demand as well as provide access to resource types not available on the local campus that are available elsewhere. Participating institutions have dual roles. First as consumers of resources when their researchers use CCC machines, and second as producers of resources when CCC users from other institutions use their resources. In order to avoid the tragedy of the commons in which everyone only wants to use resources, the resource providers will receive credit when their resources are used by others. The consumer is charged based on the quality of service (high, medium, low) and the particulars of the resource provided (speed, interconnection network, memory, etc.). Account balances are cleared monthly.

This paper describes solutions to both the technical and socio-political challenges of federating university resources and early results with the CCC. Technical issues include the security model, accounting, job specification/management and user interfaces. Socio-political issues include institutional risk management, how to manage market forces and incentives to avoid sub-optimal outcomes, and budget predictability.

Index Terms—distributed computing, cloud computing, high performance computing

I. INTRODUCTION

Computational techniques are increasingly central to research success and used across essentially all areas of science and engineering research and many humanities disciplines. Research universities face a significant challenge: How can they provide their researchers and scholars the computational infrastructure they need to be competitive with their peers, while managing costs in a time when there are increasing pressures from many directions on university budgets?

The problem is particularly acute for three reasons. First, use of cyberinfrastructure is now ubiquitous across the sciences and increasingly widely used in humanities research. Many institutions of higher education are seeing significant growth in the user base of local computational and data analysis resources. Second, federal funding for cyberinfrastructure is

at best flat, with many institutions finding it harder to obtain federal funding for locally-managed cyberinfrastructure resources than it seemed in years past, and requests for national resources such as those managed by XSEDE [1] exceeding the available resources by several fold. Last, there are now a multitude of computational modalities supporting research. Researchers perform many different types of simulations and analyses that require a variety of advanced cyberinfrastructure systems, for example: tightly coupled HPC engines, high-throughput clusters, large memory nodes, Hadoop clusters, GPU-based systems, etc.. To make matters more complicated researchers may make use of multiple computational modalities at different points in a research project or even a single distributed workflow making use of resources at a variety of scales with a variety of temporal access patterns (continuous versus bursty usage).

This paper describes the CCC, the Campus Compute Cooperative, which is entering service as a pilot project in the US. The CCC combines three basic ideas into a production compute environment. The first idea is that resource providers charge for the use of their resources and resource consumers pay for the use of resources. Note that buying and selling does not necessarily involve the exchange of real money. Instead allocation units are exchanged between participating institutions. The mechanism for resolving chronic debits is discussed in section IV. Charging for compute cycles is not novel in academia. Creating a market for cycles, where actors are both buyers and sellers is novel.

The second idea is providing differentiated quality of service levels that allow users to specify the quality of service they need. Quality of service attributes include the urgency of the job. Does the job need to run right now (urgent)? Or can the job run later (best effort)? Other quality of service attributes include the type of resource requested. Does the application have large memory requirements? Does it require a very fast interconnect? Does it require many cores on the same node? Or does it have a small memory footprint? Users select, and pay for, the quality of service they desire. By allowing users to express the value of the job in terms of what they are willing to pay we ensure that the CCC always executes the most important job next, increasing overall institutional value.

The third idea is federation: combining the resources of

multiple institutions into a single, larger compute environment. This is akin to using an M/M/K queue versus K independent M/M/1 queues. By increasing the resource pool size we can provide better quality of service.

These ideas have been around for some time. The CCC combines them into an open-membership production compute environment for academic research as a means for addressing the computational challenges facing universities.

Three simple use cases illustrate how the CCC provides value to participating institutions. First, well-funded Dr. Chemistry's research group at University 1 (U1) has a major grant review in three weeks and urgently needs immediate access to 5,000 cores for a week to prepare. U1 does not have 5,000 free cores. Most of their machine is tied up with another urgent job. Fortunately, U2 has no urgent jobs running and has plenty of capacity not being used by urgent jobs. Non-urgent jobs can be terminated or suspended and Dr. Chemistry's jobs can now be run on a mix of resources, some at each school. By selling capacity to U1, U2 gains the ability to acquire resources from U1 when it needs them. U1 is able to meet its immediate need without over-provisioning resources. *Value of CCC: CCC provides a mechanism for one participating institution to obtain "burst capacity" from other institutions now, without paying funds to a commercial provider (which it cannot afford to do or is prohibited from doing for policy or regulatory reasons).*

Second, Dr. Biology's application at University 2 (U2) needs 512 GB of memory. Dr. Biology only periodically needs to execute the jobs, but when executed dozens of 512 GB machines are needed. Unfortunately U2 only has two 512 GB nodes. U1 has two dozen such nodes. U2 could buy two dozen expensive 512 GB nodes and use them for smaller memory jobs when they are not needed for Dr. Biology. But the nodes are quite expensive. By using U1's nodes a capital expense is avoided. *Value of CCC: exchange of one type of computational resource for another, among two different institutions with different local resources saves on capital expenditure.*

Third, U1 economics professor Dr. MacroEcon models the world economy using large numbers of ensemble calculations, each one of which is itself a large scale Monte Carlo simulation. The total execution time is expected to be in the millions of CPU hours. Dr. MacroEcon does not have a large budget and is going to be in Europe for the next three months and does not particularly care when any given job completes. Instead, Dr. MacroEcon wants them to complete before returning from Europe. Dr. MacroEcon's jobs can be executed with a low priority and fill CPU slots when no higher priority jobs are ready. *Value of CCC: By being flexible about completion speed, researchers can get access to more resources for less.*

A. Campus Compute Co-operative Goals

There are two goals for the CCC. The first is to provide university scientists the computational resources they need when they need them in such a manner as to increase overall institutional value while reducing costs. We achieve this goal by deploying the CCC on participating institutional resources

and setting up a mechanism in which universities (or university units such as departments and research labs) trade the use of their computational resources with other universities, e.g., compute clusters, large shared memory machines. The CCC uses production quality software already available as open source created by the project participants. Three university computing resources serve as the pilot hardware the Rivanna cluster at University of Virginia (6000 cores), Computer Science department resources at UVA (600 cores), and Big Red II at IU (20,000 cores).

The second CCC goal is to test the hypothesis that federated differentiated quality of service among universities with market based resource allocation will result in a greater value and better ability for researchers to complete their most critical analyses for all the institutes individually and overall, compared to using local resources only. This paper focuses on the first goal.

We believe that the CCC can result in net increase in effectiveness of use of each participating institution's resources, and benefit researchers at all participating institutions, for two reasons. First, prior work has demonstrated that cyber-infrastructure (CI) resources are more effectively used when there are multiple classes of service available on those CI resources with different 'costs' to the user [2] . Second, we believe such market economies can succeed now and in the coming years within academia because the diversity of CI resources needed by researchers is too great for even the best funded universities to maintain and support all of the existing modalities of advanced CI resources on one campus.

The remainder of the paper is organized as follows. We begin with related work followed by a brief discussion of the software infrastructure being used. We then discuss the equally important non-technical political and social issues involved in a cross campus federation. Next we present our early results followed by a summary and call for participation.

II. RELATED WORK

There is a vast related literature in distributed systems, distributed scheduling systems, on-demand computing, cloud computing, grids, and grid economies going back over 40 years. Here we will focus on contemporary cloud and on-demand computing, grid economic models, distributed scheduling, pilot jobs, and one of the better known alternative systems, the Open Science Grid.

A. On Demand & Cloud Computing

Cloud computing has had a much more significant impact in the commercial space than either Grid computing or earlier metasystems efforts. This is the result of a number of factors including the use of virtual machines as sand-boxes that gave users tremendous flexibility; the rise of pay per use on line media (movies, music), social networking, and other areas that paved the way for market acceptance; and the entrance of Amazon into the market place, a vendor that was not only trusted but assumed to know what it was doing. Regardless of the cause of the change, the change is clearly upon us.

The National Institute of Standards and Technology (NIST) defines clouds as having five essential characteristics [3]: on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. The CCC is a cloud under the NIST definition. NIST further defines three service models, Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). The CCC straddles the definition of a PaaS and IaaS cloud. The CCC allows users to deploy applications that may or may not depend on CCC services, and in the near future will include the ability to deploy and execute virtual machines.

There are clearly many other vendors of cloud services. The most representative examples are Amazon with the Elastic Computer Cloud (EC2) and S3 services and Microsoft with the windows Azure Cloud. The CCC differs from these commercial cloud providers in two significant ways. First, in the CCC economic actors are both producers and consumers. When using Amazon or Azure it is clear that Amazon is selling and the user is buying. Thus, when interacting with commercial cloud providers there will always be a bill to be paid. In the CCC, bills can be paid in-kind. Second, new resource providers can join the community (enlarging the resource pool) without any change required on the client side.

Note that within Amazon EC2 there are many different “instance types” that correspond to qualities of service[4]. The Amazon instance types are partitioned in one dimension by the relative scale of the instance: micro, small, medium, large; in another dimension by characteristics such as “general purpose”, “compute optimized”, “memory optimized”, “GPU optimized”, “IO optimized”, etc.; and also by whether the instances are fixed price and will last as long as you want, or whether they are “spot” instances that are cheap but which can be terminated at any time. The CCC uses a similar model.

B. Grid Scheduling and Grid Economics

There is a rich literature in both computational grids and grid economic models [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17]. The CCC builds on and extends these earlier systems. The CCC is based on open standards developed in the Open Grid Forum [18] and elsewhere that could only be developed once the underlying concepts of authentication, job description and management, data transfer, and so on had been settled. One of the defining characteristics of most early metasystems and grid efforts was that resources were “free” or that users already had to have permission to use the resources.

In terms of grid economics [19], [6], [20] the work has been done in simulation, or developed but not used in a production environment. Several research systems like: Spawn [21], AppLes [22], Popcorn [23], G-commerce [17], OCEAN [17], Nimrod [24], GridSim [25], and GridEcon [19] have explored the use of different economic models for managing resources in grid environment. In each of these systems a market is established to trade resources between the resource providers and resource users. But none of these actually considered the quality of service requirements of the users. Buyya et al.

[6] proposed a QoS based scheduling algorithm but there the authors only considered the deadlines as a QoS measure, not the value of the job .

The CCC is different in that our explicit goal is to construct and operate a production quality market where the actors buy and sell computational services to test the hypothesis that markets combined with differentiated quality of service result in a win-win for all actors.

C. Comparison of CCC and Open Science Grid

The Open Science Grid [16] is the best known community execution environment today. With technology initially developed primarily for the high energy physics community in the late 90’s and early 2000’s, the OSG is transnational high-throughput computing resource used by several science communities. From the OSG web page [26]:

“The OSG facilitates access to distributed high throughput computing for research in the US. The resources accessible through the OSG are contributed by the community, organized by the OSG, and governed by the OSG consortium.”

There are a number of differentiators between the CCC and the OSG. The most compelling though is the difference in economic models. The underlying model in the OSG is that the community contributes resources in an altruistic manner, and that whole communities are allocated a portion of the resources for execution. There is no explicit quid quo pro. The motivation for an institution to contribute something concrete, particularly in this era of tighter resources and cost accounting is not clear. In the CCC on the other hand the motivation to make resources available is explicit.

A second difference is the differentiated quality of service mechanism that explicitly models the economic value of jobs. The CCC differentiates between less important jobs and urgent jobs, and schedules them appropriately. The goal is to increase institutional value as measured by what users (and their respective institutions) are willing to pay. Thus, urgent requests, or requests from well-funded research teams, can be moved to the head of the line in a neutral, consistent manner.

A third difference is the class of applications that are supported. OSG is explicitly designed for high throughput sequential jobs. The CCC on the other hand is built on a standard extensible resource and execution model [27], [28] that does not care what the “activities” are. The CCC currently supports sequential, threaded, and MPI jobs. Extending to VMs will not be technically difficult.

Fourth is extensibility and openness. The CCC is based upon open, publically discussed, extensible standards developed in the Open Grid Forum. The OSG on the other hand is based upon Condor, a tightly controlled proprietary system developed at the University of Wisconsin.

III. CCC SOFTWARE ENVIRONMENT

The CCC builds upon the XSEDE Execution Management Services (EMS) [29], the Global Federated File System (GFFS) [30], and the Campus Bridging use cases [31]. The

CCC software environment consists of client-side tools that are used to interact with the CCC services and Web Services containers that are deployed to proxy access to back-end compute, data, and identity resources on participating institution machines. We are using Genesis II [32], [33] for both the client and server software. Client and container installers are available for Linux, Windows, and MacOS. RPM's are available for Linux as well. The Genesis II implementation along with underlying web services “stack” are all open source and are part of the XSEDE software stack. More details about Genesis II software is available in[32], [33]. We will use the phrase GORM in this document to refer to the Genesis II OmniBus Reference Manual [32] and L3D.

Genesis II was developed at UVA and is an implementation of the XSEDE Global Federated File System (GFFS) [30] and the XSEDE Execution Management Services (EMS). The XSEDE architecture is described in detail in the XSEDE Level 3 Decomposition [29] (hereafter the L3D) and the GORM. The GFFS and EMS are based upon standard, open, Web Services components that interact as a service cloud. The “object model” used is one in which all objects are defined by a WSDL interface, many have state, have associated metadata accessible via WS Resource Framework [34] get/set resource properties, have an address represented by a WS-Addressing End Point Reference (EPR) [35], and have a globally unique WS-Naming Endpoint Identifier (EPI) [36].

Below we examine four features of the CCC: the GFFS directory structure, the security model, the execution management architecture, and the accounting mechanism. For each we give a brief description followed by CCC-specific configuration information.

A. Directory and Discovery Service

A central feature of the Genesis II and the XSEDE architecture is the Global Federated File System name space. The GFFS namespace is modeled on the Unix directory structure. Like a Unix file system maps path names to inodes the GFFS maps path names from a root “/” to resource endpoints known as EPRs that support the WS-Resource Framework model, and that implement some service type. The basic service types are WS-Trust authentication services [37], ByteIO file services [38], RNS directory services [39], directory export services (L3D), BES execution services [28], grid queue services (L3D), DAGMAN workflow execution services (L3D), and running jobs (activity endpoints) (L3D). In this proposal we will focus on Execution Management Services. For details on any of the services and how they interact see the L3D, the GORM, and the use case architectural responses [40].

The XSEDE GFFS namespace has a set of conventions on how the namespace is organized, e.g. a compute resource in the XSEDE namespace is `/resources/xsede.org/BigRed2`. A running job is `/resources/xsede.org/queues/CBQueue/jobs/mine/running/simulation-1-4-best`. A directory made available on my desktop is `/home/xsede.org/grimshaw/BD`. Keep in mind that an endpoint may have many aliases, i.e., there may

be many paths that point to the same endpoint; the CBQueue above may also exist as `/resources/CCC/normalQ`. The CCC uses the XSEDE namespace.

The Genesis II client supports access to the GFFS namespace via a command line interface, via a GUI, via APIs, and by “mounting” the GFFS namespace as a file system on a Linux system using FUSE [41], or in Windows via a SMB mount. Once the GFFS is mounted as a file system by the operating system, existing operating system tools such as bash or user applications can directly operate on remote data and compute resources, e.g., `cd` into the working directory of a running job and `tail -f` an output file.

CCC Configuration: The CCC has its own directory entries in the XSEDE namespace, `/groups/CCC`, `/home/CCC`, `/users/CCC`, `/docs/CCC`, `/resources/CCC` and `/bin/CCC`. Beneath each of groups, users, and home there are links to institution specific, and institution maintained directories that point to the institutions users, their home directories, and any institution-specific and public groups.

B. Security Model & Services

The CCC security model is described in the XSEDE L3D and has been extensively reviewed. In a nutshell on-the-wire data integrity is provided by https (using TLS), authentication is via signed delegated SAML certificates with a base of the certificate chain signed using an X.509 certificate, and authorization is via access control lists(oarwx) in which the principles are the signers of the base SAML assertions. Authentication tokens are transferred in the SOAP header of a web services call as specified in the Web Services Interoperability Basic Security Protocol, known as the WSI-BSP [42]. Clients, acting on behalf of users, accumulate a “credential wallet” that is a set of signed SAML assertions, that is passed on every web services call.

The Genesis II server implements several WS-Trust Secure Token Services (STS) for different back-end authentication protocols. The current back-end STSs implement a user-name/password using an internal database, a Kerberos back-end that authenticates against Kerberos servers, and an InCommon Enhanced Client [43] or Proxy back-end that interacts with CILogon [44] and InCommon services. STS endpoints can also represent groups (or roles) . Instead of signing an assertion stating that a particular user is JOAN or SARAH, the assertion states that the properly delegated holder-of-key is a member of the group.

Genesis II clients interact with STS services to acquire delegated signed SAML user and group assertions that are placed in the clients credential wallet for subsequent outcalls. The user can add/remove tokens from their credential wallet.

CCC Configuration: We have constructed a set of groups for the CCC and placed them in `/groups/CCC`. Each participating institution manages its own group of people authorized to use CCC resources. These groups are named `/groups/CCC/ < institution - name >`. Authorization to acquire credentials to institutional groups is managed by the participating institution not by CCC staff. CCC staff

maintain a group `/groups/CCC/ccc-users`. Authorization to acquire ccc-users credentials will be given to holders of group credentials from participating institutions.

To use CCC compute resources, an individual first authenticates and acquires an end-user credential from an STS that his/her institution trusts, such as the institution's InCommon server. The credential is placed in his/her credential wallet. The individual then requests an institutional group certificate from `/groups/CCC/ <institution-name>` and places that credential in his/her wallet. Finally the individual requests the ccc-users credential from `/groups/CCC/CCC-user` and places it in his/her wallet. Assuming the user has been properly authorized at every step, the user now has the ccc-user credential and can use CCC resources. These steps are configured to execute immediately once the user credential is available. Participating institutions choose who will use their credentials and can revoke access at any time. Similarly, CCC staff only control which institutions are permitted to acquire ccc-user credentials. Finally, note that any resource owner can stop accepting CCC jobs on a resource any time with a single command on a resource end-point, `grid chmod -rx <path-to-resource>CCC-user`

C. Execution Management

Execution Management Services are described in detail in the L3D and GORM. EMS services are concerned with instantiating and managing to completion units of work that may consist of single activities, sets of independent activities, or workflows. EMS addresses problems with executing units of work including their placement, "provisioning", and lifetime management. These problems include, but are not limited to, the following: finding candidate execution locations, selecting the execution location, preparing the execution location for execution, initiating the execution, and managing the execution.

The solution to these five problems consists of a standard job description mechanism and the use of set of services that decompose the EMS problem into multiple, replaceable components that all enable specific architecture functions. These components include the Job Submission Description Language [27] documents to describe jobs, OGSA Basic Execution Services (BES) [28] to execute jobs, Grid Queues to manage large sets of jobs and to schedule them onto BESes, GFFS directory paths [30] and resource registries [29] to discover resources, and job managers to implement application-specific functionality.

1) *Client tools*: Client GUI tools include the Job-Tool for creating and submitting JSDL files and the queue manager tool. End-users can manage (list/start/stop/re-start & clean up) their jobs with the queue manager tool. Users can also examine detailed log histories to understand why a particular job may have failed. Users can also interact with their running jobs using the GFFS by cd'ing into the job's working directory and reading (or tailing) output files being generated by the job regardless of where it is running. Administrators can change scheduler parameters and manage end user jobs. Command

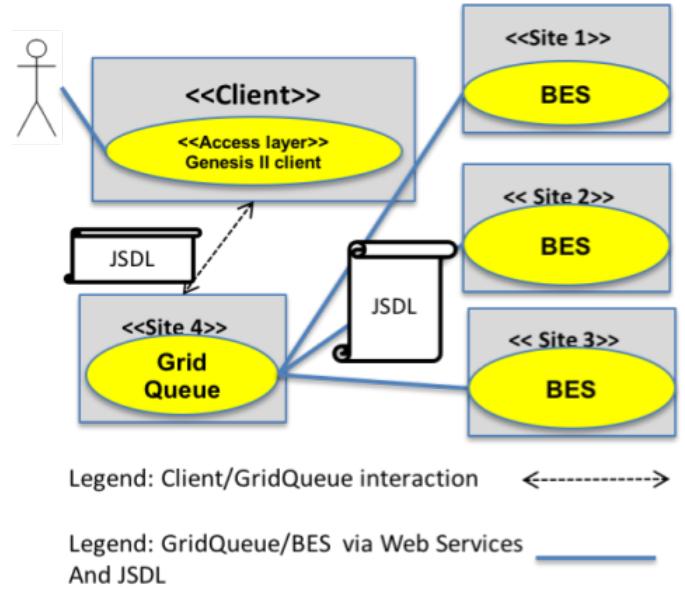


Fig. 1. Clients send jobs to gridQueues. GridQueues send jobs to BESes. BESes wrap compute resources such as clusters. BESes may execute the job directly or delegate to other BESes.

line tools to submit and manage jobs are available as well for those who prefer scripting.

2) *JSDL*: JSDL [27], [45] is a standard XML based language used to describe jobs. A JSDL 1.0 document has three main components: a resource requirements section, an application information section, and a data staging section.

The JSDL resources section contains application requirements such as operating system version, minimum amount of memory, number of processors and nodes, wall clock time, file systems to mount, and so on. It consists both of a standardized set of descriptions, as well as an open-ended set of matching requirements that are arbitrary strings.

The JSDL application information section includes items such as the command line to execute, the parameters, the job name, account to use, and so on.

The JSDL staging section consists of a set of items to stage-in before the job is scheduled in the local environment, and a list of items to stage-out post-execution. Each staging defines the protocol to use, the local file(s) to use as the source or target, and URIs for the corresponding source or target. Supported protocols include http(s), ftp, scp, sftp, GridFTP, mailto, and the XSEDE GFFS.

JSDL++ [46] is a non-standards track JSDL extension developed to address the short-coming that each JSDL document describes exactly one set of possible resource matches with exactly one corresponding application execution description. For example, "the job requires 8 nodes, each with 8 cores, 64 GB memory, and MPICH 1.4: in that environment stage-in executable Y and execute 'Y 1024 -opt1'". Suppose an equally suitable option is "the job requires 1 node, each with 64 cores, 256 GB memory, and pthreads: in that environment stage-in executable Z and execute 'Z -opt2'"? JSDL++ allows the spec-

ification of an arbitrary list of options and the JSDL processing agent is free to use any one of the options for which it can find the resources. The diversity of compute resources and tuned, platform-specific implementation for common packages, has led to use JSDL++ in the CCC.

3) *OGSA Basic Execution Services (BESs)*: OGSA BES service [28], [11] endpoints represent the ability to execute jobs, specifically execute JSDL documents. The BES interfaces combined with JSDL create a virtual execution environment for XSEDE and the CCC in which all execution resources, desktops, department servers, campus clusters, clouds, and supercomputers provide the same interface.

GridQueues: The basic idea for a grid queue is simple and is shown in Figure 1. GridQueue services are instantiated and configured to use particular BES services at each site. For example, the highPriority gridQueue will be configured to use the highPriority BESes at each site. Clients (users) send their jobs defined in JSDL to gridQueues. The gridQueue matches job resource requirements specified in JSDL against BES resource properties and schedules jobs onto BESes. In other words it provides execution services on-demand. The gridQueue monitors the job until it completes. If the job or BES fails, the gridQueue reschedules it on another resource.

CCC Configuration: To add their computing resources (clusters) to the CCC resource owners download and bring up a Genesis II container on a head node or login node that can submit jobs to their cluster. Once the container is operational the resource owners instantiate one or more BES endpoints on their new container. Each BES resource is configured to submit jobs to a particular local queue, for example the “high priority” queue or the “economy” queue. Each BES is given a path, e.g., `/resources/CCC/BigUniveristy/FooClusterHighPriority`. The BESes have their access control lists initialized (rx) to allow members of the `/groups/CCC/ccc-users` group to run and manage jobs.

Once the BES endpoints are instantiated and initialized they are added to the appropriate grid queue endpoint, e.g., `/resources/CCC/queues/highPriority` or `/resources/CCC/queues/economy` using the link command in the grid command line shell.

```
cd/resources/CCC
```

```
ln BigUniv/FooClusterHighPriority queues/highPriority/resources/BigUHighPriority
```

Once linked in, end-users jobs that match the FooCluster resource properties are eligible to be scheduled on FooCluster.

D. Accounting

Whenever a job is run on the CCC an accounting record is created on the local BES that contains the time the job started/ended, the command line, the resource used, how much resource was used, and the signed SAML assertions that indicate who the user was, and what user and group credentials were used in the execution. The signature chains provide non-repudiation. Different resources types at different qualities of service have different costs in accounting units. This conceptually similar to the Amazon EC2 pricing model.

Note that resource owners “earn” accounting units when jobs run on their resources and users home institutions “pay” for resource consumed by their users. BES resource usage records are swept up into the accounting database periodically.

The accounting database can be sliced many ways; by resource to show all jobs run by users on the resource; by user showing which resources they used; by institution as a resource consumer and producer, and so on. The objective is to provide institutions the ability to clearly understand the costs (their resources being used by others) and benefits (their users using other institutions resources) of participating in the CCC.

IV. SOCIAL AND POLITICAL ISSUES

Solving the technical challenges is a necessary but not sufficient condition for success. There are also social and institutional issues that must be addressed. Each institution has a number of stakeholders: systems administrators, institutional security offices, university administrators and committees that oversee the resources, and faculty-led teams that use the resources. These stakeholders have different goals, fears, and constraints. Faculty want to get the most research done with the minimum of hassle and cost; system administrators want to protect their resources; university administrators are responsible for providing their researchers with a competitive advantage via advanced infrastructures yet must also weigh the costs, financial obligations, and risks to the institution.

The challenge is that all three groups must be satisfied before progress can be made. At the same time there is a chicken and egg problem. Faculty teams will not spend the effort to learn a new technique until they know it will be available and provide them with an advantage. Administrators are not going to consider a new arrangement until the faculty are clamoring for it, and many system administrators will not load new software until they understand it - and they will not take the time to do so until they are told to do so by management.

A. University Administrators and Security Officers

The value proposition to institutions is simple. By providing differentiated quality of service and federating their resources with other universities they can increase the value to their stakeholders (faculty-led research teams) and reduce capital expenditures by reducing hardware diversity and eliminating the need to provision for peak demand. The simulation results in [2] and a basic understanding of micro-economics is all that is needed for most to see the potential benefit.

A potential benefit is not enough when the risks, particularly financial risks, to the institution are not clear. Signing contracts, particularly when the maximum liability of the contract cannot be easily established can be difficult for state institutions. Therefore an MOU structure is used. The MOU states that individual institutions are responsible for vetting their own users and for deciding which of their users will be given membership in their university-X-group. The institution is also responsible for their users’ usage. Every month all members are provided with the accounting records broken

down by user and institution. It is up to institutions what to do with this information. Under the terms of the MOU any institution can walk away at any time. Similarly an institution can remove any user from its institutional group at any time.

After much discussion about how to synchronize local charging mechanism between institutions we decided that how institutions provide allocations to their users is up to them. Local circumstance and culture around resource allocation is simply too difficult to harmonize. Therefore, institutions may choose to issue “grants”, give all of their users unlimited access, charge to the PI’s grants, or some other mechanism. That is not a CCC decision, it is a local institution decision.

So what happens when institutions become chronic debtors or chronically run a surplus? If an institution is running a deficit with respect to other institutions it is up to each institution pair to decide how to resolve the situation. Obvious solutions include i) that the debtor institution pays cash to wipe out its debit (this may be attractive compared to purchasing hardware to cover peak load), ii), the debtor makes more, or more attractive resources, available to the community so that it will “earn” more and can wipe out its debt, iii) the creditor stops accepting jobs from the CCC or the debtor institution. Thus all institutions run the risk that they may end with a surplus. Institutions can mitigate this risk by both encouraging their users to use the CCC - and thus other institutions resources, or by monitoring usage and shutting off access if they feel too much resource is being used.

B. Faculty-led research teams

Faculty led research teams typically are most interested in getting their results with as little effort as possible. The work of getting results is most often left to graduate students and post-docs some of whom are very computer savvy but most of whom are not. The last thing that they want is to spend a great deal of time and energy porting codes only to discover limited benefit. To address this risk we have taken a two pronged approach. First, the interaction paradigm is modeled on memes with which they are already very familiar: queues and file systems. Users define their jobs (including array and parameter sweep jobs) and submit them to queues using familiar tool names: qsub, qstat, qkill, etc. The queues have names that clearly specify the quality of service the queue provides: best effort, normal, high priority, urgent. Similarly, users discover and use resources (including files and running jobs) using a file-system like path-based naming scheme. Users navigate the namespace using familiar GUI-based browsers or Unix directory commands, ls, cd, and so on.

Second, for early users we are assisting with setting up the JSDL files necessary to run on the CCC and exploit the heterogeneous resource base. This builds up a set of examples that are available for other users in the GFFS /bin directory. For example, we have developed a LAMMPS JSDL that is configured to use either CRAY MPI, OpenMPI, or POSIX threads on a single node depending on resource availability. These sample JSDL files can then be used by others. In the

case of LAMMPS users they can use the LAMMPS JSDL with changes only to reflect input/output staging.

C. System Administrators

System administrators are (rightly) concerned with things going horribly wrong. It is their job to ensure the safe continuous operation of the resources for which they are responsible. They must be assured that installing new software and new mechanisms to access the resources do not result in a disaster, and if something goes wrong they can rapidly identify who (a person) is responsible. Equally important is that the software be easy to understand, install, and manage.

We address the first of these risks by running CCC container services as un-privileged (non-root) processes. This significantly reduces the damage a successful attack or rogue user could do. Second, all communication is encrypted and digitally signed with signature chains going back to the original authentication events. Third, we provide a tool for local system administrators that allows them to specify a job ID from the local queuing system, e.g., SLURM, that will return the user ID and authentication domain of the actual end user that is running that local job.

With respect to installation and management. Ease of installation is supported via the provisioning of both RPMs and installers that automatically install and update with just a few clicks. Further, since the accounts running the CCC services have no special privilege, local administrators can simply create two new non-root accounts and let CCC administrators do the installation and configuration.

Finally, we have found that system administrators often handle requests from end users to be placed at the head of the queue. This requires them to make judgment calls and explain to users why either their jobs did not get priority, or explain to other users why somebody else got to jump the queue. They appreciate automating that decision making process using user-specified and paid for QoS specifications.

V. INITIAL RESULTS

Below we describe our initial results on the CCC. We begin with a description of the resources included in the CCC. We then discuss the synthetic tests we have executed to ensure correct function and capacity. Finally we describe three applications that have been moved onto the CCC. The three applications were chosen because they represent three different kinds of applications that we expect to see: parallel applications, high throughput computing applications, and large single core jobs.

A. Resources

The initial rollout at two sites using existing resources (Indiana University and the University of Virginia).

1) *Indiana University*: *Big Red II* is a Cray XE6/XK7 supercomputer with a hybrid architecture providing a total of 1,020 compute nodes: 344 CPU-only compute nodes, each containing two AMD Opteron 16-core Abu Dhabi x86_64 CPUs and 64 GB of RAM, and 676 CPU/GPU compute nodes,

each containing one AMD Opteron 16-core Interlagos x86_64 CPU, one NVIDIA Tesla K20 GPU accelerator with a single Kepler GK110 GPU, and 32 GB of RAM.

2) *University of Virginia: Rivanna* is a Cray CS300AC with 240, 128 GB , 20 core, Cray nodes and approximately 1200 cores of other nodes all served by a 1.4PB Luster file system.

The CS department maintains a heterogeneous cluster of 600 cores. Because the nodes are very heterogeneous, the CS department has several BESes configured, one for each node type. Thus there are CS department BESes for fat node machines with 256GB of memory and 64 cores, less fat nodes with 128GB of memory and 32 cores; and older nodes with 48GB of memory and 8 cores.

During the initial rollout, researchers were not charged for the resource usage. We just logged the resource usage information for future simulations. However, a proposed pricing structure us defined in the MoU. Currently resource pricing at University of Virginia is driven by a cost-recovery model based on TCO.

B. Synthetic Tests

The synthetic tests are divided into two groups. The first group is the Genesis II regression tests executed against changes to the SVN repository whenever either the trunk or a development branch is updated. The regression tests are comprehensive. They test the security infrastructure, all permutations of the GFFS file system client and server mechanisms for both large multi-gigabyte files and directories with thousands of small files; the GFFS replication and recovery mechanisms; and the execution management system with thousands of jobs, file staging tests for small files and large files, parameter sweep jobs, singleton jobs, and MPI jobs. These tests are designed to test whether we have broken any functionality with the update.

We periodically (weekly to bi-weekly) execute a subset of the execution management regression test against the production grid. The subset consists of the same types of jobs (staging large and small files, MPI, HTC, etc.), but a substantially smaller number of jobs as we do not want to stress the underlying production computing environment and consume too many resources. Every five minutes a Nagios test determines whether the various endpoints are operational. These tests are designed to test whether the production system is behaving as expected.

The second set of tests are focused on performance. In particular we are interested in the performance, including capacity, of the gridQueues. (Performance notes for the GFFS can be found in the documentation section of the Genesis II web page). BES performance, i.e. jobs per second per BES, is not as important as any given BES that interacts with a single local queue is unlikely to see more than one job per second which we can easily handle. Similarly, local policies usually prevent us from placing more than a thousand jobs on any given resource (usually much less). So BES performance/capacity is not an issue.

GridQueue performance and capacity is a concern so we have tested it extensively. The tests fall into three categories:

capacity, rate, and latency. Capacity measures how many jobs the gridQueue can hold, rate measures the number of client tests per second that can be handled, and latency measures the end-to-end time for request execution.

Capacity is measured by simply loading up jobs into the queue. There are three resources that limit capacity. Disk space to store the job records in the database, memory to store the in-memory copy of the abstracted job information, and human patience to wait for a complete job list to be generated. All three costs are approximately linear in the number of jobs. Therefore one must have enough memory, disk, and patience for the desired number of jobs. Requirements for 100,000 jobs (much, much more than most people will ever see) are:

TABLE I
TIME/MEMORY/DISK FOR 100,000 JOBS IN GRIDQUEUE

Job Count	Peak Memory	Disk	Time to list jobs
100,000	4GB	10GB	243 seconds, 411 jobs/sec

Rate and latency are measured by submitting jobs to the queue. Jobs can be submitted to gridQueues in two ways, via the submit API interface used by the GUI and command line interface (qsub) and by writing JSDL files directly to the queue via the GFFS, e.g., `cp jsdl -file.xml < pathToQueue > /submissionpoint`. Jobs can be submitted as single jobs, as JSDL parameter sweeps [45] , or as vectors of single jobs in a single submission command. All measurements were performed on a 3.1 GHZ 16 core AMD 4386 with 92GB of memory. The GFFS container was configured with a maximum of 16GB of memory.

TABLE II
RATE AND LATENCY FOR JOBS SUBMITTED IN THE QUEUE

	Latency	Rate (jobs/sec)
Singleton via CLI	210 ms	3.3
1000 via CP	216 ms	3.3
1000 via parameter sweep	216 ms	3.3

C. Real Application Test

To illustrate the efficacy of our system, we ran three user applications on our infrastructure. In order to cover all the bases, we ported large sequential jobs, single and multi-node parallel (MPI) jobs and high throughput computing jobs on CCC cloud. As an example of a large sequential job we ported a user job to evaluate a search engine. To examine CCC's ability with parallel jobs we used LAMMPS.

1) *Large Sequential Jobs (Search Engine Evaluation):* To validate code portability for our system, we first experimented with a serial job with a 10 GB input file which was simulating the performance of a search engine. One of the graduate students of the CS department at UVa designed a search engine and used the simulation to evaluate the performance of the search engine in our system. The code was developed in JAVA and used the well known machine learning library “*apache lucene*” to rank the queries based on the 10 GB pregenerated index. The approximate runtime for the job was about 12 hrs.

A JSIDL file with appropriate stage-in/out files was used to submit the jobs to the CCC-Queue and the Queue was able to schedule and run the jobs successfully both on Rivanna and BigRed II. To run the job the BES staged in the input file to the appropriate scratch directory and the generated output files were staged back on the grid. For BigRed II, the 10GB input files were staged to IU within minutes.

2) Single/Multi-node Parallel Jobs (LAMMPS): LAMMPS (“Large-scale Atomic/Molecular Massively Parallel Simulator”) [47] is a molecular dynamics program from Sandia National Laboratories. LAMMPS can be used to model atoms or, more generically, as a parallel particle simulator at the atomic, meso, or continuum scale which can run on single processors or in parallel using message-passing techniques (MPI). Many of its models have versions that provide accelerated performance on CPUs, GPUs, and Intel Xeon Phis. LAMMPS has potentials for solid-state materials (metals, semiconductors) and soft matter (biomolecules, polymers) and coarse-grained or mesoscopic systems.

In our experiment, we wanted to port LAMMPS in our system such that the users can run their LAMMPS simulation seamlessly, regardless of the underlying environment and architecture. Both Big Red II and Rivanna have the LAMMPS module installed in their system for the users. However, users need different modules and scripts for running their simulation in the two environments due to the heterogeneity between the systems. In CCC, we have MPI BESes setup (one for each resource) to run parallel simulations on the resources. JSIDL++ allowed us to define one job to submit in the CCC-Queue which can be run on either of the resources. The queue decides on the appropriate resource to schedule the job based on the availability of the resources and the matching parameter(s) defined on the job description. Current JSIDL specification does not allow us to specify module loads on the resources, so we had to use the environment variables to simulate the module loads for LAMMPS on the resources. With our experiment, we were able to run both single node and multi-node MPI LAMMPS simulations with CCC. This provides an evidence of the portability of MPI jobs in CCC. The JSIDL++ project file is stored in `/bin/lammps` location in the GFFS .

3) High Throughput Computing (Astrochemical Simulation): A high-throughput, genetic programming framework was created for use on the CCC. The goal was to fit a single-threaded astrochemical simulation to experimental data by mutating the simulation’s parameters. Due to the size of the parameter space and the typical execution time of the simulation, sequential genetic programming techniques alone were not sufficient for this problem. Furthermore, the simulation’s low-memory footprint made it ideal for execution on inexpensive compute resources. The solution was to encapsulate a permutation of parameters and the simulation as a job and schedule it on the CCC. Each independent job was queued in the CCC, which allowed scheduled jobs to execute as compute resources became available. Access to the GFFS presented a uniform filesystem interface to store nearly the entire state of the genetic programming framework such as

new permutations of parameters, results of previous jobs, and accounting information. Currently an external program (one that does not execute on the CCC) orchestrates the scheduling of jobs and formulation of results. There are plans to package the orchestration program for execution on the CCC. This modification would fully automate the parameter search for the astrochemical simulation by allowing the CCC to manage all aspects of the parameter search space. Ultimately this high-throughput, genetic programming framework frees the simulation’s researcher from coding an HTC test harness herself while leveraging the appropriate compute resources.

VI. CCC-SUMMARY AND FUTURE WORK

The CCC uses open source, standards-compliant, software stack that is a part of the NSF-funded XSEDE infrastructure. The software has been extensively vetted and tested by XSEDE via XSEDE’s Software Development and Integration group. The CCC is a realization of the XSEDE Campus Bridging use case CBUC-6, a Shared Virtual Compute Facility [31]. The CCC exists within the XSEDE GFFS namespace.

The CCC is now operational between UVA and IU. The CCC provides a market-based resource sharing model that rewards resource providers for sharing their resources and charges users based on the attributes of the resource they use as well as on the quality of service provided. The goal is to provide resources to users when they need them, and provide increased institutional value at reduced cost.

We presented early results for both a set of synthetic tests as well as the experience on three applications: a sequential application, a well-known parallel community code, and a custom astro-chemistry application that uses a genetic algorithm and high throughput computing to examine a large search space. The CCC supports single and high throughput sequential, threaded, and MPI applications.

Future work. The CCC has just begun. Next steps are to move more applications onto the CCC, build up the user base, and to expand the set of participating institutions; in other words to expand the size of the market. The CCC is an open organization that other institution or research labs are invited to participate. Expanding the market will provide additional value both for the new market participants and existing market participants. Why adding new members helps the new members is clear. A larger market adds value to existing market participants by expanding the pool of resources that can be used and expanding the set of users that will use the resources.

Technically we want to

- examine moving to a dynamic pricing model where resource providers advertise their prices for different qualities of service rather than using fixed prices
- provide users with the option of creating their own custom gridQueues that optimize their own objective functions, e.g. trade-off performance for cost
- allow users to add desktop VMs into the resource mix to earn additional allocation
- support starting VM’s for users, not just jobs

We are just at the beginning of this project. We are looking forward to both creating a valuable national resource as well as having fun. If you would like to join the CCC send email to xcghelp@virginia.edu.

ACKNOWLEDGMENTS

This document was developed with support from National Science Foundation (NSF) grant OCI-1053575. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

- [1] "Extreme science and engineering discovery environment," <https://www.xsede.org/> [accessed May 2016].
- [2] M. A. Prodhan and A. Grimshaw, "Market-based on demand scheduling (mbods) in co-operative grid environment," in *Proceedings of the 2015 XSEDE Conference*. ACM, 2015, p. 26.
- [3] P. Mell and T. Grance, "The nist definition of cloud computing," *Communications of the ACM*, vol. 53, no. 6, p. 50, 2010.
- [4] "Amazon ec2 instance types," <https://aws.amazon.com/ec2/instance-types/> [accessed May 2016].
- [5] F. Berman, G. Fox, and A. J. Hey, *Grid computing: making the global infrastructure a reality*. John Wiley and sons, 2003, vol. 2.
- [6] R. Buyya, D. Abramson, and S. Venugopal, "The grid economy," *Proceedings of the IEEE*, vol. 93, no. 3, pp. 698–714, 2005.
- [7] R. Buyya and K. Bubendorfer, *Market-oriented grid and utility computing*. Wiley Online Library, 2010.
- [8] D. Weitzel, B. Bockelman, D. Fraser, R. Pordes, and D. Swanson, "Enabling campus grids with open science grid technology," in *Journal of Physics: Conference Series*, vol. 331, no. 6. IOP Publishing, 2011, p. 062025.
- [9] I. Foster and C. Kesselman, *The Grid 2: Blueprint for a new computing infrastructure*. Elsevier, 2003.
- [10] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, and J. Reich, "The open grid services architecture, version 1.5," *Open Grid Forum*, 2006.
- [11] A. Grimshaw, M. Morgan, D. Merrill, H. Kishimoto, A. Savva, D. Snelling, C. Smith, and D. Berry, "An open grid services architecture primer," *Computer*, no. 2, pp. 27–34, 2009.
- [12] A. S. Grimshaw and A. Natrajan, "Legion: Lessons learned building a grid operating system," *Proceedings of the IEEE*, vol. 93, no. 3, pp. 589–603, 2005.
- [13] A. S. Grimshaw, A. Nguyen-Tuong, M. J. Lewis, and M. Hyett, "Campus-wide computing: Early results using legion at the university of virginia," *International Journal of High Performance Computing Applications*, vol. 11, no. 2, pp. 129–143, 1997.
- [14] A. S. Grimshaw, W. A. Wulf *et al.*, "The legion vision of a worldwide virtual computer," *Communications of the ACM*, vol. 40, no. 1, pp. 39–45, 1997.
- [15] A. Natrajan, M. A. Humphrey, and A. S. Grimshaw, *Capacity and capability computing using Legion*. Springer, 2001.
- [16] R. Pordes, D. Petrack, B. Kramer, D. Olson, M. Livny, A. Roy, P. Avery, K. Blackburn, T. Wenaus, W. Frank, I. Foster, R. Gardner, M. Wilde, A. Blatecky, J. McGee, and R. Quick, "The open science grid," in *Journal of Physics: Conference Series*, vol. 78, no. 1. IOP Publishing, 2007.
- [17] R. Wolski, J. S. Plank, T. Bryan, and J. Brevik, "G-commerce: Market formulations controlling resource allocation on the computational grid," in *Proceedings of 15th International Symposium on Parallel and Distributed Processing*. IEEE, 2001, pp. 8–pp.
- [18] "Open grid forum," <https://www.ogf.org/> [accessed May 2016].
- [19] J. Altmann, C. Courcoubetis, G. D. Stamoulis, M. Dramitinos, T. Rayna, M. Risch, and C. Bannink, "Grideon: A market place for computing resources," in *Grid Economics and Business Models*. Springer, 2008, pp. 185–196.
- [20] P. Padala, C. Harrison, N. Pelfort, E. Jansen, M. P. Frank, and C. Chakkareddy, "Ocean: the open computation exchange and arbitration network, a market approach to meta computing," in *International Symposium on Parallel and Distributed Computing*. IEEE, 2003, p. 185.
- [21] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and W. S. Storn, "Spawn: A distributed computational economy," *IEEE Transactions on Software Engineering*, vol. 18, no. 2, pp. 103–117, 1992.
- [22] F. Berman, R. Wolski, S. Figueira, J. Schopf, and G. Shao, "Application-level scheduling on distributed heterogeneous networks," in *Proceedings of the 1996 ACM/IEEE Conference on Supercomputing, 1996*. IEEE, 1996, pp. 39–39.
- [23] O. Regev and N. Nisan, "The popcorn market. online markets for computational resources," *Decision Support Systems*, vol. 28, no. 1, pp. 177–189, 2000.
- [24] D. Abramson, R. Sosic, J. Giddy, and B. Hall, "Nimrod: a tool for performing parametrised simulations using distributed workstations," in *Proceedings of the Fourth IEEE International Symposium on HPDC, 1995*. IEEE, 1995, pp. 112–121.
- [25] R. Buyya and M. Murshed, "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and computation: practice and experience*, vol. 14, no. 13-15, pp. 1175–1220, 2002.
- [26] "Open science grid," www.opensciencegrid.org [accessed May 2016].
- [27] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva, "Job submission description language (jsdl) specification, version 1.0," in *Open Grid Forum, GFD*, vol. 56, 2005.
- [28] I. Foster, A. Grimshaw, P. Lane, W. Lee, M. Morgan, S. Newhouse, S. Pickles, D. Pulsipher, C. Smith, and M. Theimer, "Ogsa basic execution service, version 1.0, gfd. 108," Tech. Rep., 2007.
- [29] F. Bachmann, I. Foster, A. Grimshaw, D. Lifka, M. Riedel, and S. Tuecke, "Xsede architecture level 3 decomposition," *version 0.972, Jun*, 2013.
- [30] A. Grimshaw, M. Morgan, and A. Kalyanaraman, "Gffs-the xsede global federated file system," *Parallel Processing Letters*, vol. 23, no. 02, 2013.
- [31] C. A. Stewart, R. Knepper, A. Grimshaw, I. Foster, F. Bachmann, D. Lifka, M. Riedel, and S. Tuecke, "Xsede campus bridging use cases," Technical report, XSEDE, Tech. Rep, Tech. Rep., 2012.
- [32] C. Koeritz, "Genesis ii omnibus reference manual," Genesis II Group, University of Virginia, Tech. Rep., 2012.
- [33] "The genesis ii project, virginia center for grid research," <http://genesis2.virginia.edu/wiki/> [accessed May 2016].
- [34] K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, and W. Vambenepe, "The ws-resource framework, version 1.0," Tech. Rep., 2004.
- [35] M. Gudgin, M. Hadley, and T. Rogers, "Web services addressing 1.0-core," *W3C, W3C Recommendation, May*, 2006.
- [36] A. Grimshaw, M. Morgan, and K. Sarnowska, "Ws-naming: location migration, replication, and failure transparency support for web services," *Concurrency and Computation: Practice and Experience*, vol. 21, no. 8, pp. 1013–1028, 2009.
- [37] A. Nadalin, M. Goodner, M. Gudgin, A. Barbir, and H. Granqvist, "Ws-trust 1.3," *OASIS Standard*, vol. 19, p. 2007, 2007.
- [38] M. Morgan, N. Chue-Hong, and M. Drescher, "Byteio specification 1.0," Tech. Rep., 2006.
- [39] M. Morgan, A. Grimshaw, and O. Tatebe, "Rns specification 1.1," in *Open Grid Forum*, 2006.
- [40] "Xsede usecase registry," <https://software.xsede.org/registry-dev/index.php> [accessed May 2016].
- [41] M. Szeredi, "Filesystem in userspace," <https://github.com/libfuse/libfuse>, 2005.
- [42] M. McIntosh, M. Gudgin, K. S. Morrison, and A. Barbir, "Basic security profile version 1.0," *WS-I Standard*, vol. 30, 2007.
- [43] "Incommon: Security, privacy and trust for the research and education community," <http://www.incommon.org/> [accessed May 2016].
- [44] "cilogon," <http://www.cilogon.org/> [accessed May 2016].
- [45] M. Drescher, A. Anjomshoaa, G. Williams, and D. Meredith, "Jsdl parameter sweep job extension," *Report no. GFD*, vol. 149, 2008.
- [46] R. Ryzhkov, "Jsdl++: Extending the jsdl specification used in grid computing environments," Department of Computer Science, University of Virginia, Tech. Rep., 2013.
- [47] "Lammps," <http://lammps.sandia.gov/> [accessed May 2016].

Albatross: An Efficient Cloud-Enabled Task Scheduling and Execution Framework Using Distributed Message Queues

Iman Sadooghi, Geet Kumar, Ke Wang, Dongfang Zhao, Tonglin Li, Ioan Raicu

Department of Computer Science, Illinois Institute of Technology, Chicago IL, USA

isadooghi@iit.edu, {gkumar7, kwang22, dzhao8, tli13}@hawk.iit.edu, iraicu@cs.iit.edu

Abstract— Data Analytics has become very popular on large datasets in different organizations. It is inevitable to use distributed resources such as Clouds for Data Analytics and other types of data processing at larger scales. To effectively utilize all system resources, an efficient scheduler is needed, but the traditional resource managers and job schedulers are centralized and designed for larger batch jobs which are fewer in number. Frameworks such as Hadoop and Spark, which are mainly designed for Big Data analytics, have been able to allow for more diversity in job types to some extent. However, even these systems have centralized architectures and will not be able to perform well on large scales and under heavy task loads. Modern applications generate tasks at very high rates that can cause significant slowdowns on these frameworks. Additionally, over-decomposition has shown to be very useful in increasing the system utilization. In order to achieve high efficiency, scalability, and better system utilization, it is critical for a modern scheduler to be able to handle over-decomposition and run highly granular tasks. Further, to achieve high performance, Albatross is written in C/C++, which imposes a minimal overhead to the workload process as compared to languages like Java or Python.

We propose Albatross, a task level scheduling and execution framework that uses a Distributed Message Queue (DMQ) for task distribution among its workers. Unlike most scheduling systems, Albatross uses a pulling approach as opposed to the common push approach. The former would let Albatross achieve a good load balancing and scalability. Furthermore, the framework has built in support for task execution dependency on workflows. Therefore, Albatross is able to run various types of workloads, including Data Analytics and HPC applications. Finally, Albatross provides data locality support. This allows the framework to achieve higher performance through minimizing the amount of unnecessary data movement on the network. Our evaluations show that Albatross outperforms Spark and Hadoop at larger scales and in the case of running higher granularity workloads.

Keywords—Data Analytics, Task Scheduling, Distributed Systems, Spark, Hadoop, Distributed Task Execution, Distributed Message Queue

I. INTRODUCTION

The massive growth in both scale and diversity of Big Data has brought new challenges as industry expectations of data processing loads continue to grow. For example, more than 2.5 exabytes of data is generated everyday [1]. At various organizations, it has become a necessity to process data at scales of Petabytes or larger. However, processing data at such scales is not feasible on a single node, therefore it is vital to utilize distributed resources such as Clouds, Supercomputers, or large clusters. Traditionally, these clusters are managed by batch schedulers, such as Slurm [2], Condor [3], PBS [4], and SGE [5]. However, such systems are not capable of handling

data processing at much larger scales. Another pitfall these systems must face is processing finer granular tasks (far more in number, much shorter run times) at larger scales. This could only be handled with a sophisticated scheduler capable of handling many more tasks per second.

The above mentioned systems were designed for clusters with far fewer amounts of batch jobs that usually run for a longer time. Those batch jobs usually have a different nature than the data analytics jobs. Moreover, they all have centralized designs that make them incapable of scaling up to the needs of today's data analysis.

Data analytics frameworks such as Hadoop [7] and Spark [6] were proposed to particularly solve the problem of data processing at larger scales. These frameworks distribute the data on multiple nodes and process it with different types of tasks. However even these frameworks have not been able to completely solve the problem. Both of the above mentioned systems have centralized bottlenecks that make them unable to handle the higher rate of task and data volume. Therefore, these frameworks are not suitable for workloads that generate more tasks in shorter periods of times. To give an example, simple applications, such as matrix multiplication, may be embarrassingly parallelizable, while generating many tasks where each task occurs in a very small amount of time [28]. Our evaluations show that Spark and Hadoop are not able to schedule and execute more than 2000 tasks per second which could add significant overhead to those applications.

Other frameworks like Sparrow [8] have tried to bypass the issue of the centralized architecture on Spark and Hadoop. Their solution is to primarily dedicate each job that consists of multiple tasks to a separate scheduler. This solution raises a few issues. First, the utilization of the whole cluster will be lower than distributed task level scheduling solutions. Since different jobs have different sizes, they will cause load imbalance for the system, and since a scheduler can only handle its own job, an idle or lightly loaded scheduler will not be able to help any overloaded schedulers. Moreover, this solution may not work well for the jobs that have significantly higher number of heterogeneous tasks. Such a job could easily saturate a single centralized task scheduler and cause significant overheads to the system.

Nowadays, most of the data analytics of big data run on the Cloud. Unlike HPC Clusters and Supercomputers that have homogeneous nodes, Clouds have heterogeneous nodes with variable node performance. Usually the underlying physical hardware is being shared across multiple Virtual Machines (VMs) and subsequently, these nodes may have variable performance [27]. Our evaluations have shown that in some cases, identical instances on AWS [26] within a given region and availability zone can exhibit variable performance results.

This means some tasks can take much longer than others. It is important for a scheduler to take this into the consideration. A simple and effective solution would be breaking tasks into smaller tasks and make them more granular. This technique is called over-decomposition. If the tasks are more granular, the workload can be better spread over the nodes and more capable nodes (faster, less-utilized) will be able to run more tasks. This would allow system utilization to significantly increase [29]. However, this poses significant challenges to the scheduling system, forcing it to make faster scheduling decisions. To allow over-decomposition and handle finer granular task scheduling, it is essential for modern schedulers to provide distributed scheduling and execution at the task level rather than the job level.

It is also critical for a scheduler to impose minimal overhead to the workload execution process starting from a single node. Tasks in utilizing a fine granular workflow could take a few milliseconds of execution time. It is not practical to run such workloads on a scheduler that takes seconds to schedule and execute a single task. Some programming languages (e.g. Java and Python) that operate at a more abstract level could add more overhead to the scheduling process. Therefore it is necessary to implement the scheduler in lower level languages such as C or C++ to achieve the best performance on a single node level.

There is an emergent need for a fully distributed scheduler that handles the scheduling at the task level and is able to provide efficient scheduling for high granular tasks. In order to achieve scalability, it is important to avoid a centralized component as it could become a bottleneck. ***In this paper, we propose Albatross: A fully distributed cloud-enabled task scheduling and execution system that utilizes a distributed Message Queue as its building block.***

The main idea of scheduling in Albatross is to use Fabriq, which is a distributed message queue [9] for delivering tasks to the workers in a parallel and scalable fashion. Most of the commonly used schedulers have a central scheduler or a controller that distributes the tasks by pushing them to the worker nodes. However, unlike the traditional schedulers, Albatross uses a pulling approach as opposed to pushing tasks to the servers. The benefit of this approach is to avoid the bottleneck of having a regional or a central component for task distribution. Albatross also uses a Distributed Hash Table (DHT) [10] for the metadata management of the workloads. There is no difference between any of the nodes in Albatross. Each node is a worker, a server, and possibly a client. The DMQ and the DHT are dispersed among all of the nodes in the system. The task submission, scheduling, and execution all happen through the collaboration of all of the system nodes. This feature enables Albatross to achieve a high scalability. The communication and the routing of the tasks all happen through hashing functions that have an $O(1)$ routing complexity. That makes the communications between the servers optimal.

Albatross is able to run workflows with task execution dependency through a built in support in the DMQ. That gives Albatross flexibility to run HPC, and data analytics jobs. An HPC job is usually defined as a Bag-of-Tasks [11] with dependencies between those tasks. The built-in task dependency support will enable the application to submit jobs

to Albatross without having to provide an application level support for task dependencies. The Directed Acyclic Graph (DAG) support also enables Albatross to run various types of data analytics workloads. The focus of this paper is mainly Map-reduce workloads.

Another important feature that is required for data analytics frameworks is data locality support. Data locality suggests that since the movement of the data on the network between the nodes is an expensive process, the frameworks have to prioritize moving tasks to the data location and minimize the data movement on the system. In Albatross this feature is supported through load balancers of the Fabriq.

Our evaluations show that Albatross outperforms Spark and Hadoop that are currently state-of-the-art scheduling and execution frameworks for data analytics in many scenarios. It particularly outperforms the other two when the task granularity increases. Albatross is able to schedule tasks at 10K tasks per second rate, outperforming Spark by 10x. The latency of Albatross is almost an order of magnitude lower than Spark. Albatross's throughput on real applications has been faster than the two other systems by 2.1x and 12.2x. Finally, it outperforms Spark and Hadoop respectively by 46x, and 600x in processing high granularity workloads on grep application.

In summary, the main contributions of Albatross are:

- The framework provides a comprehensive workload management including: data placement and distribution, task scheduling, and task execution.
- It has a fully distributed architecture, utilizing a DMQ for task distribution and a DHT for workload metadata management.
- It provides distributed scheduling at the task level, as opposed to job level distributed scheduling.
- The framework provides an efficient Task execution dependency support. It enables Albatross to run a wide range of workloads including HPC and Data Analytics.
- It provides data locality optimization.
- It offers an optimized implementation for High Performance Applications, using C/C++ programming language.

The rest of the paper is organized as follows. Section II discusses the related work. Section III provides background about the two building block components of Albatross. Section IV discusses the architecture of Albatross, followed up by the implementation details of the data locality and the task execution dependency support. Next, in section V, we discuss the Map-reduce programming model support in Albatross. Section VI briefly compares the main differences of Albatross with Spark and Hadoop as the mainstream data analytics frameworks. Section VII evaluates the performance of the Albatross in different metrics. Finally, section VIII concludes the paper and discusses the future work.

II. RELATED WORK

There have been many works providing solutions for task or job scheduling in distributed resources. Some of those works have focused on task scheduling and execution while some have focused on resource management on large clusters. Condor [3] tries to harness the unused CPU cycles on servers for batch-jobs that take longer to run. Slurm [2] is a resource

manager for Linux clusters that also provides a framework for work execution and monitoring. Portable Batch System (PBS) [4] mainly focuses on HPC. It manages batch and interactive jobs.

The main limitation of the above mentioned works is their centralized architecture that makes them not capable of handling larger scales. They were all designed for longer running batch jobs and are unable to schedule workloads in fine granular task level.

Systems such as Mesos [12], and Omega [13] are resource managers that were designed for allocating resources to different applications. The focus of this work is on a task scheduling and execution solution that could run different types of workloads. Nevertheless, both of these systems have centralized architectures and could not be the ultimate solution for distributed processing of data analytics workloads.

Many industrial systems such as Spark & Hadoop utilize iterative transformations and the Map-reduce model, respectively, but still exhibit bottlenecks, particularly the centralized task/resource managers [30]. Usually a centralized version is relatively simple to implement. However, as seen in the performance evaluation in a later section, these centralized components may be the downfall of the system as scale increases. Additionally, there are systems such as Sparrow [8], which try to reduce the consequences associated with Spark's centralized job scheduler. It has a decentralized architecture that makes it scalable. Although Sparrow provides a distributed scheduler for the jobs, the task-level scheduler is still centralized. Therefore since we are focusing on the task level scheduling and not exploring the multiple-job workloads, Sparrow would not provide any improvement in our experiments.

Another approach that has been used for distributed scheduling is work stealing. It is used at small scales successfully in parallel languages such as Cilk [14], to load balance threads on shared memory parallel machines [15][16][17]. Scalability of work stealing has not been proven yet. The randomized nature of it could cause poor utilization and scalability [18].

III. BACKGROUND

Before discussing Albatross, we are going to provide information about the two building blocks of the framework. Fabriq and ZHT are the main components of the Albatross that make the task distribution, communication and the metadata management possible in this framework.

A. ZHT Overview

For metadata management, Albatross uses ZHT which is a low overhead and low latency Distributed Hash Table, and has a constant routing time. It also supports persistence. ZHT has a simple API with 4 major methods: *insert*, *lookup*, *remove*, and *append*. A key look up in ZHT can take from 0 (if the key exists in the local server) to 2 network communications. This helps provide the fastest possible look up in a scalable DHT. The following sections discuss main features of ZHT. ZHT operations are highly optimized and efficient. *Insert* and *lookup* operations take less than a millisecond on an average instance on AWS.

1) Network Communication

ZHT supports both TCP and UDP protocols. In order to optimize the communication speed, the TCP connections are cached by a LRU cache. That will make TCP connections almost as fast as UDP.

2) Consistency

ZHT supports consistency via replication. In order to achieve high throughput ZHT follows a weak consistency model after the first two replicas that are strongly consistent [19].

3) Fault Tolerance

ZHT supports fault tolerance by lazily tagging the servers that are not being responsive. In case of failure, the secondary replica will take the place of the primary replica. Since each ZHT server operates independently from the other servers, the failure of a single server does not affect the system performance.

4) Persistence

ZHT is an in-memory data-structure. In order to provide persistence, ZHT uses its own Non-Volatile Hash Table (NoVoHT). NoVoHT uses a log based persistence mechanism with periodic check-pointing [20].

B. Fabriq Overview

Fabriq is a Distributed Message Queue that runs on top of ZHT. It was originally designed for handling the delivery of high message volumes on Cloud environment. Adding an abstract layer over ZHT, Fabriq is able to provide all of the benefits of it including persistence, consistency, and reliability. Running on top of a DHT, Fabriq is able to scale more than 8k-nodes.

Messages in Fabriq get distributed over all of its server nodes. Thus, a queue could coexist on multiple servers. That means clients can have parallel access to a queue on Fabriq, making Fabriq a perfect fit for our framework. Albatross uses a DMQ as a big shared pool of tasks that could provide simultaneous access from a large number of its workers.

Fabriq guarantees exactly-once delivery of the messages. That is an important requirement for Albatross. On our previous work [22], since CloudKon was using SQS [21], and SQS could generate duplicate messages, we had to add an extra component to filter the duplicate messages. That adds more overhead to the system. Using Fabriq, we will not have that problem since we can make sure that it only delivers a message once. Fabriq is very efficient and scalable. The latency of pop and push operations on Fabriq over an Ethernet network on AWS are less than a millisecond on average. That is almost an order of magnitude better than other state-of-the-art message queues.

IV. SYSTEM OVERVIEW

Albatross is a distributed task scheduling and execution framework. It is a multipurpose framework, suitable for various types of workloads and compatible with different types of environments, especially the Cloud environment. The key insight behind the Albatross is that unlike other conventional schedulers, in Albatross, a worker is the active controller and the decision making component of the framework. In most of the schedulers, there is a central or regional component that is responsible for pushing tasks to the

workers and keeping them busy. That could bring a lot of challenges in many cases. In the case of running workloads with high task submission rates, or workloads with heterogeneous tasks, or in the case of larger scale systems, or heterogeneous environments like Clouds, these schedulers will show significant slowdowns. In such schedulers, the scheduler component needs to have live information about the workers that are being fed. That could be a big bottleneck and a source of long delays as the component can get saturated after a certain scale or under a certain load. Albatross gets rid of central or regional scheduler components by moving the responsibility from the scheduler to the workers. In Albatross, the workers pull tasks from a shared pool of tasks whenever they need to run a new task. That could significantly improve the utilization and could improve the load balancing of the system. In order to achieve that, Albatross uses Fabriq as a big pool of tasks. Fabriq is scalable and provides parallel access by a large number of workers. That makes Fabriq a perfect fit for Albatross.

A. Architecture

In order to achieve scalability, it is inevitable to move the control from centralized or regional components to the workers in the framework. In such architecture, the scheduling, routing, and task execution take place by the collaboration of all of the nodes in the system. Each worker has the same part in the workload execution procedure and there is not a single node with an extra component or responsibility.

Figure 1. Albatross Components shows the components of Albatross. The system is comprised of two major components, along with the worker and client driver programs. Fabriq is responsible for the delivery of the tasks to the workers. ZHT is responsible for keeping the metadata information and updates about the workload. This information could include the data location, and the workload DAG. Depending on the setup configurations, an Albatross node could run a worker driver, a client driver, a ZHT server instance, a Fabriq server instance, or a combination of those components. Since ZHT and Fabriq are both fully distributed and do not have a centralized component, they can be distributed over all of the nodes of the system. Therefore, each worker driver program has local access to an instance of Fabriq server, and an instance of ZHT server.

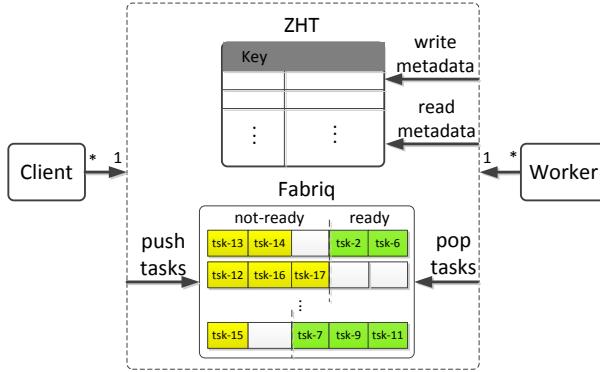


Figure 1. Albatross Components

The process starts from the Client driver. The user provides the job information, and the input dataset to the client

program. Based on the provided job information, the Client program generates the tasks and the workload DAG. Then it distributes the dataset over all of the worker nodes in the system. Finally it submits the tasks to Fabriq. The task submission on Fabriq is performed via a uniform hashing function that distributes the tasks among all the servers, leading to a good system load balance. The data placement and the task submission could also be performed through multiple parallel clients.

The Worker driver starts with pulling tasks from Fabriq. Depending on the type of the task and the workload, a Worker might access, or write into ZHT to either get metadata information about the input data location, or dependencies, or to update the metadata. The worker also fetches the input data either locally or from a remote Worker. We discuss the procedure of data locality support on Albatross in the following sections. The output of each task is written locally.

B. Task Execution Dependency

Both HPC and data analytics workloads enforce a certain execution order among their tasks. In order to be able to natively run those workloads, Albatross needs to provide support for workload DAGs. HPC tasks propose the concept the Bag-of-Tasks. Each HPC job could have some tasks that could be internally dependent on each other. Data analytics workloads often propose similar concepts. Programming models such as Dryad [23], and Map-reduce that are often used in data analytics have similar requirements. In Map-reduce, reduce tasks could only run after all of their corresponding map tasks have been executed.

The implementation of the task execution dependency support should not disrupt the distributed architecture of the system. Our goal was to keep the dependency support seamless in the design and avoid adding a central component for keeping the DAG information. Task execution dependency is supported through the implementation of priority queues in Fabriq. Each task in Albatross has two fields that hold information about its execution dependencies. ParentCount (pcount) field shows the number of unsatisfied dependencies for each task. In order to be executed, a task needs to have its ParentCount as 0. ChildrenList field keeps the list of the taskIDs of the current task's dependent tasks.

Figure 2 shows the process of running a sample DAG on Albatross. The priority inside Fabriq has two levels of priorities: 0 or more than 0. The priority queue inside each Fabriq server holds onto the tasks with non-zero pcounts. A worker can only pop tasks with 0 pcounts. Unlike conventional DMQs, Fabriq provides the ability to directly access a task via its taskID. That feature is used for Albatross task dependency support. Once a task is executed, the worker updates its ChildrenList tasks, decreasing their dependencies by 1. Inside the priority queue, once a task's pcount becomes 0, the queue automatically moves it to the available queue and the task could be popped by a worker.

C. Data Locality

Data locality aims to minimize the distance between data location and respective task placements. Since moving the data is significantly more expensive than moving the process, it is more efficient to move the tasks to where the data is located. Data locality support is a requirement for data-

intensive workloads. Albatross's goal is to minimize the data movement during the workload execution while maintaining high utilization.

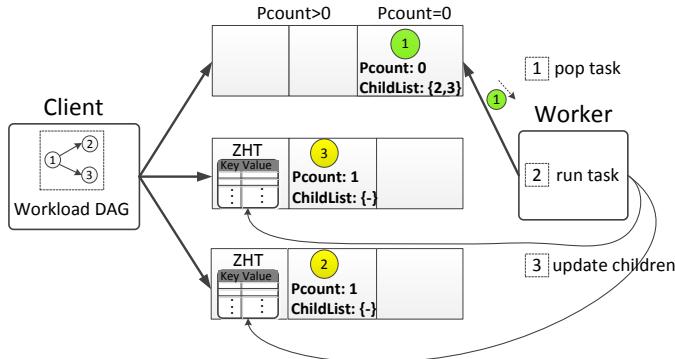


Figure 2. Task Execution Dependency Support

Data locality could be applied on different stages of the workload execution process. The common approach is to dictate the decisions on the scheduling and task placement stage. On this approach, the scheduler tries to send the tasks to their corresponding data location. This approach minimizes the number of task placements before a task gets executed. However, there are some issues with this approach that is going to hurt the overall performance of the system at larger scales. In order to send the tasks to the right location, the scheduler needs to have extensive information about the workers, and the data locations. That could slow down the process at larger scales. Moreover, this approach could lead into load imbalance and reduce the utilization as there could be some nodes with many tasks and some left idle because their corresponding tasks are dependent on the current running tasks on the workload. Also, this method is associated with the pushing approach which is not desirable at larger scales.

Albatross does not dictate any logic regarding data locality at task submission stage. It uses a uniform hashing function to distribute them evenly among servers. That means the task placement is going to be random. The data locality is achieved after the tasks are submitted to the Fabriq servers. Depending on the location of their corresponding data, some of the tasks might be moved again. Even though that adds extra task movement to the system process, it lets the workers handle the locality between themselves without going through a single centralized scheduler. Figure 3 shows the data locality process and its corresponding components. There are two types of queues and a locality engine on each Albatross server. The main queue belongs to the Fabriq. It is where the tasks first land once they are submitted by the client. The locality engine is an independent thread that goes through the tasks on main queue and moves the local tasks to the local queue. If a task is remote (i.e. the corresponding data is located on another server), the engine sends the task to the local queue of its corresponding server via that server's locality engine.

Strict dictation of data locality could not always be beneficial to the system. Workloads usually have different task distribution on their input data. A system with strict data locality always runs the tasks on their data local nodes. In many cases where the underlying infrastructure is heterogeneous, or when the workload has many processing

stages, a system with strict locality rules could have a poorly balanced system where some of the servers are overloaded with tasks and the rest are idle with no tasks to run. In order to avoid that, we incorporate a simple logic in the locality engine.

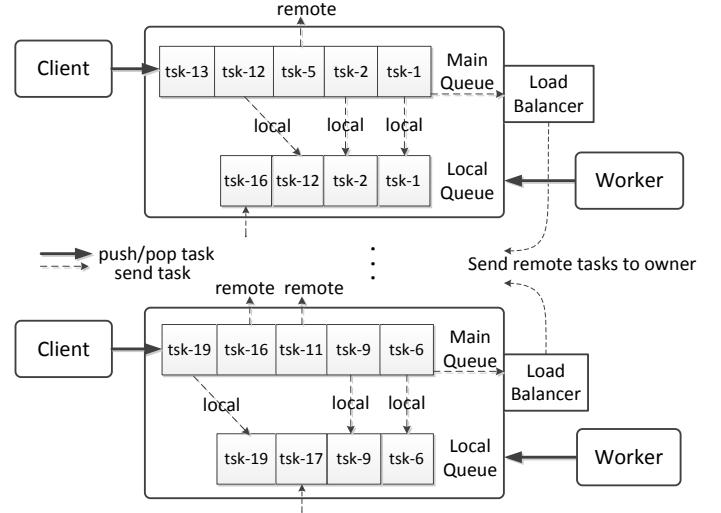


Figure 3. Data Locality Support Components

Figure 4 portrays the locality engine's decision making process. When the engine finds a remote task, it tries to send it to the remote server. If the remote server is overloaded, it rejects the task. In that case, the engine saves the task to the end of local queue regardless of being remote. The worker is going to pop tasks from the local queue one by one. Once it wants to pop the remote task, the locality engine tries to send the task to its own server one more time. If the remote server is still overloaded, the locality engine transfers the corresponding data from the remote server. This technique is similar to the late-binding technique that is used in other scheduling frameworks [8].

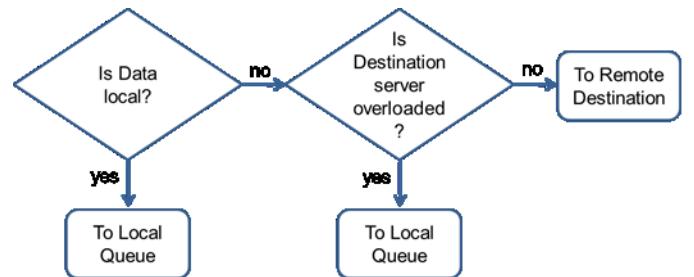


Figure 4. Locality Engine's Decision Making

V. MAP-REDUCE ON ALBATROSS

This section discusses the implementation of the Map-reduce programming model in Albatross. Figure 5 shows the Map-Reduce execution process. Like any other Map-reduce framework, the process starts with the task submission. A task could be either map or reduce. Map tasks have their pcount as 0. They usually have a reduce task in ChildrenList. Reduce tasks have their pcount as more than 0 and will be locked in the queues until their parent map tasks are executed.

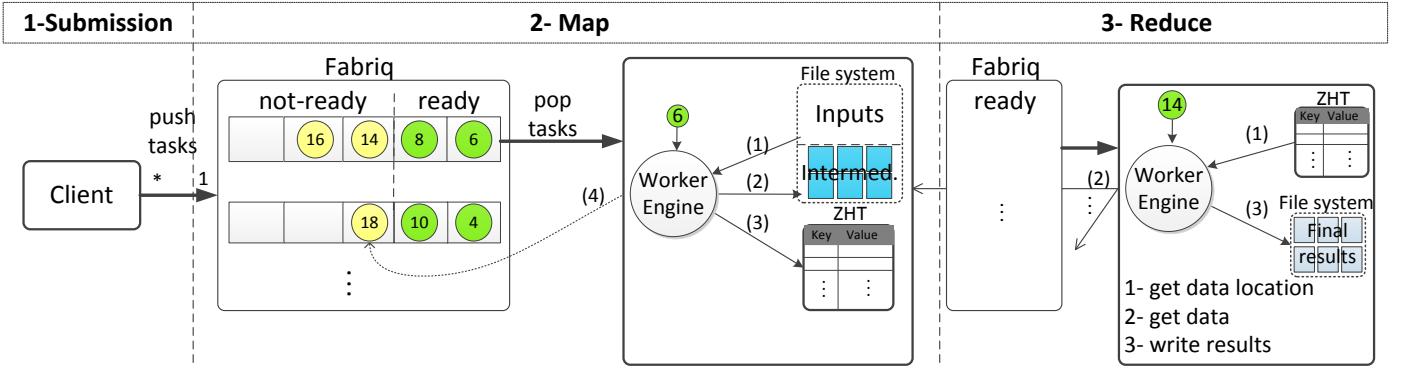


Figure 5. Map-reduce process in Albatross

Once a map task is popped by a worker, it loads its input data from the local (or remote) file system. The map function is included inside the map task. Mapper loads the function and runs it. Unless the size of the output exceeds the available memory limit, the worker writes the intermediate results to the memory. For applications like sort that have larger intermediate data, the worker always writes the output on disk. Once the task is executed, the worker adds the location of the intermediate data for this map task to the ZHT. Finally, the pcount for its dependent reduce task will be reduced by 1.

Once all of the parent map tasks of a reduce task are executed, the reduce task becomes available and gets popped by a worker. The worker gets the location of all of the intermediate data required by this task. Then the worker gets the intermediate data from those locations. Then it loads the reduce function from the task and writes the final results to its local disk. It also adds its final results location to the ZHT.

Many Map-reduce frameworks including Hadoop and Spark have many centralized points in their process. The mappers and the reducers have to go through a single component to get or update information such as the intermediate data location, the logging information and the final output location. Albatross has no centralized point of process in its Map-reduce model. The tasks are delivered through Fabriq and the other information is propagated through ZHT. Since ZHT resides on all of the nodes, mappers or reducers could all access ZHT at the same time without causing a system slow down.

VI. PERFORMANCE EVALUATION

This section analyzes the performance of Albatross. We compare the performance of Albatross using different metrics with Spark and Hadoop which are both commonly used for data analytics. First, we briefly compare the major differences of the three systems in design and architecture. Then, we compare the performance of those frameworks while running microbenchmarks as well as real applications. We measure the efficiency, throughput, and latency of the three systems while varying the granularity of the workloads.

A. Hadoop and Spark

1) Hadoop

Hadoop is a data analytics framework that adopted its architecture from Google's Map-reduce implementation. It consists of two main components which are the distributed file system (HDFS) and the Map-reduce programming paradigm. The two main centralized components of Hadoop are the

NameNode and the JobTracker. The JobTracker is in charge of tracking any disk reads/writes to HDFS. As the job tracker must be notified by the task trackers, similarly the namenode must be notified of block updates executed by the data nodes. In the newer version of Hadoop, instead of the job tracker as seen in Hadoop 1.x, there is a resource manager. The resource manager (similar to the job tracker) is in charge of allocating resources to a specific job [25]. Although the Yarn [24] version tries to provide higher availability, the centralized bottlenecks are still existent.

2) Spark

Spark, like many other state-of-the-art systems, utilizes a master-slave architecture. The flexible transformations enable Spark to manipulate and process a workload in a more efficient manner than Hadoop. One of the vital components of the Spark's cluster configuration is the cluster manager which consists of a centralized job-level scheduler for any jobs submitted to the cluster via the SparkContext. The cluster manager allocates a set of available executors for the current job and then the SparkContext is in charge of scheduling the tasks on these allocated executors. Therefore, similar to Hadoop, centralized bottlenecks are still present even though the capability of iterative workloads is better handled in Spark than in Hadoop. The primary bottlenecks in Spark's cluster mode are the task level scheduler present in the SparkContext and the job-level scheduler present in the provided cluster manager. Other than these pitfalls, Spark provides a novel idea of resilient distributed datasets or RDDs which allows it to provide support for iterative workloads. RDDs are analogous to a plan or a series of transformations which need to be done on a set of data. Each RDD is a step and a list of these steps form a lineage.

B. Testbed and Configurations

The experiments were done on m3.large instances which have 7.5 GB of memory, 32 GB local SSD storage, and Intel Xeon E5-2670 v2 (Ivy Bridge) Processor (2 vCores). Since the amount of vCores available were two, the number of reduce tasks for Hadoop was limited to only two concurrently running tasks.

For the overall execution time experiments (block size fixed at 128 MB), the workload was weakly scaled by 5 GB per added node. For the varied partition/block size experiments, since the runtimes for Hadoop and Spark were very long for very short blocks, the workload was chosen to be 0.5 GB per added node.

There were two main reasons as to why the same 5GB per node workload was not used for the varied partition/block size experiments. Spark and Hadoop started seeing a very long execution time (~4 hours for a single node experiment) and Spark which uses the Akka messaging framework uses a framesize (pool) in which the completed tasks were being stored. As the HDFS block size decreased, the number of total tasks (total number of tasks = total workload / block size) increased to amounts which the default framesize configuration could not handle. Finally, regarding the microbenchmarks, instead of focusing on the size of the input data, we focused on the amount of tasks which should be run per node. As can be seen below, a total of 1000 tasks were run per node.

C. Microbenchmarks

This section compares the scheduling performance of the Albatross and Spark while running synthetic benchmarks. These benchmarks are able to reflect the performance of the systems without being affected by the workloads or applications. We measure latency and throughput while scheduling null tasks. We did not include Hadoop in this section, as Hadoop's tasks are written to disk. That makes the scheduling significantly slower than the other two systems.

1) Latency

In order to assess the scheduling overhead of a framework, we need to measure overall latency of processing null tasks. In this experiment, we submit 1,000 null tasks per node and calculate the total time for each task. The total time could be defined as the time it takes to submit and execute a task, plus the time for saving the results on disk or memory. There is no disk access in this experiment.

Figure 6 shows the average latency of running empty tasks on the three frameworks, scaling from 1 to 64 nodes. Ideally, on a system that scales perfectly, the average latency should stay the same.

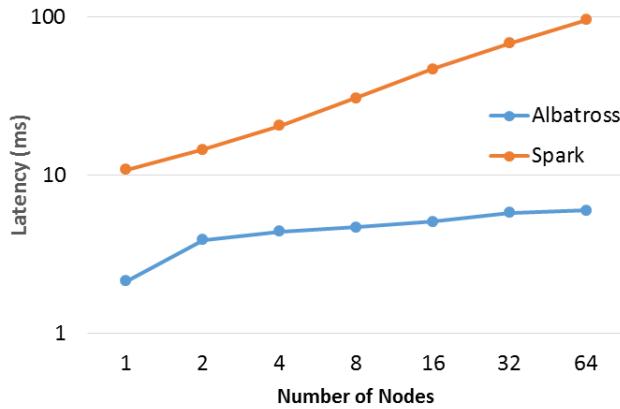


Figure 6. Average latency of in-memory null tasks

In order to fully reflect the scheduling overhead of three frameworks, we need to measure metrics like minimum, median, and maximum latency. Figure 7 shows the cumulative distribution function (cdf) of the three systems while running empty tasks. The cdf is able to show the possible long tail behavior of a system. Compared to Spark, Albatross has a

much shorter range in scheduling tasks. The slowest task took 60 milliseconds to schedule. That is 33x faster than the slowest task in Spark which took more than 2 seconds. This long tail behavior could significantly slow down certain workloads. The median scheduling latency in Albatross is 5 ms as compared to 50 ms latency in Spark. More than 90% of the tasks in Albatross took less than 12 ms which is an order of magnitude faster than the Spark at 90 percentile.

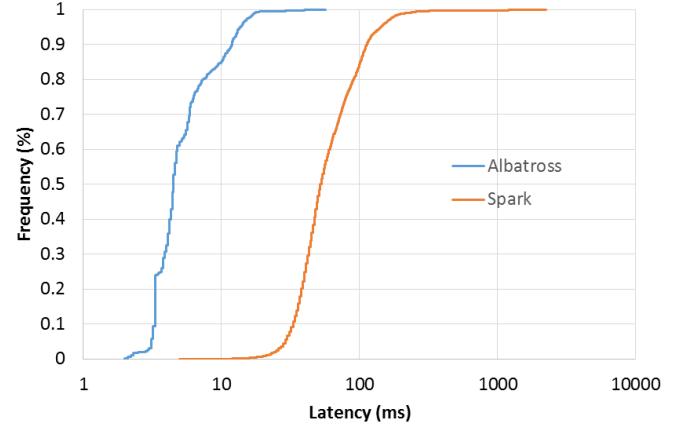


Figure 7. Cumulative distribution null tasks latency

2) Scheduling Throughput

In order to analyze the task submission and scheduling performance of the frameworks, we measured the total timespan for running 10,000 null tasks per node. The throughput is defined as the number of tasks processed per second (tasks per second).

Figure 8 shows the throughput of Albatross and Spark. Spark is almost an order of magnitude slower than Albatross, due to having a centralized scheduler, and being written in Java. Also, unlike Albatross the performance of Spark scheduling does not linearly increase with the scale. Spark's centralized scheduler gets almost saturated on 64 nodes with a throughput of 1235 tasks per second. We expect to see the Spark scheduler saturating at 2000 tasks per seconds. Albatross was able to linearly scale, reaching to 10666 tasks per second at 64 nodes scale.

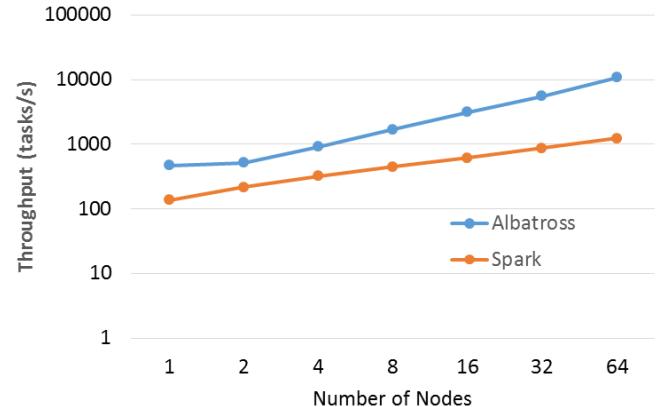


Figure 8. Throughput of null task scheduling

D. Application performance

In order to provide a comprehensive comparison, we compare the performance of the three frameworks while

running different Map-reduce applications. Applications were chosen to reflect weaknesses and advantages of frameworks in different aspects. We have measured the performance for sort, -word-count, and grep applications.

1) Sort: The sort application sorts the input dataset lines according to the ascii representation of their keys. The algorithm and the logic of the sort application is based on the Terasort benchmark that was originally written for Hadoop [25]. Unlike the other two applications, intermediate data in sort is large and will not always fit in memory [31]. The application performance reflects the file system performance and the efficiency of the memory management on each framework. Also, the network communication is significantly longer in this application. As portrayed in Figure 9, Spark utilizes a lazy evaluation for its lineage of transformations. Only when an action such as saveAsHadoopFile is received is when the entire lineage of transformations is executed and data is loaded into memory for processing. Therefore, sortByKey, which is the only transformation in this scenario has a relatively quick “execution time.” On the other hand, the action phases require loading the data in memory, actually performing the sort, and writing back to disk.

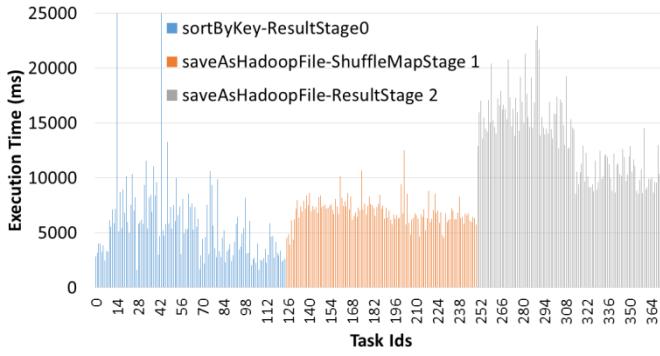


Figure 9. Spark's task breakdown (32 nodes)

As shown in Figure 10, Hadoop’s sort has two phases for the Map-reduce model. The reduce phase is relatively longer since it includes transferring data and the sorting after receiving the data. To allow for Hadoop to utilize a similar “range partitioner” as Spark, we implemented Hadoop’s sort using a TotalOrderPartitioner.

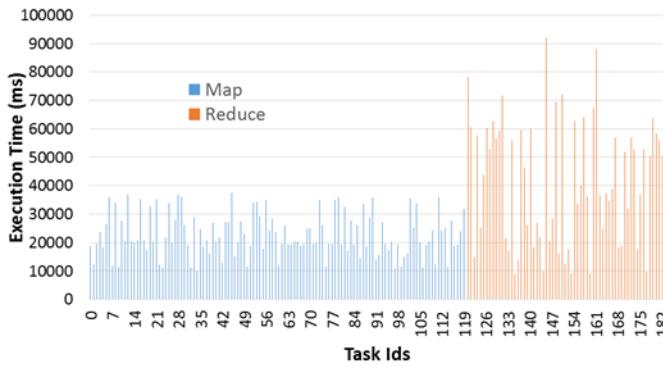


Figure 10. Hadoop's task breakdown (32 nodes)

Figure 11 portrays the map and reduce runtimes for sort application. Similar to Hadoop, Albatross has two phases for sort application. The runtime variation of tasks on Albatross is significantly lower than the Hadoop and Spark. This is a result

of the pulling approach of Albatross that leads to a far better load balancing on the workers.

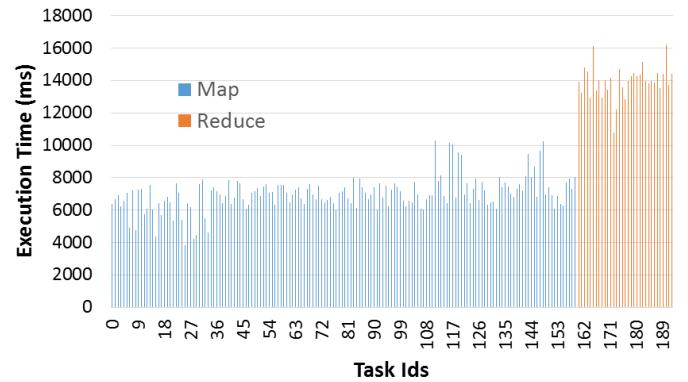


Figure 11. Task breakdown for Albatross (32 nodes)

2) Word-count: The word-count application calculates the count of each word in the dataset. Unlike sort, the proportion of intermediate data to input data is very low. Each map task generates a hash-map of intermediate counts that is not bigger than a few kilobytes. The intermediate results will get spread over all of reducers based on their key-ranges. Similar to Hadoop’s word-count, Spark’s word-count uses map tasks which output (word, 1) pairs and a reducer which aggregates all the (word, 1) pairs by key. The final action is a saveAsHadoopFile which saves the resulting pairs to a file on HDFS

3) Grep: The grep application searches for the occurrences of a certain phrase within the input dataset. The workflow process is similar to the behavior of Map-reduce. However, in Albatross, unlike Map-reduce, the intermediate result of each map task only moves to a certain reducer. That leads to far fewer data transfers over the network. In order to send read-only data along with a task to the executors, Spark encapsulates the read-only data in a closure along with the task function. This is a simple, but very inefficient way to pass the data since all workers will have duplicate values of the data (even though the variable values are the same). Since the grep implementation needs each map task to have access to the search pattern, a broadcast variable which stored the four byte search pattern was used.

We compare the throughput of the frameworks while running the applications. We measure the total throughput based on the ideal block size for each Framework. We also analyze the performance of the frameworks while increasing granularity of the workloads. Figure 12 shows the performance of sort, scaling from 1 to 64 nodes. Albatross and Spark show similar performances up to 16 nodes. However, Spark was not able to complete the workload as there were too many processes getting killed, due to running out of memory. As we mentioned earlier, intermediate results in sort are as big as the input. Using Java, both Spark and Hadoop were not able to handle processing inputs as they were generating large intermediate results. There were too many task restarts on larger scales for Hadoop and Spark. The dotted lines are showing the predicted performance of the two systems if there were not running out of memory and processes were not getting killed by the Operating System. In order to avoid this

problem, Albatross writes the intermediate results to disk when it gets larger than a certain threshold.

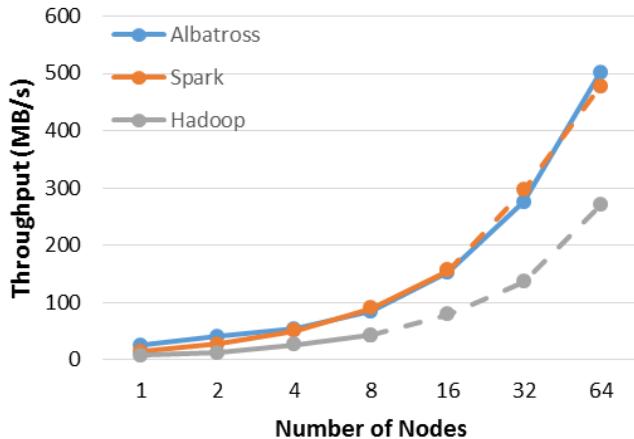


Figure 12. Throughput of sort application

Figure 13 shows the throughput of sort with different partition sizes on 64 nodes. The throughput of Spark is 57% of Albatross using 100MB partitions. However, this gap becomes more significant on smaller partitions. The throughput of Spark is less 20% of Albatross using 1MB partitions. This clearly shows the incapability of Spark on handling high granularity. On the other hand, Albatross proves to be able to handle over-decomposition of data very well. Albatross provided a relatively stable throughput over different partition sizes. The Albatross scheduling is very efficient and scalable and could handle higher task submission rate of the workload. The only exception was for the 100KB partitions. At 100KB, opening and reading files takes the majority of the time and becomes the major bottleneck on the processing of each task. Hadoop and Spark cannot use partitions smaller than 1MB due to the limitation of HDFS.

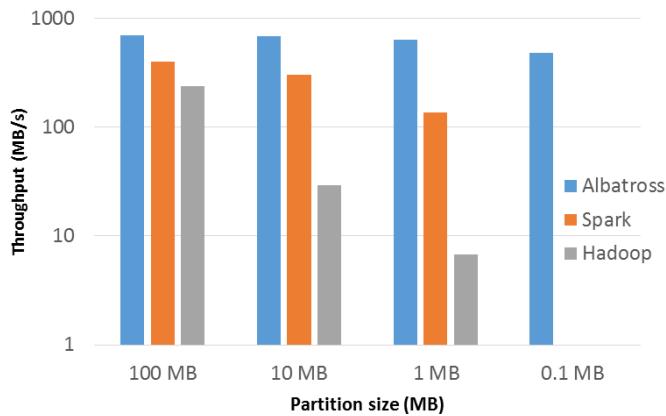


Figure 13. Sort application throughput (varied partition sizes)

Figure 14 shows the throughput for word-count using large data partitions. Spark was able to achieve a better throughput than the other two systems. Even though they have provided different throughputs, all the three systems linearly scaled up to the largest scale of the experiment.

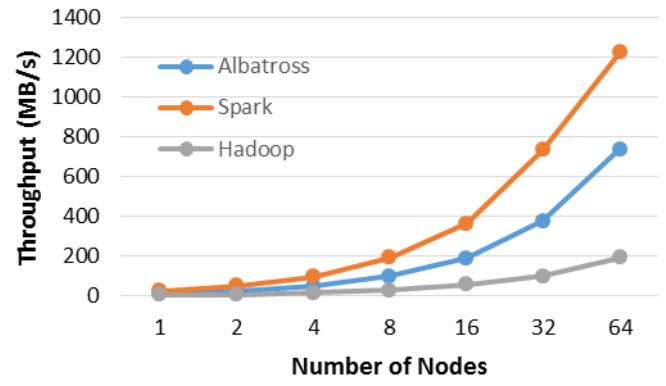


Figure 14. Word-count application Throughput

Figure 15 shows the throughput of word-count using different partition sizes on 64 nodes. Spark outperforms Albatross on 100MB partitions. However, it could not keep a steady performance at smaller partition sizes. Albatross goes from being slightly slower at the largest partition size to outperforming Spark by 3.8x. Spark is not able to schedule tasks at higher task rates. Hence the throughput drops on smaller scales.

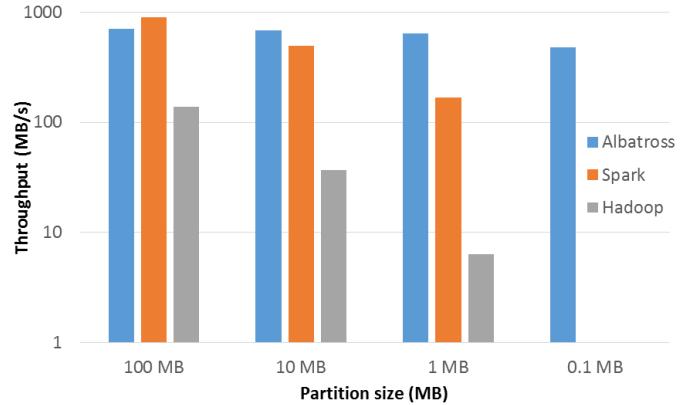


Figure 15. Word-count throughput (varied partition sizes)

Figure 16 compares the performance of grep application on the three systems using large partitions. Albatross outperforms the Spark and Hadoop by 2.13x and 12.2x respectively.

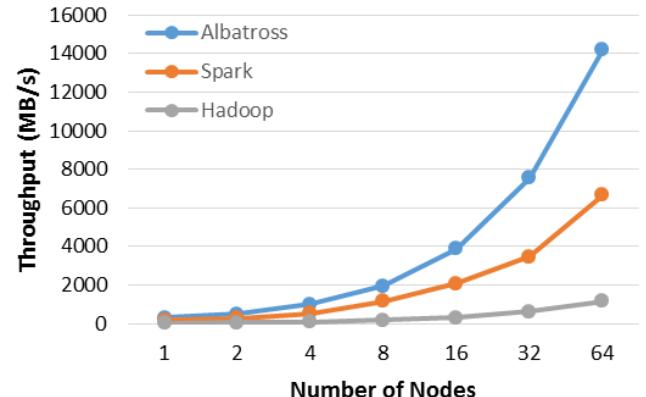


Figure 16. Throughput of grep application

Figure 17 shows the throughput of grep using different partition sizes on 64 nodes. As the partition size gets smaller,

the gap between the throughput of Albatross and the other two systems becomes more significant. Similar to the other applications, the throughput of Albatross is stable on different partition sizes.

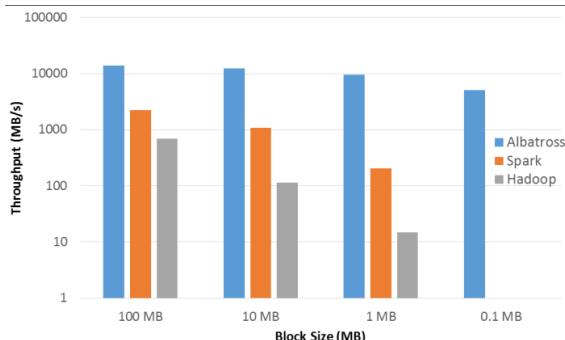


Figure 17. Grep application throughput (varied partition sizes)

VII. CONCLUSION AND FUTURE WORK

Over the past few years, Data analytics frameworks such as Spark and Hadoop have gained a great deal of attraction. With the growth of Big Data and the transition of workloads and applications to high granularity tasks with shorter run time, the requirements for the data analytics has changed. Centralized frameworks will no longer be able to schedule and process the big datasets. There is need for distributed task scheduling and execution systems. This paper proposes Albatross, a distributed task scheduling and execution framework that is able to handle tasks with very high granularities. Albatross uses a task pulling approach as opposed to the traditional scheduling systems. Instead of pushing the tasks to the workers by a central or regional scheduler, in Albatross, workers pull tasks from a distributed message queue system. That leads to a scalable system that could achieve good load balancing and high utilization. Albatross avoids any centralized components in its design. Each node could be a worker, a server, or a client at the same time. The DMQ and the DHT are distributed among all of the nodes in the system. The task submission, scheduling, and the execution are taken place through the collaboration of all of the system nodes.

Our evaluations prove that Albatross outperforms Spark and Hadoop in scheduling microbenchmarks and real applications. As can be seen, both Spark and Hadoop have centralized bottlenecks which become detrimental to their overall performance and system utilization as the task granularity decreases. Although failover options are available for both Spark and Hadoop, this will not be sufficient as even the failover node will not be able to keep up-to-date with the surplus of quick tasks waiting in its queue. Therefore, a tradeoff between task heterogeneity and performance is prevalent in these systems, but Albatross provides the user with the ability to both have heterogeneous tasks (by type and execution time) and consistently good performance. Albatross outperforms Spark and Hadoop in case of running high granular workloads with small data partitions and tasks. The task scheduling rate on Albatross is almost an order of magnitude higher than what Spark could achieve. Albatross was able to provide a high and stable throughput and latency on partition sizes as low as 100KB.

REFERENCES

- [1] M. Wall, "Big Data: Are you ready for blast-off", BBC Business, March 2014.
- [2] M. A. Jette et. al, "Slurm: Simple linux utility for resource management", In Lecture Notes in Computer Sience: Proceedings of JSSPP 2003 (2002), Springer-Verlag, pp. 44-60.
- [3] J. Frey, T. Tannenbaum, I. Foster, M. Frey, S. Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids," Cluster Computing, 2002.
- [4] B. Bode et. al. "The Portable Batch Scheduler and the Maui Scheduler on Linux Clusters," Usenix, 4th Annual Linux Showcase & Conference, 2000.
- [5] W. Gentzsch, et. al. "Sun Grid Engine: Towards Creating a Compute Power Grid," 1st International Symposium on Cluster Computing and the Grid (CCGRID'01), 2001.
- [6] M. Zaharia, et. al. "Spark: Cluster Computing with Working Sets", HotCloud'10.
- [7] T. White, "Hadoop: The Definitive Guide." O'Reilly Media, Inc., 2009.
- [8] K. Ousterhout, P. Wendell, M. Zaharia, and I. Stoica. "Sparrow: distributed, low latency scheduling". Proceedings SOSP '13.
- [9] I. Sadooghi, K. Wang, S. Srivastava, D. Patel, D. Zhao, T. Li, and I. Raicu, "Fabriq: Leveraging distributed hash tables towards distributed publish-subscribe message queues," IEEE/ACM International Symposium on Big Data Computing (BDC), 2015.
- [10] T. Li, X. Zhou, et. Al. "ZHT: A light-weight reliable persistent dynamic scalable zero-hop distributed hash table," in Proceedings of the IEEE IPDPS, 2013.
- [11] L. Thai, B. Varghese, and A. Barker, "Executing bag of distributed tasks on the cloud: Investigating the trade-offs between performance and cost," Proceedings of CloudCom, 2014.
- [12] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, et. al. Mesos: A Platform for Fine-grained Resource Sharing in the Data Center. In Proceedings of NSDI'11, USENIX Association, pp. 295–308.
- [13] M. Schwarzkopf, A. Konwinski, M. Abd-ElMalek, and J. Wilkes. Omega: flexible EuroSys '13, pages 351–364, New York, NY, USA, 2013. ACM.
- [14] M. Frigo, et. al, "The implementation of the Cilk-5 multithreaded language," In Proc. (PLDI), pages 212–223. ACM SIGPLAN, 1998.
- [15] R. D. Blumofe, et. al. "Scheduling multithreaded computations by work stealing," In Proc. 35th FOCS, pages 356–368, Nov. 1994.
- [16] V. Kumar, et. al. "Scalable load balancing techniques for parallel computers," J. Parallel Distrib. Comput., 22(1):60–79, 1994.
- [17] J. Dinan et. al. "Scalable work stealing," In Proceedings of the HPDC, 2009.
- [18] K. Wang, A. Rajendran, and I. Raicu. "MATRIX: Many-task computing execution fabric at exascale". 2013. <http://datasys.cs.iit.edu/projects/MATRIX/index.html>
- [19] T. Li, et. al. "A Convergence of Distributed Key-Value Storage in Cloud Computing and Supercomputing", Journal of Concurrency and Computation Practice and Experience (CCPE) 2015.
- [20] K. Brandstatter, T. Li, X. Zhou, I. Raicu. Novoh: a lightweight dynamic persistent NoSQL key/value store. In: GCASR' 13, Chicago, IL 2013
- [21] Amazon SQS, [online] 2014. <http://aws.amazon.com/sqs/>
- [22] I. Sadooghi, S. Palur, et. al. "Achieving Efficient Distributed Scheduling with Message Queues in the Cloud for Many-Task Computing and High-Performance Computing", Proceedings of CCGRID, 2014.
- [23] M. Isard, M. Budiu, Y. Yu, A. Birrell, D. Fetterly. "Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks", Euro. Conf. on Computer Systems (EuroSys), 2007.
- [24] V.K. Vaishpal, "Apache Hadoop Yarn – Resource Manager," <http://hortonworks.com/blog/apache-hadoop-yarn-resourcemanager/>
- [25] M. Noll. (2011, April) Benchmarking and Stress Testing an Hadoop Cluster With TeraSort, TestDFSIO & Co. [Online].
- [26] Amazon Elastic Compute Cloud (Amazon EC2), Amazon Web Services, [online] 2013, <http://aws.amazon.com/ec2/>
- [27] I. Sadooghi, J. Martin, T. Li, I. Raicu, et. al. "Understanding the Performance and Potential of Cloud Computing for Scientific Applications", IEEE Transactions on Cloud Computing (TCC), 2015
- [28] I. Raicu, I. Foster, and Y. Zhao, "Many-task computing for grids and supercomputers", In proceedings of MTAGS 2008,
- [29] K. Wang, K. Qiao, I. Sadooghi, et. al. "Load-balanced and locality-aware scheduling for data-intensive workloads at extreme scales", Journal of Concurrency and Computation Practice and Experience (CCPE) 2015.
- [30] T. Li, I. Raicu, L. Ramakrishnan, "Scalable State Management for Scientific Applications in the Cloud", BigData 2014
- [31] H. Eslami, A. Koukas, et. al. "Efficient disk-to-disk sorting: a case study in the decoupled execution paradigm." Proceedings of Workshop on Data-Intensive Scalable Computing Systems, 2015.

Prediction of workflow execution time using provenance traces: practical applications in medical data processing

Hugo Hiden

School of Computing Science
Newcastle University
Newcastle upon Tyne, UK
Email: hugo.hiden@ncl.ac.uk

Simon Woodman

School of Computing Science
Newcastle University
Newcastle upon Tyne, UK
Email: simon.woodman@ncl.ac.uk

Paul Watson

School of Computing Science
Newcastle University
Newcastle upon Tyne, UK
Email: paul.watson@ncl.ac.uk

Abstract—The use of cloud resources for processing and analysing medical data has the potential to revolutionise the treatment of a number of chronic conditions. For example, it has been shown that it is possible to manage conditions such as diabetes, obesity and cardiovascular disease by increasing the right forms of physical activity for the patient. Typically, movement data is collected for a patient over a period of several weeks using a wrist worn accelerometer. This data, however, is large and its analysis can require significant computational resources. Cloud computing offers a convenient solution as it can be paid for as needed and is capable of scaling to store and process large numbers of data sets simultaneously.

However, because the charging model for the cloud represents, to some extent, an unknown cost and therefore risk to project managers, it is important to have an estimate of the likely data processing and storage costs that will be required to analyse a set of data. This could take the form of data collected from a patient in clinic or of entire cohorts of data collected from large studies. If, however, an accurate model was available that could predict the compute and storage requirements associated with a piece of analysis code, decisions could be made as to the scale of resources required in order to obtain results within a known timescale.

This paper makes use of provenance and performance data collected as part of routine e-Science Central workflow executions to examine the feasibility of automatically generating predictive models for workflow execution times based solely on observed characteristics such as data volumes processed, algorithm settings and execution durations. The utility of this approach will be demonstrated via a set of benchmarking examples before being used to model workflow executions performed as part of two large medical movement analysis studies.

I. INTRODUCTION

Historical and ongoing research projects are investigating the links between levels of physical activity and chronic conditions such as Type II Diabetes and Myotonic Dystrophy Type I^{1,2}. The most common method for monitoring the physical activity of a subject is via the use of wearable devices such as accelerometers. Typically these take the form of a wristwatch that captures data at 100Hz over a period of several

weeks. Devices such as the GENEActiv³ and Actiwatch⁴ are commonly deployed and a wide body of research has been published using data captured in this way [1].

Movement data takes the basic form of a long timeseries containing three distinct channels of data captured from three perpendicular axes (X, Y & Z). An analysis of the patterns within this data can give an insight into a number of underlying conditions: Activity level [2], sleep quality [3] and, gait [4] which can be used to indicate quality of life, aid in assessments and measure rehabilitation of some medical conditions.

In conjunction with this movement data capture, in many studies, patients attend clinics periodically in order to discuss progress with a clinician. They will often have other measurements, such as weight, blood glucose etc. taken and their physical activity inspected.

The issue, however, is that as the wrist worn accelerometers measure movement data over three axes at approximately 100Hz for many days, the quantity of data to be analysed is large - a typical data file is approximately 800MB in size and comprises some 100 million rows of data. Once collected, the analysis procedure [5] for this data involves categorizing the acceleration signals into one of several categories for instance: Sedentary, Light Activity, Walking and Running. This is then used to make recommendations as to suitable exercise plans for that specific patient. Further analysis may also be conducted [6] which can answer different research questions such as whether the patients sleep is affected by a change in medication or investigate exercise patterns across a large population.

Although the task of processing data for a single patient is tractable, clinical studies have collected movement data from large numbers of participants and analysing this data requires larger scale facilities. The sizes of three studies are shown in Table I

Clearly, given an hour of typical processing time for a simple analysis, the task of processing data from a complete

¹<http://www.directclinicaltrial.org.uk/>

²<http://optimistic-dm.eu/>

³<http://www.geneactiv.org>

⁴<http://www.philips.co.uk/healthcare/product/HC1046964/>
actiwatch-spectrum-activity-monitor

TABLE I
MOVEMENT STUDY SIZES

Study	Participants	Data Size
Newcastle 85+	1000	300GB
Whitehall	4300	4TB
UK Biobank	100k	24TB

study is not a trivial one. For example, using the information shown in Table I, an analysis of the Whitehall study using an algorithm such as PAC1 [5] would take approximately six months of continuous computation on a single PC ignoring any data preparation and transformation required. This raises issues regarding the both provision of appropriate resources to complete an analysis in an acceptable time frame and also keeping track of the analysis process and software versions used.

This tracking requirement is particularly important as movement analysis algorithms are the subject of active research with improvements and fixes published frequently. For example, one popular algorithm [6] has seen thirteen releases in a two year period in order to fix bugs and increase functionality.

The use of workflows to coordinate the analysis of data generated in these projects is increasingly common. The developers of algorithms such as PAC1 and GGIR are experts in classifying movement based on accelerometry. However, in the context of a long running study there are usually study management issues such as participant tracking and reporting, and other associated data preparation tasks. Workflow engines mitigate some of these issues by providing suites of services used in traditional Extract, Translate, Load pipelines [7] which are convenient in this context too. The availability of these services allows the algorithm developers to focus on the medical aspects of their work and use in-built tools for the other tasks. The framework that the workflow will execute within usually also deals with concurrent executions and dependency resolution, again relieving the algorithm developer from having to consider these issues.

The e-Science Central cloud data analytics platform is an Open Source multi-user system for the storage and analysis of a wide range of data sets [8]. It provides data storage and sharing facilities along with a workflow engine designed to operate at large scale within a hosted cloud environment.

Within clinical and pre-clinical trials, usually a data analyst within the project team will determine the algorithmic needs of the study and develop the workflows which will generate reports. During the study the staff collecting the data, often in a clinic, will upload that data into e-Science Central where the workflow developed by the analyst will process it. When these workflows are executed, they are queued for execution by one of several servers dedicated to the task of running workflows (Workflow Engines). In order to maintain a fair access to these Workflow Engines, it is important to balance the calculation time between all of the users of the system and to schedule workflow execution over a range of resources. Predicting the

runtime of any given workflow, therefore, is a vital first step towards achieving this and is also critical for the provision of the estimated cost and storage requirements associated with running a trial. Additionally, because workloads on the system can require the addition of significant numbers of workflow engines, a prediction of the likely end time for these workloads would enable workflow engines to be brought online proactively and shut down in a more timely manner. This is particularly pertinent given the uneven processing requirements in these studies - devices are usually recovered sporadically when clinics run and lead to higher processing requirements at these times. At other times there will be little or no requirement for processing resources.

This paper describes a system which captures live performance data and uses it to build a suite of models that can be used to predict various characteristics of workflows. These models can be updated on demand or in response to the collection of a sufficient quantity of additional logging data.

Although the performance modelling work presented in this paper was carried out using the e-Science Central platform, the methodology and indeed the code developed is applicable to any system that can be instrumented to produce the correct form of performance logging data. The contributions of this paper fall largely into three areas:

- 1) Estimating the future performance of software components based on predictive models trained on historical data.
- 2) The ability to combine a number of models of individual unit operations in the same order as they would be invoked in arbitrary workflows and the ability to predict the likely performance of these workflows.
- 3) A system to capture and store historical performance data and generate suites of predictive models from it.

This paper is structured as follows: The following Section provides context and compares the work in this paper to previously published research. Section III gives an overview of the architecture of e-Science Central and how the performance data is captured. The process for generating and managing predictive models is described in Section IV and these are evaluated against medical workloads in Section V. Finally, conclusions are drawn in Section VII.

II. RELATED WORK

A significant quantity of research has been performed in an attempt to predict the execution time of software in order to improve scheduling, particularly within Grid and HPC environments. Some researchers have considered complete applications [9]–[11] whilst others have attempted to decompose these into components as we do [12], [13]. The work presented in by Duan [12] is of particular interest as it closely resembles ours but focuses on Grid deployment scenarios. One of the key differences is our use of the ‘Panel of Experts’ pattern [14] to generate multiple predictive models of each service rather than their use of a Radial Basis Function neural network. We have found that it is imperative to include multiple modelling

techniques as some components will model significantly better or worse depending on the technique used.

The work by Cushing [15] discusses how to scale Map-Reduce style problems based on the expected execution time. The aim here is to reduce the overall computation time by dedicating more resources to components which are expected to take a longer duration. In addition they aim to prevent starvation of future components due to a previous one having not completed execution. We do not restrict our execution pattern to Map-Reduce, although we are able to construct such a pattern using e-Science Central workflows.

Within the context of cloud computing, Roy [16] use autoregressive moving averages to predict the current workload of a system. However, they are concerned with scaling cloud architecture to minimise response time in a web application rather than scientific workflow applications which exhibit different characteristics.

Work by Miu [17] considers other features of the input data other than the size when generating predictive models for algorithms such as those found in the Weka toolkit. This work is, in some respects, more ambitious than ours but also more expensive in terms of computing power required and knowledge of the algorithm being modelled. We consider each service as a black box and make no attempt to include features other than the size of the input data and the code configuration parameters within our models. In future we would like to encompass other features of the input data but this would increase the complexity of the system. In addition we have found that many of the services used within scientific workflow applications deployed within e-Science Central to date are amenable to simple analysis based on the size of the input data including physical activity analysis and image processing algorithms [5], [6], [18].

The work around the Prophesy project to develop a general purpose performance analysis system is most similar to our approach [19]. However, they focus on lower level instrumentation of the source code than we do where our ‘building blocks’ are workflow services. Further, they require the source code to be available in order to insert the performance monitoring hooks during the compilation phase. Instead, our work has focused on instrumenting the execution environment to allow any code deployed into the environment to benefit from performance capture.

The literature around the Prophesy system also details some approaches to leveraging multiple predictive models to generate a prediction for a larger unit of work [20]. As their work is principally aimed at lower level functions with more complex inter-relationships they generate what can be considered to be a cross-product of model relationships between each ‘kernel’ of computation. Our approach differs in that we only consider the effects of the data transferred from one component to another and, given that we are dealing with higher level components without such inter-relationships, we do not need to compute the cross-product of all components. We also show that it is feasible to use the output of one predictive model as the input to another whereas other systems simply consider the

summation of the predictions from each model [21].

III. ARCHITECTURE

e-Science Central is a portable ‘platform-as-a-service’ that can be deployed on a variety of hardware platforms ranging from a Raspberry Pi to public/private clouds and supercomputing infrastructures. Cloud computing has the potential to give scientists the computational resources they need, when they need them. However, cloud computing does not make it easier to build the often complex, scalable secure applications needed to support scientists. e-Science Central was designed to overcome these obstacles by providing a platform on which users can carry out their research, and build high-level applications. Using a web browser, users can upload data, share it in a controlled way with colleagues, and analyse the data using either a set of pre-defined blocks, or their own, which they can upload or create online. A range of data analysis and programming languages are supported, including Java, Javascript, R and Octave.

The blocks which are hosted within e-Science Central can be combined into larger units of computational work, workflows, which compose multiple re-usable components to perform data analysis. Workflows in e-Science Central only include data flow – control flow, outside of each block, is not provided. However, as workflows are able to execute other workflows a simple recursive structure can be described but the termination condition must be expressed within a specialised block. We have found through extensive work with scientists in varying application areas that data flow alone is sufficient to suit their needs [22]. The benefit of supporting pure data flow, as we will see in Section IV, is that it greatly simplifies the process of generating predictive models of the workflow execution time. Workflows are created using an online editor which supports drag and drop workflow creation from a palette of both common and user supplied blocks. Blocks themselves can be created using another online editor or common software development tools such as Maven.

Versioning is an integral storage feature in e-Science Central, allowing users to work with old versions of data, blocks and workflows. All objects (data, files and workflows) are automatically versioned when they are updated by the user. From the perspective of modelling blocks, this allows us to directly compare the execution time of different versions of the same block and use the previous version when insufficient data is available for the current version of the block (see Section IV-B). In addition to each block being versioned, they may be parameterised with different runtime configurations. Such parameters include the source of the data to import, initialisation parameters for algorithms and other runtime settings. These parameters are included in the data collected for model construction wherever possible (specifically when the parameter has, or can be represented as, a numerical, non-categorical value).

Workflows are enacted by a set of workflow engines which typically run on separate machines from the main e-Science Central server. A workflow within e-Science Central differs

from the more traditional workflow model in that the data flow it represents is executed entirely on the workflow engine and does not typically make calls to external services. The individual blocks within the workflow are separate code libraries that are downloaded to the workflow engine (or potentially installed using the operating system package manager) where they then operate upon the data generated by the workflow execution. Because the workflow engines themselves perform all of the computational tasks required in order to execute a workflow, adding more workflow engines increases the processing power of the system. Depending on the characteristics of the workload, we have been able to support over 200 workflow engines using a single server. Each workflow engine executes a workflow by analysing the directed acyclic graph which represents it. From this, a sequence of block executions is constructed which allows the engine to execute them in an order which ensures that the input data is available for each block within the workflow when required.

Each time a block in a workflow is executed, a provenance message is sent to a queue for persistence in the provenance database. Currently, the provenance message for a workflow block includes details of the code used within the block, the data sets the block operated on, the configuration of the block and the user running the workflow. This information was selected because it contains sufficient data to recreate the actions performed on a given piece of data within the system [23]). During the development of the performance modelling system described in this paper, the provenance capture platform was extended to include performance data which was stored in a separate provenance database. The approach adopted was to log all of the available parameters of the execution of blocks within workflows that could possibly be used to predict performance. Specifically, the following performance attributes were logged in the performance database:

- **Execution time** This is the total time taken for a block within the workflow to execute. The time is measured from the time that the process executing the block is started to the time it terminates. This time measurement does not include the time taken by the workflow engine to deploy any code that the block depends on and is therefore a direct measurement of code execution time and not a combination of code execution time and workflow management overhead.
- **Block settings** Each block within the workflow can be configured with a number of settings. These settings are block specific and can have a significant influence on the time taken for a block to run. For example, a modelling block might include a parameter for specifying the model complexity. This would have a dramatic effect on the block execution time. The performance capture system logs any numerical block property in the performance database. References to data stored within the e-Science Central file system are treated slightly differently in that the size of the document is logged as a parameter in the performance database. The data capture was limited to

numerical properties in this case because the modelling tools selected do not operate on non-numerical data. If, however, classification algorithms that consider categorical data were found to yield useful predictions, additional block settings could be captured trivially.

- **Data volumes** The volume of data consumed by each block is a critical parameter for modelling execution time and is captured in the performance database where it is linked to individual block inputs and outputs. Model building algorithms can then access the information about the total volumes of data passing through workflow blocks.
- **Machine characteristics** The type of the machine executing workflow blocks is logged because it enables block executions to be grouped by machine type when building models. The actual machine data (CPU speed, memory type etc) is not used for modelling as it is impossible, with the current version of e-Science Central, to know in advance which engine within the execution pool will execute a given workflow. It can, however, be used to select the appropriate model to use to estimate workflow termination time once it has started and the exact characteristics of the selected workflow engine are known.

The data within the Performance server is consumed asynchronously from a JMS queue which prevents it from affecting the performance of the system that it is monitoring and allows multiple sources of performance metrics to send data concurrently. When received, the data is persisted into a Postgres database. In order to generate the models we make use of the Apache Commons Maths 3 library and a JavaEE application server to host the code. This simplifies access to the performance database and the provision of the various APIs which allow the data to be consumed by other systems. The models, once constructed, are stored in the database and allow predictions to be generated and comparisons between different models of same block to be made.

The actual selection of the parameters used in any model is delegated to the specific model building algorithms deployed within the system. This approach was adopted because the modelling system has been designed to build multiple predictive models for each workflow block, each of which is likely to include a different subset of the performance data contained in the database. The capture of this comprehensive set of parameters also allows models to be built of properties other than execution time. For example, the data captured could allow models to be built relating the physical RAM consumed by a block to the quantity of data processed.

There are two ways that the models and predictions can be consumed by other interested parties: a simple web application is provided to allow users to view performance data and generate predictions whilst a REST based API allows integration with the core e-Science Central server and other external systems.

Although currently we are only concerned with modelling performance data collected from e-Science Central workflows,

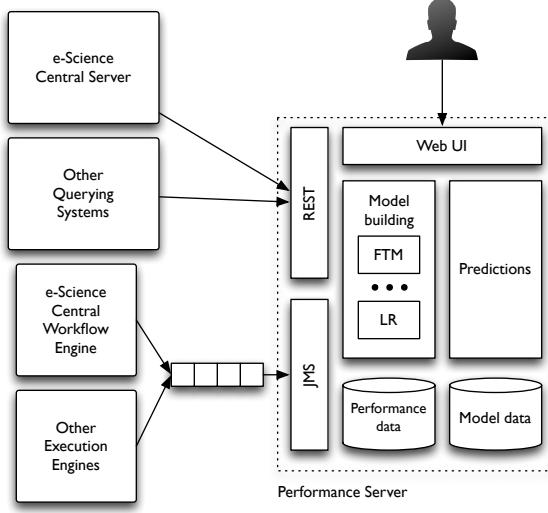


Fig. 1. Architecture of the Performance Server

the architecture is generic and can interface with other systems in order to generate and consume performance data. For instance, workflow enactors such as Taverna or Galaxy could be instrumented to submit performance data into the system. The only modification required would be to the logic for combining the predictive models generated for each block or action within the workflow. Indeed it is not limited to capturing workflow based execution data: any system which is able to log the performance characteristics of a task could submit data. In addition, we have integrated the prediction code into the e-Science Central workflow editor to allow real time feedback to users as they are creating their workflows.

IV. MODELLING PERFORMANCE

Within e-Science Central, workflows can be considered as a linked set of individual software components (blocks) which act sequentially upon items of data. Execution proceeds in the following manner:

- 1) The workflow is analysed to discover all of the blocks that contain no input connections. These are defined as data source blocks that act to bring data to the server hosting the workflow engine.
- 2) Once the data source blocks have been identified, an execution thread is started which starts from the first data source block and propagates data through all of downstream blocks, which are executed in an order which ensures that all of the required input data items for each block have been produced by any linked upstream blocks.
- 3) Execution terminates once all of the possible execution paths from the data source blocks have been traversed.

The basic structure of an e-Science Central workflow is illustrated in figure 2 which shows a number of connected blocks ($B_1 \dots B_n$). Each of these blocks can contain a property set that defines its behaviour ($P_1 \dots P_n$). The analysis pass of the workflow execution process will identify B_1 as the single

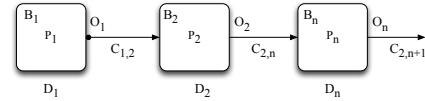


Fig. 2. e-Science Central workflow structure

data source block. The execution thread will first execute B_1 using the property set, P_1 . This will take a period of time, D_1 , and produce a piece of output data, O_1 . This data item will be propagated to the second block in the workflow, D_2 , along the connection, $C_{1,2}$. The second block, B_2 , will then be executed using its property set, P_2 , and input data set, O_1 . This process will take D_2 seconds.

From this it can be seen that the total actual execution duration for the workflow, D_{wf} can be expressed as:

$$D_{wf} = \sum_{i=1}^n D_i \quad (1)$$

To generate an estimated execution duration for the workflow, \hat{D}_{wf} , a summation of duration estimates for the individual workflow blocks is therefore required:

$$\hat{D}_{wf} = \sum_{i=1}^n \hat{D}_i \quad (2)$$

This approach is applicable to the e-Science Central workflow engine because it does not attempt to execute any blocks in parallel, so the total execution time can easily be calculated. For cases where workflow paths can be operated in parallel, the longest duration for each parallel path must be summed in order to predict the total execution time. In order to generate a prediction of the execution duration for a particular block, \hat{D}_i , a relationship needs to be defined that relates execution duration to the various attributes of the block that can influence performance. In general the execution duration for a block will be a function of the input data size to the block, O_{i-1} , the actual code within the block and the block parameter settings, P_i (see Section III). The estimated duration of any block within the workflow can therefore be calculated using:

$$\hat{D}_i = f_{Di}(P_i, O_{i-1}) \quad (3)$$

where f_{Di} represents the predictive duration model for the i^{th} workflow block. From this, it follows that the total workflow duration can be predicted by:

$$\hat{D}_{wf} = \sum_{i=1}^n (f_{Di}(P_i, O_{i-1})) \quad (4)$$

Because the duration estimate for each block within the workflow is dependent upon the size of the data flowing into it, the process of estimating the duration of a multi-block workflow is complicated by the fact that, for non data source blocks (i.e. most blocks within the workflow) a value for the input data size must also be estimated. If the output data size

for a block is assumed, like the duration estimate, to be a function of the input size (O_{i-1}) and the block settings (P_i), the output data size for a given block within a workflow can be modelled using:

$$\hat{O}_i = f_{O_i}(P_i, \hat{O}_{i-1}) \quad (5)$$

Where f_{O_i} represents the predictive output size model for the i^{th} workflow block. During the process of producing a duration estimate for an entire workflow, this size estimate is propagated throughout the workflow in place of the actual data sizes. It follows, therefore that as the size of the workflow increases, the model prediction will be degraded by both the errors in predicting the duration of each block and also the errors accumulated by propagating size estimates to each duration prediction. The availability of accurate models which can predict the quantity of data produced by executing individual blocks is therefore central to accurately estimating total workflow execution time. Section V will investigate whether, for the workflows examined in this paper, this is indeed the case.

A. Model Types

The relationship between block duration and observed execution data for blocks within an e-Science Central workflow can fall into one of three broad categories:

- 1) The block duration can be estimated using a linear combination of the execution data contained within the performance database. In this case a simple linear model of the form $y = mx + c$ can be used to estimate the execution duration.
- 2) The block duration follows a non-linear relationship between execution time and the captured performance data. In this case one of a number of non-linear models (for example, polynomial regression or a neural network) can be used to estimate execution duration. During our experiments, none of the deployed blocks exhibited a non-linear relationship so these model types were not considered. It should, however be noted that some of the blocks studied in this paper could eventually demonstrate a non-linear relationship as larger data volumes are processed. In this situation it would be fairly straightforward to add additional model types to the performance modelling system allowing non-linear duration models to be constructed.
- 3) The block duration exhibits no correlation to any of the observed execution data. In this situation, the duration prediction for the block is modelled as the average execution time for all observed executions of the block contained within the performance database.

The performance modelling system can maintain models for each version of each block observed during workflow executions and generate duration predictions using the most appropriate model on demand. This requires models to be managed (Section IV-C) and also the facility to generate some sort of prediction even in situations where the quantity

of observed data is insufficient to create one of the models described above (Section IV-B).

B. Model Fallbacks

One of the key requirements for the performance modelling application is to provide a robust prediction of performance properties that are refined as more data becomes available. Therefore, a number of fallback predictions are provided to cater for situations where models are unavailable for a block. This could be because a block has never been executed or that the data collected at the current time is unsuited to generating predictions. The following fallback logic has been implemented:

- 1) If there is no model available for a specific version of a block a version agnostic model is used. This model is constructed from all of the executions of a block and covers data collected from all versions.
- 2) If there are no models of any sort for a block, but there is at least one observation for a block, average values for execution duration and output size will be used.
- 3) If there is no data of any sort for a block, the average duration for all blocks will be used and the average output data size will be used for predicting output sizes.
- 4) If the system has just been initialised and no data of any type is present, no prediction will be returned.

The reasoning behind the above logic is to return a prediction wherever possible and to always return the best prediction that the system can provide at a given point in time. Predictions returned are marked with a quality flag which indicates whether the prediction has been generated using data collected for the correct version of a block or whether any fallback predictions have been used.

C. Model management

Because the nature of a block cannot generally be determined *a priori*, the performance modelling system must be able to determine automatically whether the execution duration of a block is linear, non-linear or uncorrelated with respect to the observed data. In order to achieve this, the system builds every type of model contained within its library for each block. In the experiments presented in this paper, this involved building linear and mean predictor models at each model update step. This pattern has been adopted for some earlier chemical modelling work [14] and has been referred to as the “panel of experts” approach. Once built, the model demonstrating the best performance on a set of test data is used to generate duration predictions until the next model update step. During the generation of the results shown in Section V, the models were updated only once, after the initial data sets had been collected. In an actual deployment, an automatic model updating strategy would be required. This could be triggered, for example by the availability of a significant quantity of new observations, an increase in model prediction error or the age of the models within the library passing some threshold. Regardless of the strategy adopted, the process of rebuilding the models is identical: a model update message is

sent to the performance server which then initiates a number of model update threads. These threads rebuild the models without requiring an interruption to the data collection process.

V. EXPERIMENTAL RESULTS

In order to demonstrate the suggested approach to modelling workflow performance, a number of experiments were performed. Initially, these focused on running simple workflows containing a set of trivial blocks under ideal conditions to investigate whether it was indeed possible to generate reliable performance models. Experiments were then performed using the same simple workflows in a more challenging Cloud environment (Amazon EC2) to establish the level of inaccuracies introduced by operating within a multi-tenanted environment. The next set of experiments focused on running a workflow containing blocks performing more complex work. These were performed in Amazon EC2.

A. Simple workflow tests

The workflow studied in the initial experiments contained ten simple Java blocks that are provided with every installation of e-Science Central. These blocks perform basic data manipulation tasks and are therefore more IO than CPU intensive. As such, the execution of these blocks is likely to be very highly correlated to the data volumes being passed through them. The workflow used is shown in figure 3 and contains the following basic blocks:

All models built during these experiments were compared using the Root Mean Squared Error measurement (RMSE), and the correlation (r^2) between the predicted and observed execution durations.

1) Experiment 1: For the first experiment, the workflow shown in figure 3 was executed 250 times with a set of randomly generated input files ranging in size from 7KB to 14MB. These files were pre-generated and one was selected at random for each of the 250 executions. The same sets of data were transferred to Amazon EC2 for the second experiment. The results of this experiment demonstrated that:

- 1) It was possible to generate an accurate prediction of the volumes of data produced by each of the blocks studied. For example, the volume of data produced by the Import File block is predicted with almost 100% accuracy using a linear model with the size of the imported file captured as a block property as it's single input variable.
- 2) For the majority of blocks, it was possible to generate an accurate prediction of execution time based upon the size of the input data and the captured block configuration parameters. For example, figure 4 shows the duration prediction model for the Shuffle block ($RMSE=0.344, r^2=0.999$).
- 3) For some blocks, the duration model, at the data sizes tested, did not generate a good prediction. For example the model for the Import File block demonstrated a poor fit to the observed data (see figure 5). However, an examination of the spread of execution times shows a fairly small spread when compared with other data

intensive blocks. It is likely that, at the scales tested (a maximum data size of 14MB), the imported file sizes do not take a significant time to copy to the workflow engine meaning that there is an insufficiently rich set of data to build any meaningful predictive models.

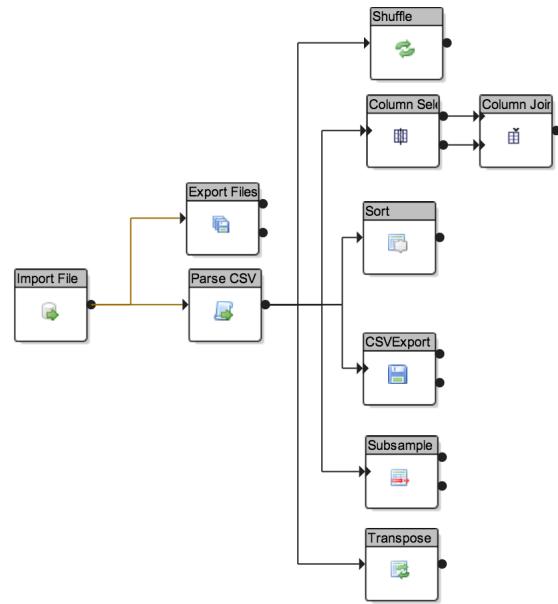


Fig. 3. Simple data generation workflow

This set of 250 workflow executions yielded 2500 observations within the performance database which were then used to build the suite of predictive models. In order to assess the performance of these models when applied to different workflows (containing a subset of the blocks shown in figure 3), a different workflow was constructed which again processed a set of randomly generated data (figure 6).

This workflow was executed multiple times and for each execution, the actual duration was recorded along with a duration estimate generated by the performance monitoring system ($RMSE=4.077, r^2=0.997$). The results of this exercise are shown in figure 7. Figure 7 shows a number of runs executing significantly faster than predicted. An examination of the results shows that these are the final executions of the experiment. Because the workflow engine executes multiple workflows concurrently, towards the end of any run, in the current setup, there is a strong chance that one workflow will be left executing alone. This workflow will not be competing for resources (filesystem, CPU etc) with other workflows and, as such, executes faster than predicted by the model.

2) Experiment 2: The second experiment carried out was a direct repeat of the first, but performed on a public cloud (Amazon EC2). The installation was configured in a manner typical of many e-Science Central installations and comprised a single server machine containing the JBoss application server and Postgres database with two additional machines, each containing a workflow engine. All machines used were m1.xlarge instances configured with 4x 2GHz Intel Xeon

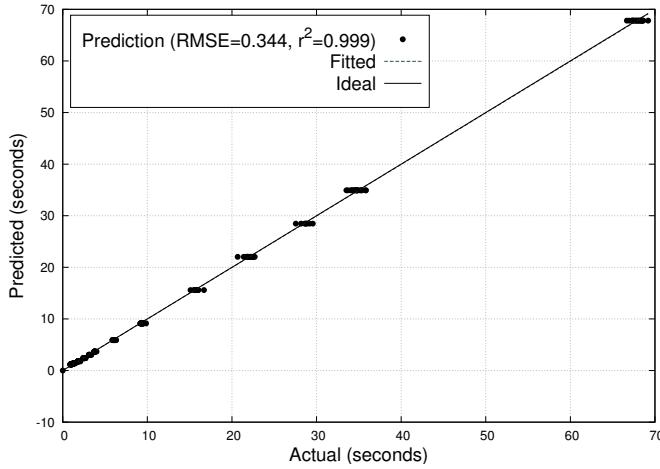


Fig. 4. Prediction of execution duration of the Shuffle block on local server

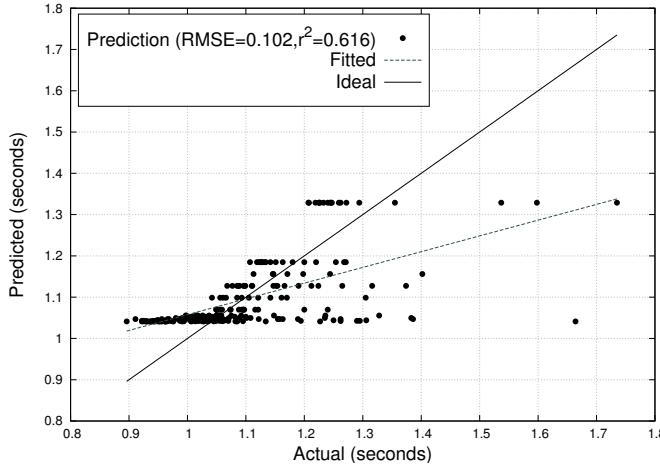


Fig. 5. Prediction of execution duration of the Import File block on local server

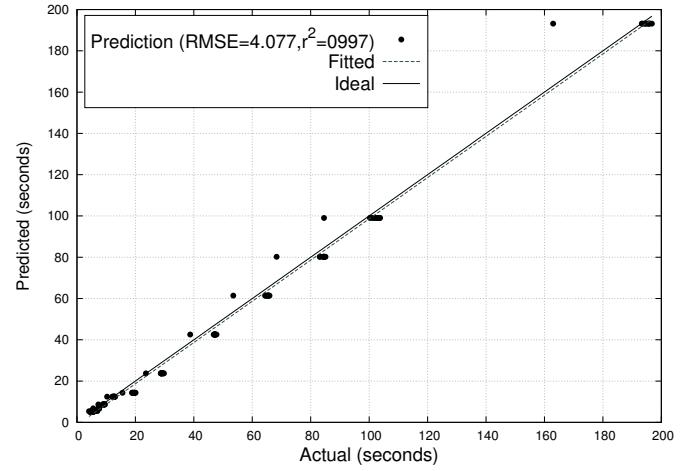


Fig. 7. Execution duration prediction for new workflow

CPUs and 15GB RAM. Once again, 250 executions of the simple calibration workflow shown in figure 3 were started and the results captured using the performance monitoring system. The results of this experiment also indicated a good model performance (see, for example figure 8, which illustrates the model performance on the Shuffle block), albeit with a slightly larger spread in predicted execution times when processing larger data files.

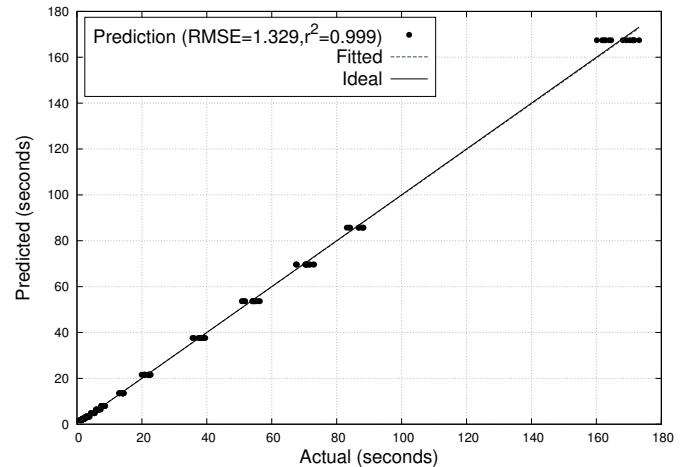


Fig. 8. Prediction of execution duration of the Shuffle block on Amazon EC2

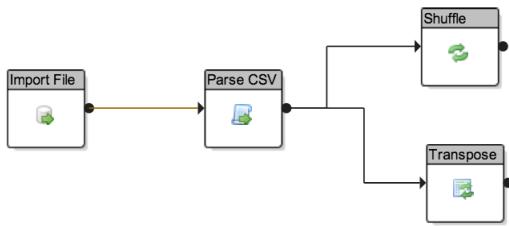


Fig. 6. Testing workflow

VI. MOVEMENT PREDICTIONS

Having determined the feasibility of generating models via the use of the ambient provenance and performance data capture architecture shown in Figure 1, the next experiment produced predictive models of a set of movement analytics routines operating within the e-Science Central platform. Although the platform has been deployed in multiple studies, the models built in this section focus on a standard library provided in R (GGIR) [6] that process raw accelerometry data collected from wearable devices. This library was selected

because it has seen the widest adoption of the various algorithms contained in the system and has the largest collection of performance data available. The algorithm itself comprises two operational phases:

- **Part 1:** Is a computationally expensive process that is performed once for each data file. Its purpose is to generate a set of metadata that can be used to facilitate a range of different analyses and produce various summary reports.
- **Part 2-n:** Once the summary metadata has been generated a set of second phase analyses can be carried out to generate reports.

The workflows modelled therefore follow a standard pattern: Data import and parsing, GGIR part 1 metadata generation then GGIR part 2 report generation. Data was collected from each of the two GGIR phases over a set of studies with the aim of generating duration and output size models.

Models for execution duration were generated using data gathered from the Optimistic (Observational Prolonged Trial In myotonic dystrophy type 1 to Improve Quality of Life Standards, a Target Identification Collaboration) study, which was operated by Newcastle University with data collection distributed between Newcastle, Munich, Nijmegen and Paris. This study focussed around approximately 300 people with a genetic diagnosis of myotonic dystrophy type 1, 18 years old and able to walk independently [24]. As part of this study, participants were asked to visit an assessment centre at least five times over a seventeen month period and were required to use a wrist worn accelerometer.

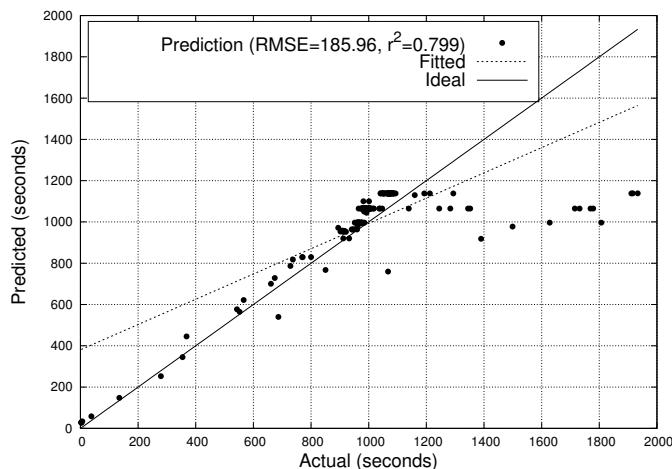


Fig. 9. GGIR Part 1 Duration Model

Data gathered from the three centres was uploaded to a central e-Science Central installation where workflows automatically performed a GGIR Part 1 processing step, followed by the generation of a customised report (a GGIR Part 2 operation). Interestingly, the GGIR Part 1 model demonstrated a roughly linear performance up until the execution time reached about 1200 seconds. Subsequently, the code took substantially longer than predicted to execute. After an investigation, it

became apparent that the data files were uploaded by to the e-SC server by the various study centres in batches. This had the effect of a number of files competing for resources on the system, with the effect that some executions took longer than predicted. The output size model (i.e. the size of the generated report) demonstrated a broadly linear response with the actual metadata size being in the range 7-10MB for each participant collection exercise. It is also apparent from the graphs that the majority of durations and output size predictions are clustered in a fairly tight area. The most likely cause of this observation is the fact that the study protocol produced very similar data file sizes as the participants all wore the devices for broadly similar periods of time.

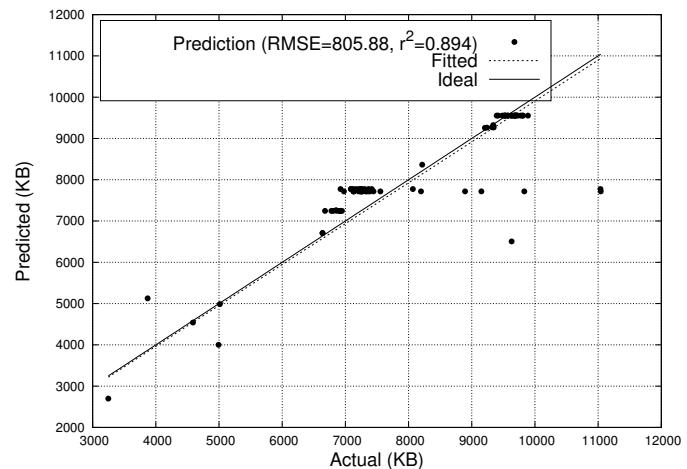


Fig. 10. GGIR Part 1 Generated Data Size Model

An updated version of the GGIR algorithm, which combines both Part 1 and Part 2 demonstrates this clustering behaviour more clearly, with an upper predicted execution time of approximately 1400 seconds. (Figure 11).

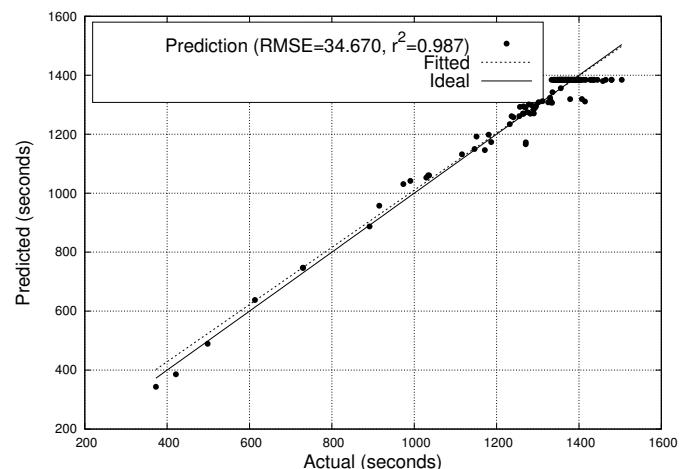


Fig. 11. GGIR Version 2 Duration Model

VII. CONCLUSIONS

This paper has demonstrated that, for the set of workflow components studied it is feasible to build reliable predictions of the execution time and volume of data produced from past executions and configuration parameters. These models have been shown to be robust in both controlled, local, environments and using shared virtualised environments provisioned in a commercial cloud provider. The components modelled include internal e-Science Central blocks and also open source algorithms used in the analysis of data in clinical trials. Using these models it is possible to predict the performance of large workflows, a capability which has been implemented in the e-Science Central Performance Server. Further work will include assessing the reliability of these predictions during subsequent studies which are scheduled to begin mid 2016.

Although we have only integrated the performance capture and modelling system with the e-Science Central platform, both the concepts and code should be applicable to any system that can generate performance logging messages in a similar format.

ACKNOWLEDGEMENTS

This work has been funded by the Digital Institute at Newcastle University, UK.

REFERENCES

- [1] S. Sabia, P. Cogranne, V. T. van Hees, J. A. Bell, A. Elbaz, M. Kivimaki, and A. Singh-Manoux, "Physical activity and adiposity markers at older ages: Accelerometer vs questionnaire data," *Journal of the American Medical Directors Association*, vol. 16, no. 5, pp. 438.e7 – 438.e13, 2015.
- [2] S. E. Crouter, J. R. Churilla, and D. R. Bassett, "Estimating energy expenditure using accelerometers," *European Journal of Applied Physiology*, vol. 98, no. 6, pp. 601–612, 2006. [Online]. Available: <http://dx.doi.org/10.1007/s00421-006-0307-5>
- [3] V. T. van Hees, S. Sabia, K. N. Anderson, S. J. Denton, J. Oliver, M. Catt, J. G. Abell, M. Kivimki, M. I. Trenell, and A. Singh-Manoux, "A novel, open access method to assess sleep duration using a wrist-worn accelerometer," *PLoS ONE*, vol. 10, 11 2015.
- [4] A. Godfrey, R. Conway, D. Meagher, and G. ÓLaighin, "Direct measurement of human movement by accelerometry," *Medical Engineering and Physics*, vol. 30, no. 10, pp. 1364 – 1386, 2008, special issue to commemorate the 30th anniversary of Medical Engineering and Physics30th Anniversary Issue.
- [5] S. Zhang, A. Rowlands, P. Murray, and T. Hurst, "Physical activity classification using the geneva wrist-worn accelerometer," *Medicine and Science in Sports and Exercise*, vol. 44, no. 4, pp. 742–748, April 2012.
- [6] V. T. van Hees, L. Gorzelniak, E. C. Dean Leon, M. Eder, M. Pias, S. Taherian, U. Ekelund, F. Renstrom, P. W. Franks, A. Horsch, and S. Brage, "Separating movement and gravity components in an acceleration signal and implications for the assessment of human daily physical activity," *PLoS ONE*, vol. 8, no. 4, p. e61691, 04 2013. [Online]. Available: <http://dx.doi.org/10.1371%2Fjournal.pone.0061691>
- [7] U. Dayal, M. Castellanos, A. Simitsis, and K. Wilkinson, "Data integration flows for business intelligence," in *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, ser. EDBT '09, 2009, pp. 1–11.
- [8] H. Hiden, S. Woodman, P. Watson, and J. Cala, "Developing cloud applications using the e-science central platform," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 371, no. 1983, 2013. [Online]. Available: <http://rsta.royalsocietypublishing.org/content/371/1983/20120085.abstract>
- [9] M. Dobber, R. van der Mei, and G. Koole, "Effective prediction of job processing times in a large-scale grid environment," in *High Performance Distributed Computing, 2006 15th IEEE International Symposium on*, 2006, pp. 359–360.
- [10] F. Nadeem and T. Fahringer, "Predicting the execution time of grid workflow applications through local learning," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, ser. SC '09. New York, NY, USA: ACM, 2009, pp. 33:1–33:12. [Online]. Available: <http://doi.acm.org/10.1145/1654059.1654093>
- [11] F. Nadeem and T. Fahringer, "Using templates to predict execution time of scientific workflow applications in the grid," in *Proceedings of the 2009 9th IEEE ACM International Symposium on Cluster Computing and the Grid*. IEEE Computer Society, 2009.
- [12] R. Duan, F. Nadeem, J. Wang, Y. Zhang, R. Prodan, and T. Fahringer, "A hybrid intelligent method for performance modeling and prediction of workflow activities in grids," in *Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on*, 2009, pp. 339–347.
- [13] X. Liu, Z. Ni, D. Yuan, Y. Jiang, Z. Wu, J. Chen, and Y. Yang, "A novel statistical time-series pattern based interval forecasting strategy for activity durations in workflow systems," *Journal of Systems and Software*, vol. 84, no. 3, pp. 354 – 376, 2011. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0164121210003262>
- [14] P. Watson, H. G. Hiden, S. J. Woodman, D. E. Leahy, J. Cala, and P. Missier, "The panel of experts cloud pattern," in *Proceedings of the third international workshop on Cloud data management*, ser. CloudDB '11. New York, NY, USA: ACM, 2011, pp. 23–24. [Online]. Available: <http://doi.acm.org/10.1145/2064085.2064091>
- [15] R. Cushing, S. Koulouzis, A. S. Z. Belloum, and M. Bubak, "Prediction-based auto-scaling of scientific workflows," in *Proceedings of the 9th International Workshop on Middleware for Grids, Clouds and e-Science*, ser. MGC '11. New York, NY, USA: ACM, 2011, pp. 1:1–1:6. [Online]. Available: <http://doi.acm.org/10.1145/2089002.2089003>
- [16] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, 2011, pp. 500–507.
- [17] T. Miu and P. Missier, "Predicting the Execution Time of Workflow Activities Based on Their Input Features," in *Procs. WORKS 2012*, I. Taylor and J. Montagnat, Eds. Salt Lake City, US: ACM, 2012.
- [18] Q. Wu and V. V. Datla, "On performance modeling and prediction in support of scientific workflow optimization," in *Proceedings of the 2011 IEEE World Congress on Services*, ser. SERVICES '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 161–168. [Online]. Available: <http://dx.doi.org/10.1109/SERVICES.2011.37>
- [19] V. Taylor, X. Wu, and R. Stevens, "Prophesy: an infrastructure for performance analysis and modeling of parallel and grid applications," *SIGMETRICS Perform. Eval. Rev.*, vol. 30, no. 4, pp. 13–18, Mar. 2003. [Online]. Available: <http://doi.acm.org/10.1145/773056.773060>
- [20] V. Taylor, X. Wu, J. Geisler, and R. Stevens, "Using kernel couplings to predict parallel application performance," in *High Performance Distributed Computing, 2002. HPDC-11 2002. Proceedings. 11th IEEE International Symposium on*, 2002, pp. 125–134.
- [21] S. Sadjadi, S. Shimizu, J. Figueira, R. Rangaswami, J. Delgado, H. Duran, and X. Collazo-Mojica, "A modeling approach for estimating execution time of long-running scientific applications," in *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, 2008, pp. 1–8.
- [22] J. Cala, H. Hiden, S. Woodman, and P. Watson, "Cloud computing for fast prediction of chemical activity," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1860 – 1869, 2013. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0167739X13000253>
- [23] S. Woodman, H. Hiden, P. Watson, and P. Missier, "Achieving reproducibility by combining provenance with service and workflow versioning," in *Proceedings of the 6th workshop on Workflows in support of large-scale science*, ser. WORKS '11. New York, NY, USA: ACM, 2011, pp. 127–136. [Online]. Available: <http://doi.acm.org/10.1145/2110497.2110512>
- [24] B. van Engelen, "Cognitive behaviour therapy plus aerobic exercise training to increase activity in patients with myotonic dystrophy type 1 (dm1) compared to usual care (optimistic): study protocol for randomised controlled trial," *Trials*, vol. 16, no. 1, pp. 1–19, 2015. [Online]. Available: <http://dx.doi.org/10.1186/s13063-015-0737-7>

Accelerating Data-Driven Discovery With Scientific Asset Management

Robert E. Schuler, Carl Kesselman, and Karl Czajkowski

{schuler, carl, karlcz}@isi.edu

Information Sciences Institute

University of Southern California

Marina del Rey, CA 90292, USA

Abstract—The overhead and burden of managing data in complex discovery processes involving experimental protocols with numerous data-producing and computational steps has become the gating factor that determines the pace of discovery. The lack of comprehensive systems to capture, manage, organize and retrieve data throughout the discovery life cycle leads to significant overheads on scientists' time and effort, reduced productivity, lack of reproducibility, and an absence of data sharing.

In “creative fields” like digital photography and music, digital asset management (DAM) systems for capturing, managing, curating and consuming digital assets like photos and audio recordings, have fundamentally transformed how these data are used. While asset management has not taken hold in eScience applications, we believe that transformation similar to that observed in the creative space could be achieved in scientific domains if appropriate ecosystems of asset management tools existed to capture, manage, and curate data throughout the scientific discovery process. In this paper, we introduce DERIVA, a framework and infrastructure for asset management in eScience and present initial results from its usage in active research use cases.

I. INTRODUCTION

Scientific discovery is undergoing a profound transformation driven by high-throughput instruments, pervasive sensor networks, large-scale computational analyses, growing dependence on collaborations and virtual organizations, and a changing scientific communications ecosystem that places increased emphasis on data sharing. Discovery driven by computational analysis and exploration of data has been recognized as the fourth paradigm of scientific discovery alongside empirical and theoretical methods, leading us to a new era of *Discovery Science*, a methodology of developing and testing hypotheses from large, rich, and complex data where the initial observation may precede the hypothesis [1].

Traditionally, a knowledge cycle in scientific discovery has been regarded as the formation of new knowledge through repeated turns of hypothesis, prediction, observation, and analysis, which are punctuated by transmission of results in the form of publications [2]. In the new paradigm of discovery science, these *knowledge turns* are increasingly

dependent on a scientist's ability to acquire, curate, integrate, analyze, and share data well beyond the compact reports offered by traditional publication. While the details of these cycles vary from domain to domain, and indeed across time within a single discovery process, data driven discovery cycles share common characteristics and face similar data related problems regardless of the domain. Experimental protocols involve multiple steps with data capture from diverse, high throughput, high fidelity instruments, analytic pipelines or computational simulation models. The results of one cycle of experiments are iteratively fed back into the system to refine the next series of experiments. Data must be contextualized within the protocol steps they are captured and may be related to physical specimens and other details of the experimental stage that produced the data. Data will also be produced by computational and human analyses in other steps of the protocol and also must be contextualized and linked to related data.

As the complexity and volume of the data that are used to drive knowledge turns in eScience applications increase, the ability of the scientist to manage the logistics of executing these cycles can become a rate limiting step. Current approaches to scientific data management have failed to keep pace with the needs of increasingly data-intensive science. Managing data is often done manually with “meaningful” file names and directory hierarchies, locally coded spreadsheets, and ad hoc laboratory notebooks. As noted by [3], “large amounts of data are generated using a variety of innovative technologies and the limiting step is accessing, searching and integrating this data.” Scientists and other data analysts report spending 50% or more of their time on data wrangling (locating, extracting, cleaning, formatting, organizing) tasks [4], which leads to potential misinterpretation of results, misuse of data, forgetting pertinent details to describe the data, inability to find and retrieve previously captured data and results, and other similar issues. Concerns over repeatability of scientific results are growing along with an increase in scientific retractions [5], and some reports indicate that there is as little as 10% reproducibility of scientific results which cite lack of data publication as one of the significant factors [6]. As noted in [7], tools merely to support data capture are “just dreadful” and “we lack good tools for both data curation

The work presented in this paper was funded by the National Institutes of Health under awards 5U54EB020406, 1R01MH107238-01, 5U01DE024449 and 1U01DK107350

and data analysis.” These sentiments echo similar observations made decades earlier by J. C. R. Licklider in his seminal work on the man-computer symbiosis, as he states, “...my choices of what to attempt and what not to attempt were determined to an embarrassingly great extent by considerations of clerical feasibility, not intellectual capability [8].” Decades later, these same issues continue to obstruct data-driven discovery.

While scientific data management has lagged behind the needs of data-driven discovery, this is not the case in other creative activities that also depend on managing large, complex data sets. For many years, the prevailing method of collecting digital pictures was to categorize images following user-defined, ad hoc naming conventions, resulting in a rigid, hierarchical, and often-confusing array of files and directories, much like scientific data is managed currently. Today, photographers use em digital asset management (DAM) systems such as Apple Photos or Google Photos to automatically discover and catalog digital images on their cameras or hard disk drives; extract metadata from the imported media; add user annotations; organize pictures into virtual collections (i.e., photo albums); browse and search on metadata, annotations or features such as faces in the picture; export data for manipulation by external photo editing tools; and publish to cloud-based sharing or printing services. By comparison, scientific data management has failed to address the data wrangling tasks incumbent on scientists today.

A DAMS ecosystem that streamlines the acquisition, management and sharing of digital assets such as pictures and music, has had a transformative effect on how we think about and use these data. We assert that creating a digital asset management ecosystem for eScience data could have a similar impact: to transform the way that scientists interact with data so as to radically accelerate knowledge turns for scientific discovery.

This paper makes the following contributions:

- 1) We introduce the idea of asset management as a way of addressing eScience data management challenges.
- 2) We present the requirements for asset management systems based on analysis of eScience applications.
- 3) We propose a general architecture for scientific asset management ecosystems.
- 4) Finally, we describe the first platform implementation for scientific asset management.

In the following sections, we will provide desirable characteristics and requirements for an eScience DAMS. We will then describe our proposed architecture for scientific asset management and the implementation of DERIVA, a platform for introducing DAMS functions into eScience applications. We then share a preliminary evaluation of the approach and its framework in the context of science applications that use it regularly. Finally, we present plans for future work and then offer our conclusions.

II. CHARACTERISTICS AND REQUIREMENTS FOR SCIENTIFIC ASSET MANAGEMENT

The specifics of a data driven discovery process will vary from domain to domain, and even from instance to instance. However, it is also the case that there are common needs and requirements that cut across all types of data driven discovery. By considering these needs across a wide number of different use cases, we have propose that an eScience DAMS ecosystem should provide the following capabilities:

- Acquisition and characterization of diverse scientific assets, including experimental data from instruments (e.g. microscopes, sequencers, flow cytometers), outputs from computational models, and results from analysis pipelines. Data must flow freely and automatically from the points of production into the management system, much like pictures flow from a smartphone into a management application.
- Model-driven organization and discovery of assets. Successful DAMS systems in the consumer space provide end users with intuitive and interactive ways of organizing and discovering assets that may be related via a complex underlying model. For example, in music DAMS systems, one may discover based on artist, group, work, composer, instrumentation, genre, etc. Similarly, an eScience DAMS should provide model based organization and discovery, in spite of the fact that the models may vary radically from domain to domain, may cross domains (multi-disciplinary collaborations) and vary over time as the discovery process unfolds.
- Storage and retrieval of eScience data assets. These assets may be very large, and may be physically distributed in local, enterprise, and cloud based storage systems.
- Aggregation and exchange of data collections. An eScience DAMS should be viewed as the hub of a data management ecosystem and not create unnecessary data silos. Hence, it is critical that the DAMS allow users to assemble and export data sets for consumption by other tools and users.
- Rights management/access control. A core function of DAMS is management of IP associated with assets. In the eScience environment, this means enforcing data use agreements, access to proprietary data, time driven data embargoes, and different user roles within and across collaborations.

A. Acquisition and Characterization of Scientific Assets

We represent data driven discovery in terms of an evolving set of *scientific “digital assets”* (i.e., research data or simply assets), which are described and related to one another via a *domain model*. The idea of an asset contextualized within a domain model is similar to semantic models for describing argument and evidence as “micropublications” [9] where the continuous acquisition and characterization of assets throughout the discovery process may be viewed as incremental micropublications with each asset as an embodiment of evidence on which scientific arguments may be grounded.

Diverse sources of assets. An asset may be generated by sensors, instruments, or as the result of a computation. Like photos on a smartphone, assets should be seamlessly integrated into the asset management system. The production of assets is itself fluid and cyclic throughout the discovery process as new data are generated, analyzed, shared with collaborators, and exported for publication. As new assets are acquired, they may be immediately consumed by other actors such as collaborators or computational workflows and analysis pipelines.

Diverse forms of assets. Scientific assets come in different forms, formats, and conceptually at different levels of granularity. For example, a video recording can also be represented more granularity as a time series of individual frames. In some cases, therefore, it may be useful to reference the parts versus the whole and conceptually to model the video as an aggregation of frames.

Incremental derivation of assets. The framework must allow incremental derivation of assets throughout the scientific discovery process. Data may be captured early in the discovery process. However, at the point of acquisition researchers may only have minimal contextual information to describe the asset, such as the date and time of capture, the instrument type and settings, and identity of the investigator. It is not until later that more information regarding the asset will be known, such as measures of data quality. The acquired data may also become the input for downstream computational analyses, which will generate new insights and may result in derived data of its own. Data does not enter the discovery process fully formed but often goes from a minimally- to maximally-described state [9].

Automation and self-service curation. Manual data entry has been noted as one of the key barriers to successful adoption of data management services [10]. The framework must enable self-service curation of assets with simple user interfaces and automated services to reduce manual effort where possible. However, the diversity of scientific domains and data and their unique requirements means that simple one-size fits all asset management applications will not suffice. At the same time, developing new interfaces and applications for each use case is prohibitively expensive and time consuming. The user applications must adapt to the underlying domain model. They cannot assume a rigid structure but must be flexible both to the structure of the data and also the workflow of researchers using the system.

B. Models and Evolution for Scientific Asset Management

Domain models represent the key concepts, behaviors and relationships of the participants and elements in a real-world system. Domain models are used to describe scientific assets in order to link, organize and contextualize them within the discovery process. Effectively modeling the information for such diverse and complex domains of science motivates the need for domain modeling approaches that support well-defined, structured information. Yet, the process of scientific discovery is always unfolding and yielding new insights and

understanding of the domain and hence systems to support it must be able to evolve as well.

Complex data models. We argue that when one considers the complexity of scientific information and the importance of precise descriptions of research and results that a *structured* data model is necessary, ideally one based on a strong formalism such as relational or graph theory. For example, the *entity-relationship model* (ERM) can serve as the underlying meta-model for domain models in scientific asset management. ERMs are extremely expressive and can be used to describe data in tabular as well as graph structures through commonly used idioms. Another important factor in considering ERMs as an underlying meta-model is that the majority of scientific data is in practice represented in tabular structure [11] and therefore is a natural fit for use in scientific data modeling.

While it has become increasingly popular to abandon structured data models like ERMs in favor of so-called *NoSQL*, *schema-less*, *key-value*, or their decades old predecessor *entity-attribute value* (EAV) databases in order to avoid data modeling, there is nothing inherently static or restrictive about ERMs; in fact, most modern database management systems support schema manipulation. Scientific query workloads have been shown to depend on complex query operations not possible with simpler query dialects and less structured models [12]. More recently, traditional database systems have added support for features that were thought of as the domain of “document-oriented” databases, such as special handling of JSON and text processing, thus offering the features of relaxed schema with the advantage of having full capabilities for structured data management.

Evolution and introspection of models. Upfront data modeling is a significant bottleneck to the adoption and usage of complex data models. For science applications, it may be impossible to define a data model upfront and the model may change in unexpected ways throughout the discovery process. It is therefore essential that the ecosystem for managing scientific assets be based around generic tooling that *introspects* domain models and adapts to them. The systems must allow scientists to begin with minimally-specified models and evolve the model and data throughout the discovery life cycle, while continually refining and increasing the richness and rigor of the structures used to describe their research. Likewise, the ecosystem of tools and services for asset management must be sensitive to the changes in the model. Since data producers and consumers operate independently and asynchronously, they must adapt dynamically. Even in the middle of an interaction with the data, the model may change based on the interactions of another agent in the system. Tools should be model-agnostic so that they may be useful across diverse scientific applications and to allow model evolution without having to upgrade software repeatedly for every model or data change.

One key obstacle for supporting flexible yet structured data models, is that many applications developed in the popular object-oriented programming (OOP) paradigm are often developed using *object-relational mapping* (ORM) libraries that define their own proprietary query dialects and suffer

from the well-known *object-relational impedance mismatch* problem i.e., objects and classes map poorly to relations. Unlike ORMs, we take an approach that does not hide or obfuscate the underlying model from the client. Instead, *the elements of a domain model are represented and exposed directly and transparently through protocols and interfaces* appropriate to the technology platform in which asset management is implemented. This argument is similar to the “don’t repeat yourself” or “DRY” principle – that information should have a single unambiguous representation. By contrast, ORMs and popular Web frameworks introduce additional, obfuscated layers that must be updated whenever the data model changes.

Extending and enriching models with annotations. While a formal meta-model, like the ERM, is necessary to model scientific domains, it may be insufficient to describe the complete semantics of the model. In a relational model, one cannot describe anything more about a particular element in the model other than that it is a ‘table’ or a ‘column’ or a ‘constraint.’ Additional semantics on the model are needed as a way of extending and enriching the basic concepts of the meta-model. For this purpose, *model annotations* are a way of filling the gap beyond what the meta-model can provide. The annotations are applied to various levels of the domain model including the individual schemas, tables, columns, and foreign key references. The annotations can be used to indicate additional semantics about an element of the model, provide presentation hints to the user interface for how to render a model element or its data in an interface, and other uses applicable to the domain or applications and agents involved in mediating and manipulating the schema. Examples of annotations are shown in Table I.

TABLE I
EXAMPLES OF ANNOTATIONS USED TO DESCRIBE MODELS.

Annotation	Description
tag:misd.isi.edu,2015:default	Schema or table to be selected by default by user interface presentation.
tag:misd.isi.edu,2015:url	Table or column should be rendered as an link.
tag:misd.isi.edu,2015:vocabulary	Table should be interpreted as a controlled vocabulary.

Heuristics for understanding models. While domain models will be extremely diverse especially from one scientific domain to the next, we have found that a small number of *heuristics* can be very powerful in enabling presentation and interaction without statically coding behavior into the applications and tools that operate over structured data models. Our heuristic approach makes some simple assumptions about the semantics of the model, particularly as it concerns rendering, navigating, and updating data in a complex domain model. For example, when rendering a detailed view of an entity (i.e., a row) from a table, we can apply heuristics that denormalize the rendering of the entity in order to contextualize with its “neighbors” in the linked data graph of relationships it forms with references to and from other entities in the ERM. This

heuristic strategy can be applied generally across different models, domains, and with alternative meta-models as an approach to developing general tooling for working with asset management systems. Examples of heuristics are shown in Table II.

TABLE II
EXAMPLES OF HEURISTICS USED TO INTERPRET MODELS.

Heuristic	Description
Ignore auto-generated	Disable input for system-generated keys.
Extended record	When displaying a record, extend the record with entities joined by many-to-one relationships.
Nested entity	When displaying a record, show preview of entities joined by one-to-many relationships.

C. Storing and Accessing Scientific Assets

Some of the closest approximates to asset management storage systems (a.k.a., data stores) come in the form of version control systems and object stores that operate on data as atomic units and permit changes only through explicit versioning semantics. In general, conventional storage systems do not necessarily suffice for the needs of scientific asset management. Here we discuss the requirements unique to storing and accessing scientific assets.

Stable naming of assets. Assets and their metadata are referenced by name, and data names can be shared between actors by external means or embedded within other data, which may exist in the same or different data stores. Therefore, references to data are not necessarily under the control of the data store holding the referenced data. Rather, references may be asynchronously communicated between distributed parties. Within the context of asset management, a data store must deliver certain guarantees: a reference to an asset should never become ambiguous, but rather should always denote one particular asset whether or not the data is currently available for retrieval; a retrieval operation should be unambiguous, with a consumer being able to determine whether they have successfully retrieved the denoted asset or have encountered an access error. In short, the retrieval of named assets should be atomic, consistent and stable.

Immutability of assets. Scientific processes depend on reliable acquisition and sharing of data in order to support reproducible results, thus once an asset has been acquired it must not be mutated. Consumers such as manual reviewers of imaging data or computational workflows that take complex data as input must be assured that the assets they consume are exactly the assets that were produced and shared by the data producing actors. A close corollary to stable names for assets (discussed above) is that assets must therefore be immutable, such that once generated no edits or other in-place changes may be allowed. It is more acceptable, though perhaps not ideal, for an asset to be *retracted* from the system just as publications may be retracted from the scientific literature. In these exceptional cases, a marker sometimes called a “tombstone”

should be left in the asset's place so that references to the asset may be resolved at least and consumers can understand why the asset is no longer available. In data preservation terms, immutability is often called data or file *fixity* meaning the contents of a file are fixed and cannot be changed.

Versioning of assets. At times, a new asset will be generated based on incremental changes starting from an existing asset and in such cases, these assets enter the system as new *versions* of an earlier asset. Versioning, however, must be done explicitly as a new asset with an explicit version relationship to another existing asset and as stated earlier must not mutate an asset in place. To assist in this a data store of assets should issue a name for an asset only once and may issue *version-qualified names* to add convenience to the consumers and references of assets.

Linking and provenance of assets. Versioning and other operations that derive one asset from another further motivate the need for facilitating models of provenance. We see these as one natural application of the general requirement for linked data, where the relationship between data is semantically described in terms of the processes and actors that produced an asset and existing models of provenance [13] should be employed for this purpose.

D. Aggregation and Exchange of Assets

Methods must exist to extract collections of assets under management for consumption by external human or computational agents. Several alternatives exist for exchanging complex information via various forms of *data aggregation packages*, for example Research Objects [14] or structured file formats such as HDF. These methods can be used to facilitate exchange of asset sets from a DAMS.

E. Policies for Managing Assets

Numerous forms of policy must be considered when managing scientific assets.

“Protected” data. Research involving sensitive data collected during studies involving human subjects are governed by policies administered by Institutional Review Boards (IRBs) or other bodies that set Data Use Agreements (DUAs). These data must be handled in compliance with rules governing privacy and strict data sharing limits. Under these circumstances, even when research concludes the IRB or DUA policies may stipulate how storage systems must be purged of stored information.

Private use of data. In general, data are of great value to the researchers that produce them. Data sharing embargoes specify limitations to access of data while the investigators, those that generated the data, use them in analysis and until they have been published. The ecosystem of tools for asset management must be cognizant of the policies, whether governed by formal policies or defined by individual researchers, and the tools must limit use and sharing of data in compliance with the specified policies.

III. SCIENTIFIC ASSET MANAGEMENT ARCHITECTURE

Based on the requirements and characteristics of asset management from our analysis in the previous section, we have defined the scientific asset management architecture (Fig. 1). Our proposed model extends from the standard distributed systems architecture of the Web according to the unique requirements of scientific asset management to enable an ecosystem of tools and services for discovery science. Our goal is not to define an exhaustive collection of providers for various capabilities but rather to identify general classes of services and tools that are necessary for implementing asset management solutions in various configurations and for diverse scientific applications. We now describe the primary components of our scientific DAM architecture.

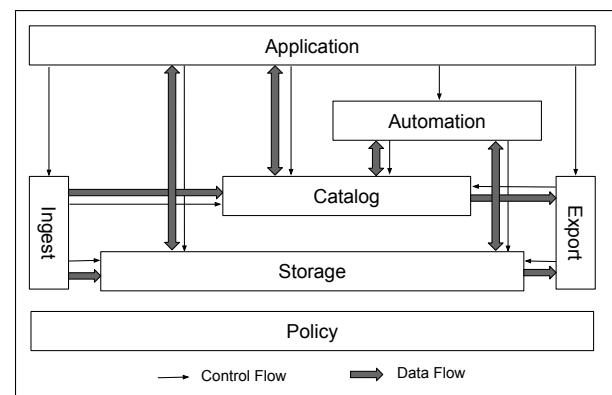


Fig. 1. Scientific Asset Management Architecture.

Application. Applications provide adaptive user interfaces for acquiring, curating, and consuming assets. They are model-agnostic and function by introspecting the data stores, in particular the structured data catalogs that hold the domain models and descriptive information. These applications support searching and browsing paradigms to help users locate assets of interest. They let users define custom subsets and slices of data and support entry and editing of new data and annotating of assets. Applications must be model-driven and reflect the current state of the catalogs and data stores without hard-coded assumptions and semantics about domain models.

Catalog. The Catalog layer represents specialized forms of the data store for recording, querying, and retrieving descriptive information (i.e., metadata) about scientific assets. As specified in the requirements, in order for the catalog to sufficiently handle domain models for scientific applications, it must support structured data models, model evolution, interfaces so that clients can introspect domain models, complex query operations and named queries, and model annotations. The catalog is primarily a passive service that applications and other tools (like ingest and export pipelines) interact with to store, query, and retrieve metadata about assets in the system.

Storage. The Storage layer represents the data store for bulk asset storage and retrieval. It must satisfy requirements for transactional update, permanent naming of assets, and non-destructive updates. It must ensure the immutability or fixity

constraints and may support versioning. Like the catalog, the data store is primarily a passive service that provides interfaces for applications and other utilities (like ingest and export pipelines) to store and retrieve scientific assets.

Ingest. The role of ingest utilities and services is to identify new assets of interest, extract and harvest metadata from them, record descriptive information so that assets may be contextualized with who, what, when and how an asset was generated, which may only be possible to do at acquisition. It must interact with catalog and storage services to record and store assets. The ingest layer may be configured offline or may be administered through applications.

Export. Export services or utilities, on the other hand, extract assets and metadata from catalog and storage services, serialize and package them in well-defined package formats, and may deposit them on recipient systems. As with the ingest layer, the services and utilities of the export layer may take an active (e.g., by polling) or passive (e.g., responding to events or other signals) role in identifying what and when to export assets. They may also be configured offline or administered online through applications.

Automation. A wide range of data management and manipulation services may be automated in an asset management system. The services and utilities of the automation layer represent a potentially broad array of capabilities that will be needed in scientific asset management. Automation services will primarily take an active role interacting with catalog and storage layer services to perform a variety of domain- and task-dependent operations, including but not limited to metadata extraction and harvesting, format conversion, image stacking, image tiling, down-sampling, applying compression to various content types, or indexing data.

Policy. The Policy layer represents a broad class of services, utilities, and other tools to help specify, evaluate, enforce, and otherwise make policy decisions. These tools tend to play a central and ubiquitous role in all manner of distributed systems. In asset management, the components of the policy layer may be used to express and evaluate policy rules for determining individual and role-based access to assets and metadata, for example. The proposed architecture does not specify or require that policy be implemented centrally or decentralized by the individual components of the different layers of the architecture which may make independent and local policy decisions in practice.

IV. DERIVA: A PLATFORM FOR SCIENTIFIC ASSET MANAGEMENT

Based on the above analysis, we have created a DAMS platform for eScience applications called the **Discovery Environment for Relational Information and Versioned Assets** (DERIVA). The implementation of the DERIVA platform consists of a multi-tenant relational data service for domain models (ERMREST), a client library (ERMRESTJS), an object storage service for assets (HATRAC), a suite of adaptive user interface applications (CHAISE), a suite of utilities for ingest

and export of assets and metadata (IObox), an asset aggregation package format (BDBag), and a shared authentication layer (WebAuthN).

A. CHAISE

CHAISE is the user interface application suite for DERIVA. CHAISE is implemented as JavaScript programs that dynamically generate adaptive, model-driven user interface applications for discovery, analysis, visualization, data entry, annotation, sharing and collaboration over scientific assets. CHAISE applications introspect and dynamically render relational data resources based on a small set of baseline assumptions, combined with its rendering heuristics, which may be influenced, informed or overridden by model annotations defined on the domain models that are catalogued in ERMREST, and finally user preferences further refine the interfaces.

CHAISE is intended to support specific asset management interactions. As such, its presentation capabilities are narrowly scoped. CHAISE makes a few assumptions about how users will interact with the underlying data. A few representative but non-exhaustive examples of these assumptions include:

- Search, explore, and browse catalogs of assets;
- Linked data navigation between records;
- Add, edit, remove records from the catalog;
- Create, alter, or extend the domain model in the catalog;
- Subset and export collections of assets and metadata;
- Share collections with others;
- Annotate records with tags or controlled vocabulary terms.

CHAISE is structured as a set of programs that generate interfaces to search and browse assets using the powerful ‘faceted search’ paradigm (see fig. 3); deep drill-down, visualization, and ‘linked data’ navigation on individual records (see Fig. 4); and record entry and edit. Internally, the CHAISE applications are built on AngularJS¹, a front-end model-view-controller (MVC) framework developed by Google, and a common library of routines used across CHAISE applications (see fig. 2). CHAISE uses the ERMRESTJS library, which provides JavaScript language bindings for the ERMREST protocol and a comprehensive set of APIs for working with the ERMREST relational data service.

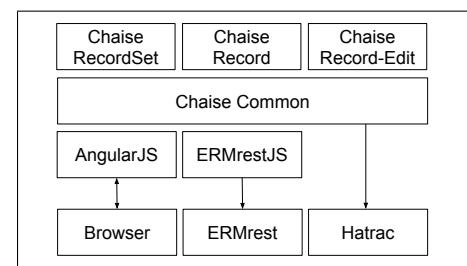


Fig. 2. Chaise layered architecture.

CHAISE makes no assumptions about the structure of the underlying data model, such as its tables, columns, keys, and

¹<https://angularjs.org>

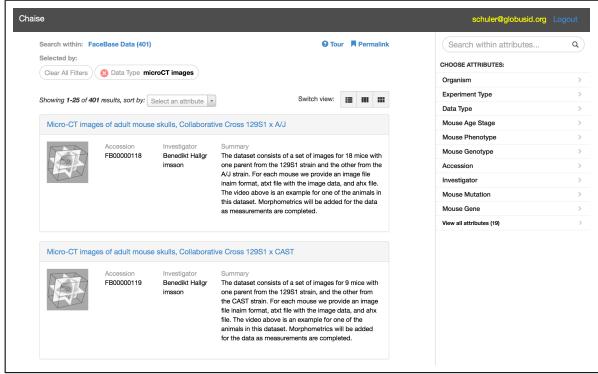


Fig. 3. Faceted search application.

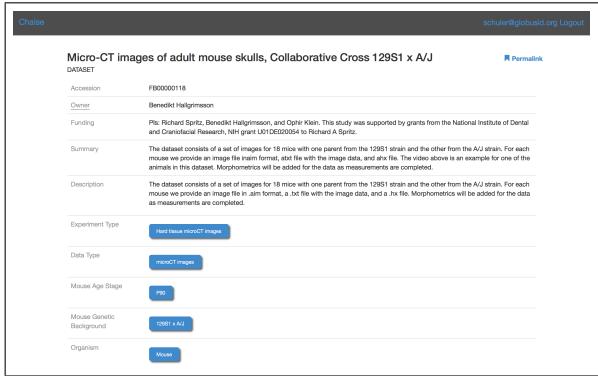


Fig. 4. Record details application.

foreign key references. It begins by introspecting the data model by getting the schema resource from ERMREST. It then uses rendering heuristics to decide, for instance, how to flatten a hierarchical structure into a simplified (or “denormalized”) presentation for searching and viewing. CHAISE then interprets the model annotations to modify or override its rendering heuristics, for instance, to hide a column of a table or to present a related entity as an embedded Web resource in an inline frame (`iframe`). CHAISE uses model annotations to determine when and how to integrate visualization tools into the display, including D3, Plotly, 3D volume rendering, and 2D, pyramidal, tiled image rendering.

B. ERMREST

ERMREST (Entity Relationship Model via Representational State Transfer) is a relational data service for the Web, and allows general entity-relationship modeling and manipulation of data resources by RESTful access methods. ERMREST serves as the metadata catalog of DERIVA and enables the evolving and dynamic data models needed for describing, contextualizing, and linking scientific assets.

The design goals for ERMREST were to enable non-expert users to create and evolve data models that represent the semantic concepts in their domain without the typical round trips from user to developer to database administrator and back. Many use cases can map into simple models with just a handful of entities and relationships and non-experts can

easily think about their domain in terms of the main concepts that they want to manipulate [11]. By providing methods for incrementally creating these models, and by allowing users to express domain concepts directly in the catalog, it offers a platform where the data models can be created and maintained by the user community. While there will be situations in which a more formal up-front data modeling activity will be required and for which the full power of SQL may be needed, a significant number of important usage scenarios fall within the design space of ERMREST.

Our approach was to develop a service that supports resource manipulation idioms common to RESTful Web services. ERMREST maps *Entity*, *Attribute*, *Schema*, *Table*, *Column*, and other relational concepts to Web resources, which are referenced by Web resource identifiers (i.e., URIs). It supports the following interfaces:

- ***catalog***: reference, retrieve, create, alter, delete “catalog” resources, each of which is an independent relational data store;
- ***schema***: reference, introspect, and alter entity-relationship models (i.e., schema name spaces, table and column definitions, key and foreign key constraints, etc.);
- ***entity***: reference, query, and manipulate entity records (i.e., rows of a table);
- ***attribute***: reference, query, and manipulate projected attribute records (i.e., subsets of attributes of relations);
- ***attributegroup***: reference and query projected attribute group records (i.e., attributes grouped by a subset of attributes of relations);
- ***aggregate***: reference and query projected aggregates with supported *aggregate functions* such as count, min and max.

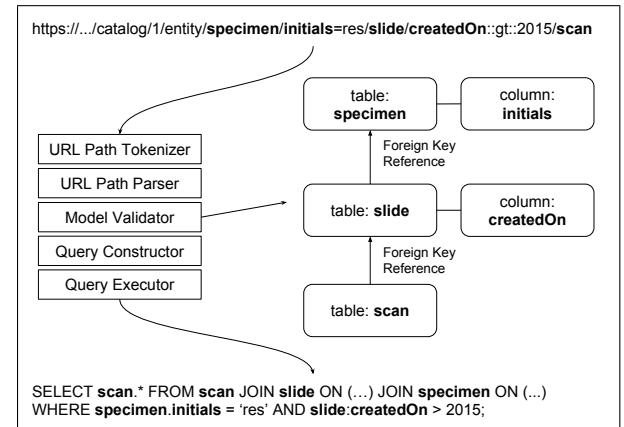


Fig. 5. ERMREST query processing example, showing the query URI (top), model definition (middle right), conceptual processing stages (middle left), and generated SQL statement (bottom).

The core of the implementation is in query processing. A request consists of the HTTP method (e.g., GET, POST, PUT, DELETE), URL path (e.g., `catalog/1/entity/scan`), and optional HTTP message body (e.g., JSON or CSV resource representation). The query language is a formally-defined context-free grammar in BNF notation and is parsed

using a generated “look ahead left-to-right” (LALR) parser. Fig. 5 depicts the query processing steps taken to satisfy each request to ERMREST.

First, the request handler tokenizes and parses the URL path into an Abstract Syntax Tree (AST) representation of the query. The catalog resource (e.g., `catalog/1/`) indicates which relational data store to address with the query. Next, the interface part of the path (e.g., `entity/`) indicates which API will ultimately be used to answer the query. The remainder of the URL is what we call the *data path*, a pseudo-hierarchical expression that references, joins, and filters tables of entities and attributes. In the example, the first table in the path is the `specimen` table, and it is filtered by the binary predicate `initials=res`, which references the `specimen.initials` column in the left operand. The next URI component, which is delimited as usual by the ‘/’ character, implicitly specifies a join with the `slide` table resource. The model validator will ensure that an unambiguous foreign key reference exists, in either direction, between the `specimen` and `slide` tables. Next, the filtered product of joined tables will again be filtered, this time by the binary predicate `createdOn::gt::2015` which filters slides that were created on or after year 2015. Finally, the last URI component again specifies an implicit join, in this case with the `scan` table.

The `entity` interface returns whole entities (i.e., rows of a table resource) and therefore does not require an explicit projection of attributes. With the `attribute` interface, however, URL *data paths* terminate with a list of attributes to be returned. More complex *data path* expressions are supported including projections involving multiple tables in the expression, joins between multiple tables at different depths of the expression hierarchy, table alias assignments and referencing, aggregation and grouping.

Finally, ERMREST recently incorporated row-level access control policies from its underlying PostgreSQL database engine. Combined with user- and role-based authentication from WebAuthN (discussed later), ERMREST can enforce fine-grain access control over the contents of the catalog. For example, permissions may be granted on tables based on group membership; visibility or editing may be enforced on a row-by-row basis, and more sophisticated policies are also possible.

C. HATRAC

HATRAC (pronounced “hat rack”) is a Web service for storage and retrieval of scientific assets as Web resources in a RESTful service model. HATRAC treats scientific assets as generic, opaque, byte-sequences and may be viewed as an *object store* for asset management. It supports *atomic operation semantics* – that is, an asset is created and named, updated or deleted in an atomic operation where the operation either finishes and succeeds completely as expected or terminates the operation. Once an asset has been stored in HATRAC, it guarantees *data fixity*, first by enforcing immutability of stored objects, and second by maintaining check sum message digests which can be used to ensure data integrity. An “update” of an

asset in HATRAC is non-destructive, such that the service preserves the current state of the asset while adding a new version of the asset in a *version-qualified naming scheme*. Similarly when “deleting” an asset, the service marks the named asset as deleted but does not release the name for reuse, so that it prevents names from being reused and potentially violating the *stable reference semantics* required by asset management. Finally, HATRAC supports a *hierarchical naming scheme* and allows users to define *access control policies* on assets by name and by subtree names in the name space to simplify management of access controls. At present, HATRAC supports two configurations: it may be deployed as a standalone server with assets stored on a local or remote file system; or it may be deployed on Amazon AWS with assets stored in the Amazon S3 object store.

D. IObox

IObox is a suite of modular utilities for ingest and export of assets to and from DERIVA and external data sources. Asset management for science must support diverse sources and formats. The utilities of IObox may be combined in a variety of configurations to support the unique requirements of different scientific applications. The utilities in this project are generally categorized in terms of extract, transform, or load (ETL) operations and uses the BDBag package format. In general, *extract* operations acquire assets from data sources (files or databases) and generate a BDBag package; *transform* operations alter the format or structure of the contents of a BDBag package; and *load* utilities take a BDBag package and upload the contents to catalogs and storage services.

At present, the utilities in the IObox suite include:

- *bag2dams*: takes a bag and imports it into DERIVA data stores;
- *dams2bag*: extracts metadata and assets from DERIVA data stores, serializes the contents, and generates a bag;
- *sql2bag*: connects to an ODBC-compatible database management server (e.g., Microsoft Access, Microsoft SQLServer, MySQL, etc.), executes user-specified queries, serializes the results and generates a bag;
- *xls2bag*: parses a Microsoft Excel spreadsheet and generates a bag;
- *xml2bag*: parses eXtensible Markup Language (XML) documents, serializes in tabular format, and generates a bag;
- *iobox-win32*: uses Win32 APIs to “watch” a Microsoft Windows file system, as files are generated asynchronously by connected instruments (e.g., microscopes, sequencers, etc.), it executes regular expression rules to identify files and extract metadata, and ingest directly to the DERIVA DAMS.

In addition, we have developed or integrated parsers for many important data formats for scientific assets, including but not limited to HDF5, NetCDF, CSV, Excel, TIFF, CZI, OME, NIfTI, DICOM, FASTA, FASTQ, and VCF.

E. BDBag

BDBag is a specification for asset aggregation packages. BDBag extends *The BagIt File Packaging Format (V0.97)*², incorporates the *BagIt Profiles Specification*³, and adopts the *Research Objects* [14] semantic model for describing packages and provenance. The BDBag utilities are a collection of software programs for working with the enhanced specifications for bags. These utilities combine various other components such as the BagIt creation utility and the BagIt profile validator utility into a single, easy to use software package.

F. WebAuthN

WebAuthN is a compact, modular authentication provider framework written to support Python-based, RESTful Web services and is used by ERMREST and HATRAC. It allows deployment-time configuration of several alternative identity and attribute provider modules to establish client security contexts for Web requests by talking to a local or remote provider. The **client provider** is used when establishing the client (or user) identity while the **attribute provider** is used when establishing additional attributes, such as roles or groups associated with the client identity.

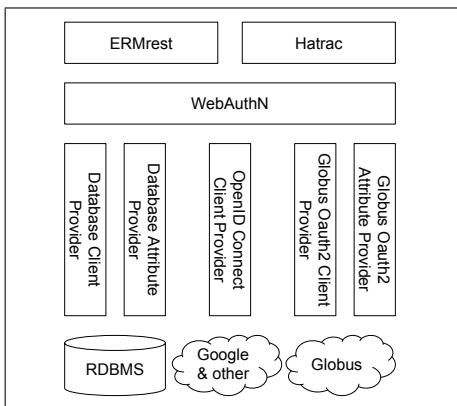


Fig. 6. WebAuthN layered architecture.

At present, WebAuthN is distributed with three identity provider (IdP) integrations:

- *OAuth2 OpenID Connect Provider*: any standard OpenID Connect IdP, such as Google, for client authentication;
- *Globus OAuth2 Provider*: Globus IdP which also supports delegated group management and access to numerous campus IdPs;
- *Standalone Database Provider*: standalone database IdP that can be deployed with the DERIVA suite.

The interfaces are well-defined and alternative implementations may be developed thus integrating different IdPs (e.g., LDAP, Active Directory, etc.) into DERIVA easily.

V. USE CASES

To validate the utility of a DAMS based approach to scientific data management as well as the applicability of the

²<https://tools.ietf.org/html/draft-kunze-bagit-13>

³<https://github.com/ruebot/bagit-profiles>

DERIVA platform, we have applied DERIVA to a range of different eScience use cases. In each situation, the resulting solution was provided to domain scientists who are using it as part of their ongoing scientific explorations. Many of these case studies are in their initial phase and we do not have quantitative data as to the impact of DAMS based approaches. However, from these studies we can conclude that the underlying DERIVA platform can be readily adapted to diverse domains, and that the domain scientists report that these systems are simplifying the process of getting their science accomplished.

Generation of atlases of kidney development. The goal of this collaboration is to collectively annotate high-resolution microscope images so as to trace the development of anatomical structure in the developing human kidney. DERIVA automatically ingests images taken from a microscope and puts them into the asset catalog during which high-quality images are identified based on visual inspection. The domain model for images was extended to include annotation locations, an anatomical term obtained from a controlled vocabulary and a running set of comments on the annotation. Facets are used to manage annotation status. Policy mechanisms are used to separate the ability to annotate, comment, and make the final decision on the annotation value.

Platform for Phenotype Wide Expression (PheWAS) from neuroimaging studies. For a given brain scan, it is possible to computationally produce a large number of phenotypes such as the size, density and curvature of each region of the brain. By aggregating these phenotypes across multiple subjects it is possible to make connections to specific genetic conditions. In practice, domain scientists become rapidly overwhelmed by the tens of thousands of files that contain the phenotype values, by keeping track of image based quality control, and the association of statistical analysis to specific data sets. We are using DERIVA to automatically ingest all of the analysis results and associate them with images, to track necessary manual review of some of the results, and to assemble data sets of phenotype to input to statistical analysis tools to look for significant correlations.

Determination of three dimensional structure of G-Protein Coupled Receptors (GPCR). Protein structure determination by X-Ray diffraction requires many steps to synthesize and analysis steps to crystallize and measure the protein. At each stage measurements are made and analyzed, and at the final step, large amounts of diffraction data must be collected and processed to reveal the protein structure. We are using DERIVA to manage the data that is being generated by a multi-site consortium. Our platform is automatically acquiring and integrating data spanning protein design, flow cytometry, chromatography and gel electrophoresis. Policy mechanisms distinguish between academic and industrial affiliates.

VI. RELATED WORK

Digital asset management systems have been used widely by creative and business organizations, however, the closest comparisons supporting science may be imaging [15] and

microscopy management systems [16]. There is a lack of general-purpose asset management capabilities that can span a wide range of multi-domain, multi-modal scientific data and which support the dynamic, rapidly evolving, heterogeneous research activities. Metadata catalogs [17] provide a useful building block but do not provide a full solution. Electronic Notebooks such as IPython and Jupyter, are another approach to organizing and visualizing scientific data. However, these approaches do not provide facilities to capture data from instruments, and they are not intended to manage large volumes of data throughout their many transformations.

Current systems and tools to support data-driven discovery, such as computational pipeline systems (e.g. [18]) tend to focus on computation and data analysis. While analysis is clearly important, we assert that data is the currency around which discovery is made and we should take the perspective that discovery is largely data-centric, not process-centric.

Digital repository systems, such as DSpace [19] and Globus Publish [20], may be used to develop institutional repositories which support preservation of digital works and enable open access to data. Digital repositories are primarily concerned with publication, as opposed to the discovery process itself where one's understanding of the domain model may evolve considerably.

Dataspaces [21], and SQLShare [22] also advocate the need for incrementally expanding and evolving data workspaces for individual or collaborative usage. [23] also argue that introspection is a necessity for dynamic, fluid databases, a concept that we further develop in our approach. While these approaches agree with our observations of rapidly evolving, heterogeneous scientific data, they only address query and analysis not capture of assets, organization and annotation.

VII. CONCLUSIONS AND FUTURE WORK

The challenges of data management in eScience applications require a new data-centric approach and we have identified digital asset management as being suitable. We have identified a common set of principles for applying asset management to scientific data and have shown through diverse use cases that a platform built on these principles can benefit a broad range of eScience applications.

In future work, we plan to provide tighter integration of ontologies into the platform. Of particular interest is in defining and refining ontologies as part of the evolution process. Currently, automation is implemented in a combination of cron jobs and shell scripts. We plan to develop a more complete automation service around event-condition-action rules to trigger automation activities and to allow user interfaces to specify automation activities on given events. We also plan to provide more user friendly tools for dynamically evolving the data model. We will need to introduce model versioning along with value versioning as part of this work. Finally, we are integrating online analytics into the system, with the goal of allowing the presentation and interaction to adapt to common usage patterns.

ACKNOWLEDGMENTS

The authors wish to thank the DERIVA developers: Jennifer Chen, Joshua Chudy, Mike D'Arcy, Laura Pearlman, Mei-Hui Su, Jessie Wong, and Serban Voinea. Alejandro Bugacov, Anoop Kumar, Hongsuda Tangmunarunkit, and Cristina Williams have contributed significantly to use case analysis and deployments. We also thank our science collaborators, Kristi Clark, Andrew McMahon, Jill McMahon, Seth Ruffins, Michael Hanson, Raymond Stevens, Scott Frasier, Donald Arnold, and William Dempsey.

REFERENCES

- [1] P. Fox and J. Hensler, "The Science of Data Science," *Big Data*, vol. 2, no. 2, pp. 68–70, jun 2014.
- [2] C. Goble *et al.*, "Accelerating Scientists' Knowledge Turns," in *Communications in Computer and Information Science*, 2013.
- [3] B. L. Claus and D. J. Underwood, "Discovery informatics: its evolving role in drug discovery," *Drug Discovery Today*, vol. 7, no. 18, pp. 957–966, sep 2002.
- [4] S. Kandel *et al.*, "Enterprise data analysis and visualization: An interview study," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 18, pp. 2917–2926, 2012.
- [5] R. G. Steen *et al.*, "Why Has the Number of Scientific Retractions Increased?" *PLoS ONE*, vol. 8, no. 7, 2013.
- [6] C. G. Begley, "Six red flags for suspect work." *Nature*, vol. 497, no. 7450, pp. 433–4, may 2013.
- [7] J. Gray *et al.*, "Scientific data management in the coming decade," *SIGMOD Rec.*, vol. 34, no. 4, pp. 34–41, 2005.
- [8] I. M. J. C. R. Licklider, "In Memoriam: J. C. R. Licklider 1915–1990," 1990.
- [9] T. Clark *et al.*, "Micropublications: a semantic model for claims, evidence, arguments and annotations in biomedical communications," *Journal of Biomedical Semantics*, vol. 5, no. 1, p. 28, 2014.
- [10] B. Plale *et al.*, "SEAD Virtual Archive: Building a Federation of Institutional Repositories for Long-Term Data Preservation in Sustainability Science," *International Journal of Digital Curation*, vol. 8, no. 2, pp. 172–180, nov 2013. [Online]. Available: <http://ijdc.net/index.php/ijdc/article/view/8.2.172>
- [11] B. Howe *et al.*, "Database-as-a-Service for Long-Tail Science," Portland, Oregon, 2011.
- [12] S. Jain *et al.*, "SQLShare: Results from a Multi-Year SQL-as-a-Service Experiment," in *SIGMOD'16*. San Francisco, CA, USA: ACM, 2016.
- [13] L. Moreau, "The Foundations for Provenance on the Web," *Foundations and Trends® in Web Science*, vol. 2, no. 2-3, pp. 99–241, 2010.
- [14] S. Bechhofer *et al.*, "Why linked data is not enough for scientists." *Future Generation Computer Systems*, vol. 29, no. 2, pp. 599–611, 2013.
- [15] D. S. Marcus *et al.*, "The Extensible Neuroimaging Archive Toolkit: an informatics platform for managing, exploring, and sharing neuroimaging data." *Neuroinformatics*, vol. 5, no. 1, pp. 11–34, 2007.
- [16] J. R. Swedlow *et al.*, "Bioimage informatics for experimental biology," *Annual review of biophysics*, vol. 38, pp. 327–46, jan 2009.
- [17] E. Deelman *et al.*, "Grid-based metadata services," in *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*, jun 2004, pp. 393–402.
- [18] J. Goecks *et al.*, "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences." *Genome biology*, vol. 11, no. 8, p. R86, 2010.
- [19] M. Smith *et al.*, "DSpace: An Open Source Dynamic Digital Repository," *D-Lib Magazine*, vol. 9, no. 1, 2003.
- [20] K. Chard *et al.*, "Globus data publication as a service: Lowering barriers to reproducible science," in *11th IEEE International Conference on eScience*, 2015.
- [21] M. Franklin *et al.*, "From Databases to Dataspaces: A New Abstraction for Information Management," 2005.
- [22] B. Howe *et al.*, "Automatic example queries for ad hoc databases," in *Proceedings of the 2011 international conference on Management of data - SIGMOD '11*. New York, New York, USA: ACM Press, jun 2011, p. 1319.
- [23] A. Halevy *et al.*, "Principles of Dataspace Systems," Chicago, Illinois, USA, 2006.

Cooperative Human-Machine Data Extraction from Biological Collections

Icaro Alzuru, Andréa Matsunaga, Maurício Tsugawa, José A.B. Fortes

Advanced Computing and Information Systems (ACIS) Laboratory
University of Florida, Gainesville, USA

Abstract—Historical data sources, like medical records or biological collections, consist of unstructured heterogeneous content: handwritten text, different sizes and types of fonts, and text overlapped with lines, images, stamps, and sketches. The information these documents can provide is important, from a historical perspective and mainly because we can learn from it. The automatic digitization of these historical documents is a complex machine learning process that usually produces poor results, requiring costly interventions by experts, who have to transcribe and interpret the content. This paper describes hybrid (Human- and Machine-Intelligent) workflows for scientific data extraction, combining machine-learning and crowdsourcing software elements. Our results demonstrate that the mix of human and machine processes has advantages in data extraction time and quality, when compared to a machine-only workflow. More specifically, we show how OCropus and Tesseract, two widely used open source Optical Character Recognition (OCR) tools, can improve their accuracy by more than 42%, when text areas are cropped by humans prior to OCR, while the total time can increase or decrease depending on the OCR selection. The digitization of 400 images, with Entomology, Bryophyte, and Lichen specimens, is evaluated following four different approaches: processing the whole specimen image (machine-only), processing crowd cropped labels (hybrid), processing crowd cropped fields (hybrid), and cleaning the machine-only output. As a secondary result, our experiments reveal differences in speed and quality between Tesseract and OCropus.

Keywords—Digitization; human-machine; data extraction; biological collections; optical character recognition; crowdsourcing

I. INTRODUCTION

The extraction of information from historical data sources, like medical records and scientific collections, is a challenging task. Standards were not used or have changed since these paper documents were created, and standards will continue to evolve. These data sources mix typed, printed, and handwritten text on paper that in some cases already turned yellow or stained.

Nevertheless, the information stored in these documents is a valuable heritage, which helps us understand the past, and more importantly could allow us to forecast and improve our future. Biological collections, for example, could help us model past and future environmental changes, develop new medicines, control agricultural pests, and understand or avoid epidemics, among many other benefits [1].

Governments and institutions have recognized the value of these biological collections and the importance of providing access to the cataloged specimens not only to researchers but to the general public. Projects like the Integrated Digitized

Biocollections (iDigBio: <https://www.idigbio.org>), the Global Biodiversity Information Facility (GBIF: <http://www.gbif.org>), and the Atlas of Living Australia (ALA: <http://www.ala.org.au>), are stable providers of universal access to millions of specimens.

The challenge is that the actual number of specimens to digitize, as stored in collections worldwide, has been calculated in more than a billion [4]. Considering the current digitization rate, which is on the order of minutes per specimen, these data could take several decades to be processed [3]. This time does not account for the time to train the personnel who perform the digitization or the domain expert's time to manage the process and validate results. For mass-digitization of specimens, the recommended approach has been to divide the process into image capture and metadata transcription stages [5]. This alleviates the need for institutions to prioritize only the “most important” collections or specimens, and focuses the effort on curation and scanning. The transcription of the text can be performed at a later time, using the captured image, which opens the possibility of crowdsourcing [6][7]. In this work, we follow this mass-digitization approach, and assume that there are millions of specimen images that require data extraction.

Some researchers believe “there is no point in collecting complete metadata if these are not going to be used for any purpose” [5], but in this Big Data era, which has been defined as “collect now, sort out later” [9], we should not decide what metadata is important and what is not. The digitization process needs to be accelerated in order to digitize all historical data sources, ensuring the best result quality possible, and using experts only when strictly needed (mainly for verification). Experts should ensure the digitization’s quality, while crowds and machines make high volume data processing possible. It is this mix of human- and machine-intelligent processes where we believe is the ideal solution. The goal of this study is to demonstrate that a good balance of these processes, in a single workflow, can lead to an improved overall process.

Today, a pure machine-intelligent process to transcribe text from biological specimen images using Optical Character Recognition (OCR) tools does not produce a good result. This kind of historical data source includes a wide range of challenges for the OCR tools, whose recognition algorithm is based on training. The variety of font types (printed and typed) and sizes, languages, background colors, lines, stains, and other elements in the image affect the recognition rate of the OCR. The other alternative, pure human-intelligent approaches [7][8], like crowdsourcing, can get better accuracy, but deal with a different set of challenges, like training the crowd and handling consensus among the diverse set of results.

In this paper, we propose to improve the overall output quality by combining the OCR execution (a machine-intelligent process) with the help of humans (crowdsourcing), who crop the text areas that the OCR has to process. The cropping reduces the complexity, noise and size of data that the OCR tool has to binarize, segment, and recognize; leading to higher data quality and time-saving. The crowdsourcing step has been simplified to only require cropping of text, which does not demand a complex training and requires less effort than full text transcription. Crowdsourcing tasks were performed using a cropping web application developed for the HuMaIN (Human- and Machine-Intelligent Network) project, see <http://humain.acis.ufl.edu>.

To demonstrate that the cooperative human-machine data extraction improves the quality of the OCR process, getting closer to the ideal (expert-equivalent) result, we use 400 specimen images of the iDigBio project, for which the experts' transcription of the label and fields are known. These images cover three specimen types: Entomology (i.e. Insect), Bryophyte, and Lichen. Four approaches are evaluated:

1. Machine-only: the OCR tool is run on the whole original image, and the result is compared to the experts' label transcription. This establishes a comparison baseline.
2. Hybrid: humans crop the entire labels (using the HuMaIN interface [32]) from the specimen images and the OCR is run on them. The results are compared to approach 1.
3. Hybrid: considering a higher granularity level, humans crop individual fields (using the HuMaIN interface) from the specimen images and the OCR is run on them. The result is compared to the two previous approaches.
4. Improved machine-only: due to the amount of extra characters and digital noise generated by the OCR on the whole image, we add a simple cleaning algorithm on the machine-only results to compare between explicit noise removal, through cropping, and implicit noise removal, through elimination of non-interpretable set of characters.

Our results show that OCR's recognition rate improves by at least 42% for any of the two approaches where the text areas are cropped. In order to guarantee that the obtained results are independent of the used software or metric, two OCR tools (OCRopus and Tesseract) and three string similarity metrics (Damerau-Levenshtein, Jaro-Winkler, and the rate of matching words) were used. The total execution time of the cooperative data extraction process (machine + human) was reduced when using OCRopus, but increased when using Tesseract. This is due to the fact that Tesseract is faster than OCRopus while both provide similar recognition quality. None of the OCR tools were trained or tuned, i.e., the default version of the English language dictionary that these tools provide was used.

II. RELATED WORK

The Human-Computer Cooperation field is broad. In the HuMaIN project, and specifically in this study, we show the benefits of this interaction and how it can be applied to improve data extraction. There are data sources for which an automated data extraction process is sufficient to get the information we need. But for other data sources, like scientific collections, the data extraction must still rely on humans.

In 2011, iDigBio created the Augmenting OCR Working Group (A-OCR) with the goal of generating tools that improve the OCR process, either in output quality, speed, cost, or efficiency [10]. The group has organized several events which have generated a number of software solutions. One of these tools is the Semi-automatic Label Information Extraction system (SALIX) [11][12]: using a friendly graphical interface, the user can run the OCR and SALIX automatically assign text into individual fields. Corrections can be applied to the result. We agree with SALIX in the semi-automatic (machine and human) nature of the solution, considering how difficult it is to obtain a perfect OCR output for this type of data source. Nevertheless, in this work, our goal is to improve the output of the OCR, which would be the input of SALIX, i.e., we do not create a natural language post-processing tool. Moreover, our focus is on open source tools, while SALIX is mainly tested with ABBYY®, a proprietary OCR tool.

The Apiary project (<http://www.apiaryproject.org>), of the Botanical Research Institute of Texas (BRIT), has developed a "High-Throughput Workflow for Computer-Assisted Human Parsing of Biological Specimen Label Data" [14]. HuMaIN follows the same spirit as Apiary, with the difference that even though we are using biological collections as a use case, our final goal is a general platform for the definition of hybrid data extraction workflows in any area, generalizing the hybrid (Human-Computer) concept. Apiary includes a web application, called HERBIS, which inspired SALIX and works similarly.

A good amount of scientific projects (Apiary, the Atlas of Living Australia, Les Herbonautes, and Symbiota [34], among many others) have developed products and conducted research to digitize specimens, creating workflows that integrate crowdsourcing and machine-intelligent tools. This study uses a human task (cropping images) to improve the result of a machine-intelligent process (OCR), reinforcing the idea that we need both, humans and machines, to get an optimal result.

In the business world, several crowdsourcing vendors serve as a link between companies and crowds, like computer programmers, designers, or transcribers. One interesting case is CrowdFlower, which employs crowdsourcing in data extraction workflows to ensure or improve data quality, an area where crowdsourcing has been especially successful. Nevertheless, our initiative points to an open, customer managed, platform.

Several open source OCR products are available: Tesseract, JOCR (GOOCR - <http://jocr.sourceforge.net/>), and OCRopus (OCROpy), among others. In [15] a comparative study between Tesseract and GOOCR conclude that Tesseract has better accuracy and precision than GOOCR. Creators of OCRopus, show in [16] mixed results for Tesseract and OCRopus error rates. OCRORACT [13] uses an iterative method, as it trains Tesseract (segmentation based OCR) with the data from OCRopus (segmentation-free OCR) and then trains OCRopus with the data generated by Tesseract. After few iterations, the method is able to reduce the misinterpretation rate from 23% to 7%. OCRORACT requires an expert who provides the unique set of characters for the documents and their Unicode representation. Nevertheless, in the case of biological collections, with the diverse mix of typed, printed, and handwritten text, this training oriented approach would not be the most appropriate.

III. DOMAIN DATA SETS AND SOFTWARE TOOLS

This section describes the main characteristics of the images used in the study, the OCR tools, the Web application used to crop labels and fields, and the metrics employed to measure string similarity or quality of the results.

A. Data Set

Between 2011 and 2014, the Augmenting OCR Working Group (A-OCR) of the iDigBio project, organized several events to generate content and tools for the digitization community. Among their results, there is a dataset with 400 images of catalogued specimens [27] and their corresponding whole label transcription as well as information parsed into individual fields by domain experts' transcription. Individual fields correspond to Darwin Core standard terms [25]. These images of biocollection specimens belong to 3 specimen types: 100 insects, 100 bryophytes, and 200 lichens. Distinctive characteristics of the images are:

- **Entomology images** (i.e., insect images): include a picture of the insect and a ruler, which helps measure the size of the insect. Instead of a single metadata label, there are several pieces of paper varying in size, color, and border. Some images have a scientific name written at the bottom left corner, which is an annotation made by the current institution, not the original one. See upper right image of Figure 1.
- **Bryophyte images**: are the largest images of the experiment (generating longer OCR times). Besides the labels, they include the specimen and other elements like stamps, maps, and bar codes. There can be segments of handwritten text and labels can have different orientations, but the background and image are mostly clear. See left side of Figure 1.

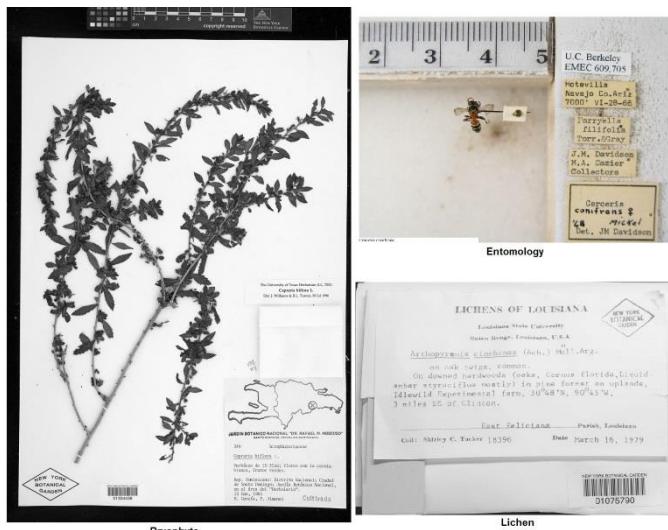


Figure 1. Bryophyte, Entomology, and Lichen images.

- **Lichen images**: do not include the specimen, and are basically big labels, but the resolution and contrast of the image are not the best. There is a stamp and a bar code in most of them. See lower right image of Figure 1.

Challenges in automatically extracting data from these images are: unformatted text, mixed with pictures, maps, stamps, and bar codes; different fonts and sizes; several

languages; different background colors and resolutions; handwritten and underlined text. Some of their technical characteristics are specified in the following table:

Table 1. Number, size, and resolution of the specimen types

Specimen type	Number of images	Avg. Size (KB)	Dimension	Resolution (dpi)
Entomology	100	325	1600x1200	180
Bryophyte	100	1214	3744x5616	300
Lichen	200	153	1530x1128	96

B. Optical Character Recognition (OCR) Technology

The OCR process is the extraction, in machine encoded-format, of the typed, printed, and handwritten text of an image [21]. This process consists of a sequence of machine learning steps. In order to add generality and robustness to our results, two OCR tools are used:

OCRopus (OCRopy): is a group of open-source document analysis programs, which can be integrated as a Character Recognition System [17]. Its modularity makes it ideal for code reuse and teaching purposes. It requires running three steps or programs in sequence: the binarization (creates a black & white version of the image), the segmentation (divides the image in multiple text segments), and the recognition (applies to each segment the Recurrent Neural Network for text identification). We created a script to perform the execution of the three OCR steps [22]. In our study, OCRopy v1.0 and its ad-hoc English model were used, no further training or configuration was done. OCRopus recommends a 300dpi resolution as a minimum, which only bryophyte images comply.

Tesseract: is an open-source OCR engine initially developed by HP (between 1984 and 1995). It is sponsored, since 2006, by Google [18] [19]. Tesseract recommends a minimum resolution of 300dpi. Its artificial intelligence model must be trained, we used the default English trained model available in Tesseract 3.04. Tesseract is executed in a single step or program.



Figure 2. HuMaIN interface for fields cropping.

C. HuMaIN

The Human- and Machine-Intelligent Network (HuMaIN – <http://humain.acis.ufl.edu>) is a project in development by the Advanced Computing and Information Systems (ACIS) laboratory (<https://www.acis.ufl.edu>) of the University Florida. It is funded by the National Science Foundation, with the goal

of researching and creating hybrid (crowdsourcing and machine learning) workflows of software components for data extraction.

Figure 2, shows the interface developed for cropping the fields of the images. The user picks the field name from one of the list boxes, selects the area where the value of the field is, and clicks on the green arrow button of that field. After selecting all the fields of a specimen, the user clicks on the “Save and Next” button to store the coordinates in the database. Later, these coordinates are used to generate cropped images of each field.

The label cropping interface works in a similar way, but the user only needs to select one text area. These two web applications were used to get the data for approaches 2 and 3. They are available at <http://humain.acis.ufl.edu/app.html>.

D. Metrics

Comparing strings is a common Data Science problem. Many similarity metrics are token-based, which make them unsuited to compare sentences. Token-based metrics, require strings to have the same length and would penalize too much the characters inserted or omitted by the OCR, being that these events modify the absolute position of the characters that follow. Due to these reasons, it was decided to use edit-based metrics.

- **Damerau-Levenshtein (DL) similarity:** The DL distance of two strings is the minimum amount of insertions, deletions, substitutions, and transpositions of two adjacent characters, required to convert one string into the other [31]. The DL similarity is computed as the complement to 1 of the normalized DL distance:

$$sim_{DL}(x, y) = 1 - \frac{DL\ distance(x, y)}{\max(|x|, |y|)} \quad (1)$$

For this study, the Geoffrey Fairchild’s DL normalized distance implementation of the algorithm [33] is used.

- **Jaro-Winkler (JW) similarity:** The JW algorithm considers the number of matching characters and adjacent transpositions, giving better rating to the letters that match at the beginning of the string [24]. The JW distance is not a metric in the mathematical sense [20], while its results are normalized (range 0 - 1), they do not represent a real distance. In our study, it was used the JW implementation available at the jellyfish 0.5.3 library [23].
- **Matched words (mw) rate:** This is an empirical metric, $mw(x, y)$ is equal to the number of words of x that are in y , divided by the number of words in x :

$$mw(x, y) = \frac{|words\ in\ common\ between\ x\ and\ y|}{|x|} \quad (2)$$

The order and frequency of words are not considered [22].

For these three metrics, a 0 (minimum value) means totally different strings, while 1 (maximum value) implies the strings are exactly the same (or “it is included in” for mw).

IV. EXPERIMENTAL SETUP, RESULTS AND ANALYSIS

In this section, the experimental setup is detailed and the four approaches explained in Section I are evaluated with regard to consistency of outputs and the effectiveness of the human-machine cooperation.

A. Experimental Setup

The machine used to run OCropus and Tesseract has the following characteristics:

Hardware:

- System: IBM BladeCenter HS22 7870-AC1
- CPU: 2x Intel Xeon, E5540 (8 cores/16 HyperThreads)
- Storage: HGST HTS725050A7 (HD, 2.5 Inch, 500 GB)
- Memory: 48 GB, 12 x 4GB PC3-10600R DDR3 RAM

Software:

- CentOS Linux release 7.2.1511
- Python 2.7.5 (default, Nov 20 2015, 02:00:19)
- gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-4)
- Tesseract 3.04.00, with leptonica-1.72
- OCropy v1.0 (OCropus)

B. Approach 1 (Machine-only – OCR whole image)

The OCR process was executed on the 400 images (divided in Entomology, Bryophyte, and Lichen specimens) using OCropus and Tesseract; and evaluated considering the Damerau-Levenshtein (DL), Jaro-Winkler (JW), and matching words (mw) similarity metrics. The average similarity, with respect to the experts’ transcription, is shown in Figure 3 and summarized, for the DL metric, in table 2.

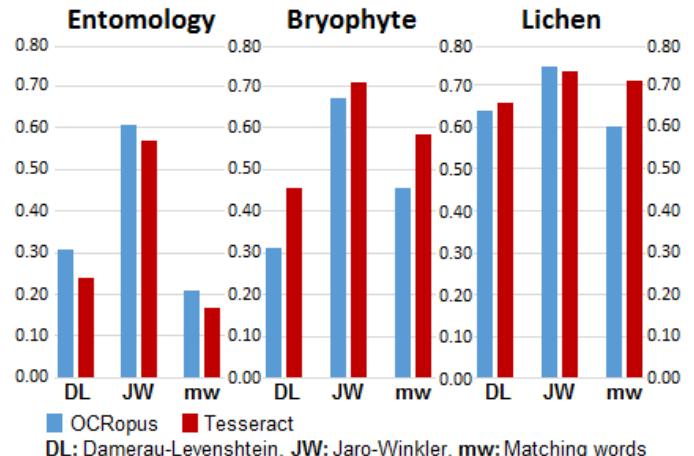


Figure 3. Average similarity per specimen type, OCR tool, and metric

The character recognition process worked better for lichens, followed by bryophyte and entomology images, considering the values obtained for the Damerau-Levenshtein (DL) metric.

Table 2. Average Damerau-Levenshtein (DL) similarity

	Entomology	Bryophyte	Lichen
OCropus	0.31	0.31	0.64
Tesseract	0.24	0.46	0.66

Despite having the lowest resolution (only 96dpi), the text in lichen images is the easiest to be interpreted by the OCR: the three similarity metrics used were better for lichen images than for entomology and bryophyte images. In lichen images more than 60% of the words are recognized in both OCR technologies. These images are basically the label of the specimen, with few stamps, bar codes, or graphical elements in them, and a mostly white background.

Entomology images have text with non-white background, many lines (boxes and underlined text), and dark or shadowed regions around the text labels, that create additional borders or lines which mix with the text. Even though bryophyte images have more graphical elements than entomology images, their clean background, better resolution and contrast, make them get more accurate OCR results.

In Figure 3, the matching words (mw) metric, a string similarity measure which can be thought as more restrictive than the DL and JW metrics, got better or similar results (for Bryophyte and Lichen) than the DL metric. This shows that many of the words in the image are recognized, but added noise characters make the DL algorithm getting a worse similarity.

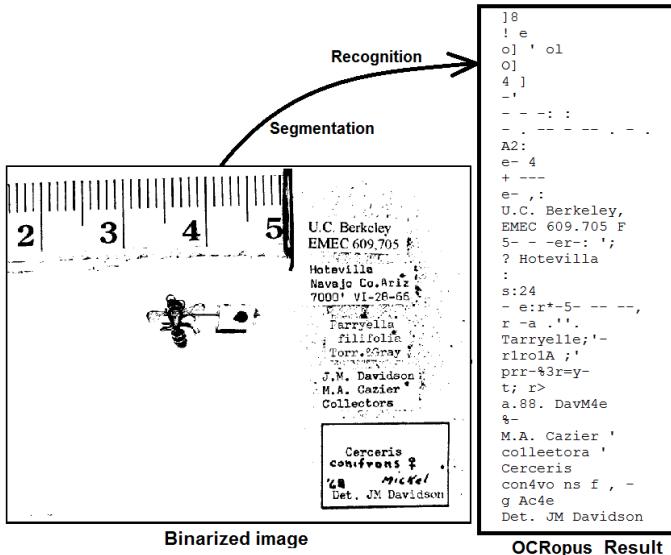


Figure 4. Result of the binarization (left) and interpreted text (right) of specimen EMEC 609,705

The result of the OCropus binarization process for the entomology image presented in Figure 1 is illustrated in Figure 4. After the binarization, OCropus executes the segmentation and recognition processes, to get the final result (right).

If the background is not completely white, the binarization process removes some pixels from the characters, making these letters more difficult to recognize. The dark areas around the small labels, create figures that the OCR tries to interpret. Boxes and underlined text also confuse the recognition algorithms.

For the specimen in Figure 4 (EMEC 609,705), the following similarity results were obtained:

Table 3. Similarity values obtained for specimen EMEC 609,705

Similarity \ OCR software	OCropus	Tesseract
Damerau-Levenshtein	0.3627	0.3941
Jaro-Winkler	0.5943	0.6514
Matching words	0.4	0.4

In general, Jaro-Winkler metric shows more “optimistic” results, returning a higher similarity value than the Damerau-Levenshtein and matching words similarity metrics.

The execution time of the OCR process is shown in table 4. Tesseract was in average 18.5 times faster than OCropus. During the study, none of the tuning features these tools provide

were utilized. The main reason for the execution time difference is that OCropus generates intermediate on-disk results during the binarization and segmentation steps, while Tesseract works entirely in memory, as a single process.

Table 4. Approach 1 – OCR’s average execution time (s)

Specimen type \ Tool	Avg. Execution Time (s)	
	OCropus	Tesseract
Entomology	28.36	3.60
Bryophyte	158.57	4.54
Lichen	30.46	1.95

In Figure 5, it is shown the similarity box chart for approach 1. We observe the Jaro-Winkler metric offers less variability in the results. We also notice high maximums (images where almost every word was identified) and very low similarity results (images for which the OCRs were not able to correctly identify a single word).

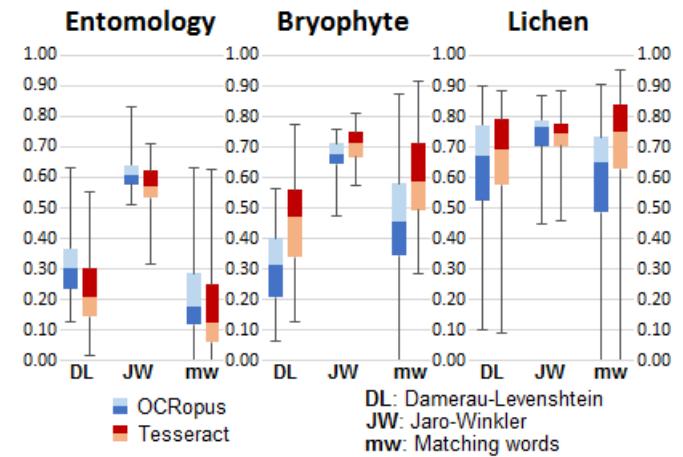


Figure 5. Box and whisker representation of the obtained similarity per specimen type, metric, and OCR tool.

Lichen images have some attributes which make their text easier to be recognized than the other specimen types, but not all the images have the same quality. One of the images with poor conditions is lichen TENN-L-0000003, see Figure 6, for which the DL similarity was 0.1 in OCropus and 0.15 in Tesseract; and none OCR tool was able to match a single word. The low contrast of the image makes it difficult for the OCR tools generate good results. Increasing the contrast would improve these cases, but another machine process would be needed to auto-select the level of contrast, as elevating contrast for all will cause other labels to decrease their recognition rate.

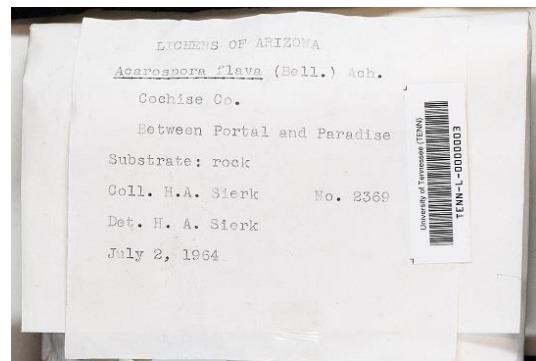


Figure 6. Lichen TENN-L-0000003

C. Approach 2 (Cooperative – Crop and OCR label)

Using the HuMaIN interface [32], the label of the 400 images were cropped by two volunteers. The result of this process is a rectangular image with the text in it. In the case of entomology images, there are several pieces of paper with data, hence the rectangle includes all these areas. In the case of bryophyte images, the final cropped label may include pieces of other elements, which reverts to Approach 1 in these cases. Figure 7 shows the cropped version of the images in Figure 1.



Figure 7. Cropped labels of images in Figure 1

The HuMaIN's app, randomly picks, the next image to be cropped. The coordinates of each cropped label are stored in a database, as well as the time the user spent in the process. Every label was cropped at least three times by the volunteers. The consensus criteria to select the coordinates (image) to process with the OCR was choosing the image with the largest area.

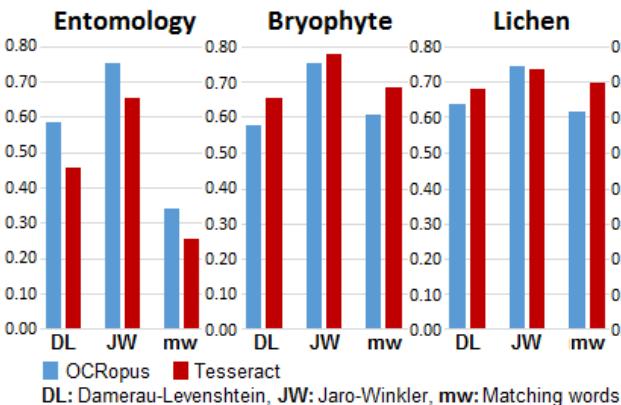


Figure 8. Average similarity for cropped label images

Figure 8 shows the DL, JW, and mw similarity values for the cropped entomology, bryophyte, and lichen labels, OCRed with OCropus and Tesseract.

Figure 9 exhibits the absolute similarity variation when executing the OCR on the original images (Approach 1) vs. the cropped labels. For lichen images, there was no significant variation because the cropped labels are very similar to the original images. Lichen images are basically a cropped label.

Approaches 1 vs. 2 - Similarity variation

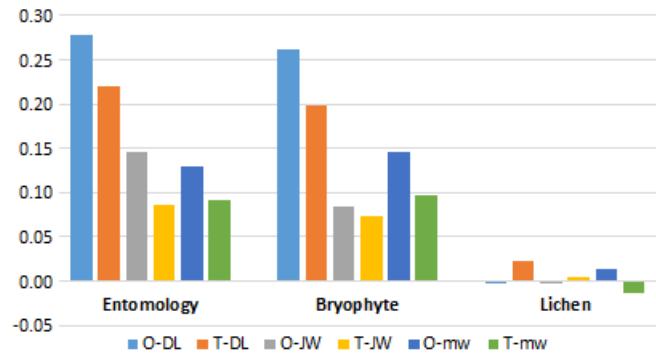


Figure 9. Average similarity difference between approaches 1 and 2

In the case of entomology and bryophyte images, there was a clear improvement. The similarity of the OCR generated text with respect to the experts' transcription improved about 0.22 considering Damerau-Levenshtein, 0.08 for Jaro-Winkler, and 0.12 considering the matching words similarity metric. The DL metric showed a higher improvement than the JW metric, likely because JW similarity values were already high for Approach 1.

Figure 10 shows the cropped text area of specimen EMEC 609,705 after being binarized, and the final OCropus result. In comparison to Figure 4, we can observe that the initial undesired characters were eliminated and a pair of zones (around "Navajo" and the first appearance of "J.M. Davidson") were better interpreted.

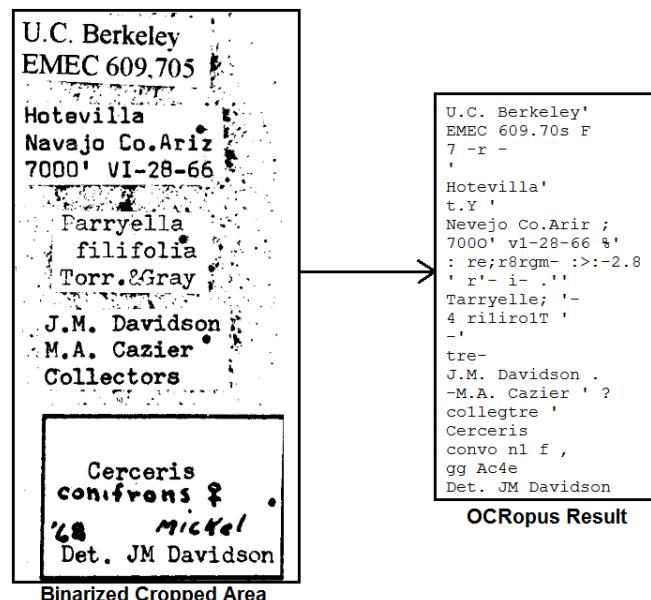


Figure 10. Result of the binarization (left) and interpreted text (right) of the cropped area of specimen EMEC 609,705

In the box chart of Approach 2 (Figure 11) it is remarkable how the matching word metric for some bryophyte images raised to 100%. This increment together with the average results for approach 2 shown in Figure 8, confirm that cropping the text area (made by humans) before running the OCR, improved the quality of the OCR output (a machine-intelligent process).

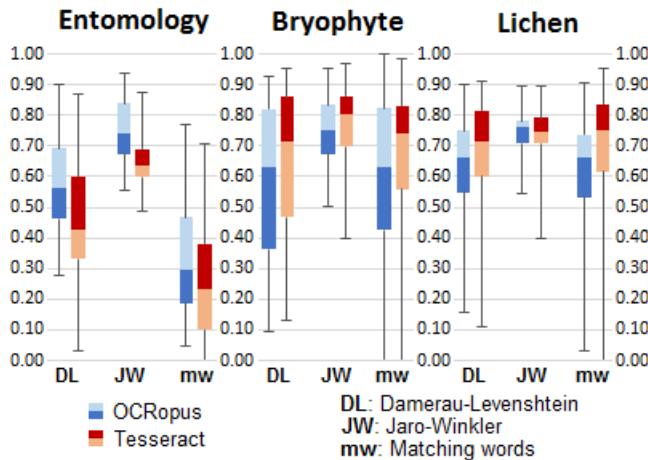


Figure 11. Similarity of the cropped labels per specimen type, metric, and OCR tool.

Comparing figures 11 and 5, we also observe that the variability increased. Although the average OCR's recognition rate improved, there are images which OCR process did not benefit from processing a smaller version of them.

In Approach 1 the total time is equal to the OCR execution time. For Approach 2 (see Table 5), we include the time users spent cropping the image (O: OCRopus, T: Tesseract). This cropping time was measured as the interval between the web page loads and the user pressing the "Save and Next" button. In-between these events, the image loads, the user interprets the image, marks the area to crop, and clicks the "take coordinates" (green arrow) button, see Figure 3.

Table 5. Approach 2 - Average execution time (s)

Type \ Tool	Execution time (s)				
	Cropping	OCRopus	Tesser.	Tot. O	Tot. T.
Entomology	15.36	15.65	2.47	31.01	17.83
Bryophyte	24.56	32.74	1.68	57.30	26.24
Lichen	15.13	25.52	1.82	40.65	16.95

For entomology and lichen images the average cropping time was 15 seconds, but for bryophytes the process was more complex, and took on average 25 seconds. Considering only the OCR execution time, Approach 2 was on average 2.62 and 1.74 times faster than Approach 1, for OCRopus and Tesseract respectively. Considering the total time, Approach 2 was 1.48 times faster with OCRopus and 6.48 times slower with Tesseract, with respect to Approach 1.

D. Approach 3 (Cooperative – Crop and OCR fields)

Using the HuMaIN web interface, 8 Darwin Core fields: scientific name, event date, latitude, longitude, identified by, country, county, and state/province were cropped and then OCRed. It is important to highlight that only 6 of these fields are not modified or interpreted. The fields Scientific name, Event date, Latitude, and Longitude were collected as dwc:verbatim

[26], which means they are the original values present in the image. Additionally, the fields Identified by, and County are usually not modified or completed by the expert. On the other hand, State/Province and Country fields, use to be abbreviations which are completed or interpreted. For example, in the country field, "Mex." is interpreted as "Mexico"; while "US", "U.S.A.", and "USA" are interpreted as "United States". This affects the comparison because we are not doing these interpretations of the OCR output. As mentioned before, the objective of this study is not to develop a data extraction product as SALIX [29] or LabelX [30], but showing the benefits of the human-machine cooperation in data extraction. Some random examples of cropped fields are shown in Figure 12.



Figure 12. Examples of cropped fields

Every cropped field image was resized to 600 x 600 pixels because this is the minimum image size permitted by OCRopus. During this enlargement process, the cropped area was not changed, but the surrounding area was filled with "silver" (light gray) color. In preliminary tests, we filled the image with white background; but we found, after trying with white, silver, gray, and black colors, that silver gave the best result. In 20 test images, silver background increased the character recognition rate by about 12% (in OCRopus and Tesseract) with respect to white background. The reason for this difference is the artificial border around the cropped area created by the binarization process when white background is used. In the case of silver background, the contrast is reduced and the border disappears or is minimized, see Figure 13. The border shown around the scientific name at the left side of Figure 13 does not exist in the original image, see Figure 1 – entomology specimen. When the border is present, the recognition rate decreases.

White background



Silver background



Figure 13. Binarization result of the same cropped image, filled with white (left) and filled with silver (right)

The amount of cropped fields, per specimen type and field, is shown in the following table:

Table 6. Numbers of cropped fields per specimen type and field

	Entomology	Bryophyte	Lichen
dwc:country	7	75	63
dwc:county	55	0	52
dwc:verbatimEventDate	89	98	191
dwc:identifiedBy	51	55	89
dwc:verbatimLatitude	2	6	89
dwc:verbatimLongitude	2	8	97
aocr:verbatimScientificName	52	97	196
dwc:stateProvince	61	26	157

The “matching words” (mw) metric, used in the first 2 approaches and defined as the percentage of exact words recognized by the OCR, was not used in this case because most of the fields have only one or a very low number of words.

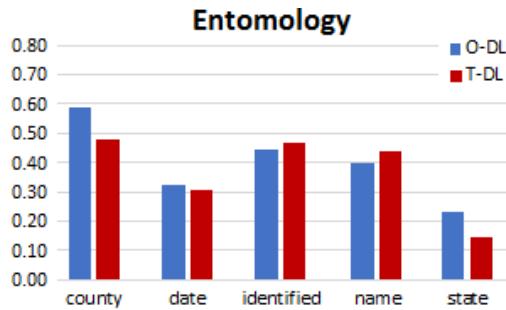


Figure 14. Average similarity for entomology fields

Figures 14, 15, and 16 present the similarity value for the entomology, bryophyte, and lichen fields. For simplicity purposes, we only show the results for the Damerau-Levenshtein similarity metric in OCropus (O-DL) and Tesseract (T-DL).

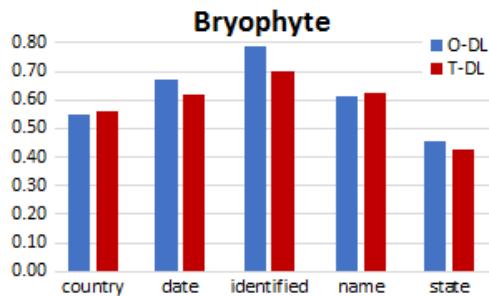


Figure 15. Average similarity for bryophyte fields

Figures 14 and 15 omit the fields for which we collected less than 10 values per specimen type, see Table 6. Entomology images show the worst results. Each field has its own challenges:

- The degree symbol in Latitude and Longitude is usually not recognized. The slash and hyphen symbols of the Event date also confuse the interpreters.
- Some Scientific names are underlined, which mixes with the letters and confuses the OCRs.
- State/Province and Country are sometimes abbreviated in the images and completed in the experts’ results. Hence, the value for these fields do not really represent the quality of the OCR’s output.

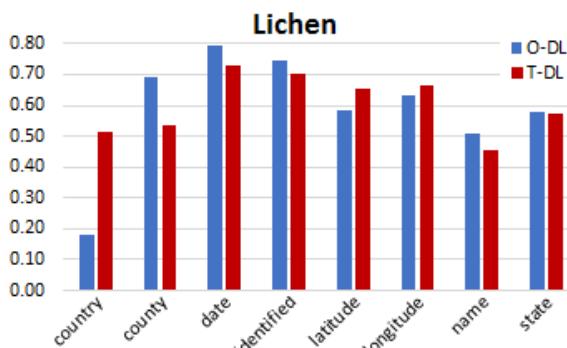


Figure 16. Average similarity for lichen fields

On average, “Identified by” was the easiest field to interpret; while Country and State fields, because they can be abbreviated, present the worst similarity. Lichen fields obtained a higher similarity than entomology and bryophyte fields. Table 7 shows the average Damerau-Levenshtein similarity per specimen type in each of the 3 approaches. Fields State/Province and Country were not considered to calculate the averages for Approach 3.

Table 7. Average DL similarity by approach

	Entomology	Bryophyte	Lichen
A1: whole image	0.31	0.31	0.64
A2: cropped label	0.59	0.58	0.64
A3: cropped field	0.44	0.69	0.66

Bryophyte and Lichen information was better interpreted in Approach 3 (Cropped fields), while entomology text was better recognized in Approach 2 (Cropped label). The result was not absolute in terms of better similarity for Approach 3, but data consistently show that when unnecessary areas are discarded, the OCR generates a better result.

Considering only entomology and bryophyte images, (since lichens images are already cropped labels), the similarity improvement when humans cropped the label or the fields, with respect to approach 1, was on average 0.27. Table 8 shows the percentage improvement with respect to the machine-only approach, obtained by the two cooperative approaches.

Table 8. Avg. DL improvement of A2 and A3 with respect to A1

	Entomology	Bryophyte
A2 vs. A1.	90%	87%
A3 vs. A1	42%	123%

The machine-intelligent process performed by the OCR was improved when a human-intelligent process (cropping the text areas), was added to the processing workflow.

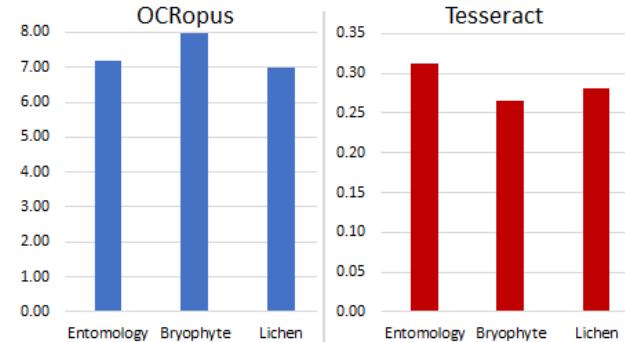


Figure 17. OCR Execution time (s) by tool and specimen type

Figure 17 shows that in OCropus, bryophyte images are the slowest in being processed, while for Tesseract those are precisely the fastest to process. It is important to note that the scales are different. On average, Tesseract spends 0.29 sec processing a field image, while OCropus takes 7.39 sec, which is about 25 times slower than Tesseract.

E. Approach 4 (Machine-only - Data cleaning)

In Approach 1, we executed the OCR process and observed that the results include unexpected characters, which could affect the quality of the similarity values. We developed a very simple and fast filtering script which omits the words (sequence of characters) that contain symbols which are uncommon punctuation characters [22].

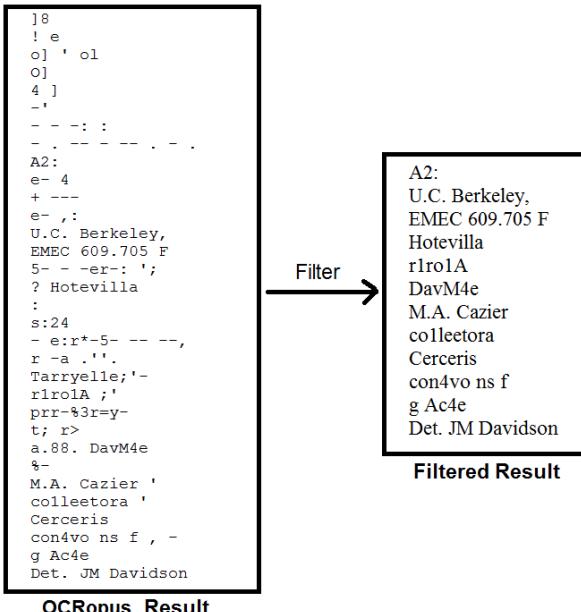


Figure 18. Data cleaning process: source (left), result (right)

Our objective was to simplify the output to improve the readability and verifying if the similarity values get better. Figure 18 shows the output after cleaning the OCRopus result of our example entomology specimen 609,705.

Table 9. Similarity variation when cleaning the Approach 1's output

	Damerau-L.		Jaro-Winkler		Matching w.	
	Ocro.	Tess.	Ocro.	Tess.	Ocro.	Tess.
Entomology	0.00	0.00	0.01	0.00	-0.01	-0.01
Bryophyte	0.08	0.01	0.02	-0.01	-0.04	-0.02
Lichen	-0.04	-0.02	-0.01	0.01	-0.03	-0.05

The cleaning script was executed on OCRopus and Tesseract results of Approach 1. The similarity values were recomputed for these outputs and we compared the obtained results with the Approach 1's similarity values. Table 9 shows the variation in similarity for each OCR tool, metric, and specimen type. The cleaning process did not improve the similarity, therefore the comparison of Approaches 2 and 3 with respect to Approach 1 is fair even after cleaning its output. Despite not improving similarity, the cleaning process reduced the output size and simplified the result, which is a positive behavior.

F. Time, Cost, and Scalability

Consider 1 billion of scientific images to be processed, a single server with a Total Cost of Ownership (TCO) of \$3000 per year [35], a single person with a base salary of \$10/ hour, an OCR with the average behavior between Tesseract and OCRopus, and the following 4 approaches:

0) Human-only: where the average transcription time is about 9 minutes [7]. Instead of DL similarity, the percentage of tasks where consensus is reached was used [7].

1) Machine-only: where the average OCR time is 37.9 sec (OCRopus and Tesseract average), see Table 4.

2) Cooperative-Crop Label: where the average crowdsourcing (cropping) time is 18.3 sec (see Table 5), and the average OCR time is 13.3 sec.

3) Cooperative-Crop fields: where 10 fields are transcribed per specimen, the average cropping time is 20 sec per field (according to our experiments with volunteers), and the average OCR time is 3.84 sec per field.

Time and cost can be estimated as summarized in Table 10. In general, adding human effort increases execution time and cost, but improves the quality of the result. Note that even though the total time in years is provided, both machine and human efforts are fully parallelizable.

Table 10. Time, Cost, and DL Similarity per Approach

Appr.	Human+Machine (Time in years)	Cost (\$ in Millions)	DL similarity
0	17123 – 0 (17123)	1500.00	0.79 [7]
1	0 – 1202 (1202)	3.61	0.42 (Table 7)
2	580 – 422 (1002)	52.10	0.60 (Table 7)
3	6342 – 1218 (7560)	559.21	0.60 (Table 7)

The machine-only option is the cheapest option, but generates the worst output quality. On the other hand, the human-only option is the most expensive and the most accurate. The hybrid approaches balance these two extreme cases. When adding the most trivial human work (cropping whole labels), the cost increases, the required overall time is actually reduced and the quality improves. Cropping fields, requires detecting the different fields, increasing the time and cost while maintaining quality when compared to the label-cropping hybrid approach.

In this cost evaluation, we considered that workers are compensated. However, if the crowdsourcing task can be made entertaining (e.g., as an application that museum visitors could use while interacting with the items in displays, or as an online competition game), where volunteers would be willing to contribute their work, then the cost would be drastically reduced.

V. CONCLUSIONS

In this work, we demonstrated that a single workflow with cooperation of human- and machine-intelligent processes led to improved quality, while not sacrificing significant time, when compared to a machine-only workflow. Even though we did not explicitly compare to human-only workflows, related work [7] has shown that human-only workflows demand user training, are time intensive (require multiple users to perform the same task), quality is not perfect, and require solutions to deal with variations in human opinion, bias, and error.

Improvements in output quality were assessed for two workflows with machine and human processes that require minimal user training to generate segments with text from the image: whole labels and individual parsed fields. These workflows were compared to a typical machine-only workflow and an improved machine-only workflow that implicitly removes noise from the output. The quality of the hybrid workflow was at least 42% superior. A secondary future goal of collecting text region information, is to investigate the ability to train a machine-learning model to look for and find regions with this characteristic. During segmentation, OCRopus uses different heuristics to find region of text and eliminate images, but these heuristics assume publication type layout, and specimen images do not follow such a constrained format. We also experimented with other tools that can detect text on photographs. While those could detect text within an image with

texture, they failed on biological specimen images. Collecting training data for tools that make use of supervised machine-learning algorithms is time demanding, and a hybrid workflow as presented in this work can also facilitate the tuning of such machine-only workflows.

In addition to these main findings, we also provided detailed insights into the performance of OCropus and Tesseract. Because Tesseract was on average 25 times faster than OCropus while maintaining the quality of output, cropping the label accelerated the OCropus execution time, but decreased Tesseract execution performance. Similarly, when considering the crowdsourcing time, the hybrid approach resulted in time efficiency gains with OCropus, and time efficiency loss with Tesseract. Factors such as yellowed paper, underlined text, low contrast text and graphical elements that touch characters, had a higher negative impact in the character recognition rate than the resolution.

ACKNOWLEDGEMENT

This work is supported in part by the National Science Foundation (NSF) grants No. ACI-1535086, No. EF-1115210, DBI-1547229, and the AT&T Foundation. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF or the AT&T Foundation.

REFERENCES

- [1] "A Matter of Life and Death: Natural science collections: why keep them and why fund them?," 1st ed. UK: NatSCA, 2005.
- [2] A.C. Bentley, "Scientific Collections: Mission-Critical Infrastructure for Federal Service Agencies," Interagency Working Group on Scientific Collections (IW/GSC), 2009.
- [3] R.S. Beaman and N. Cellinese, "Mass digitization of scientific collections: New opportunities to transform the use of biological specimens and underwrite biodiversity science," *Zookeys*, pp. 7-17, 07/09. 2012.
- [4] A.H. Aríñ, "Approaches to estimating the universe of natural history collections data," *Biodiversity Informatics*; Vol 7, no 2 (2010) DOI - 10.17161/bi.v7i2.3991, 10/09. 2010.
- [5] V. Blagoderov, , I. Kitching, , L. Livermore, , T. Simonsen, , V. Smith. "No specimen left behind: industrial scale digitization of natural history collections," *ZooKeys*, 209. 133–146, 2012.
- [6] P. Flemons and P. Berents, "Image based Digitisation of Entomology Collections: Leveraging volunteers to increase digitization capacity," No Specimen Left Behind: Mass Digitization of Natural History Collections. *ZooKeys*, vol. 209, pp. 203-217, 2012.
- [7] A. Matsunaga, A. Mast, J.A.B. Fortes, "Workforce-efficient consensus in crowdsourced transcription of biocollections information," *Future Generation Computer Systems*, Vol 56, March 2016, 526-536, ISSN 0167-739X, <http://dx.doi.org/10.1016/j.future.2015.07.004>.
- [8] Notes From Nature. Available: <http://notesfromnature.org>. [Accessed: 23- May- 2016].
- [9] D. Wagner, "Store First & Ask Questions Later," [allanalytics.com](http://www.allanalytics.com), 2014. Available: http://www.allanalytics.com/author.asp?doc_id=275286. [Accessed: 23- May- 2016].
- [10] "Augmenting OCR," iDigBio Wiki, 2014. Available: https://www.idigbio.org/wiki/index.php/Augmenting_OCR. [Accessed: 22- May- 2016].
- [11] D. Lafferty and L. Landrum, "SALIX, the Semi-automatic Label Information Extraction system," 1st ed. Tempe: School of Life Sciences, Arizona State University, 2012.
- [12] A. Barber, D. Lafferty and L.R. Landrum, "The SALIX Method: A semi-automated workflow for herbarium specimen digitization," *Taxon*, vol. 62, pp. 581-590, 2013.
- [13] A. Ul-Hasan, S. Bukhari, and A. Dengel, "OCRoRACT: A Sequence Learning OCR System Trained on Isolated Characters," 12th IAPR International Workshop on Document Analysis Systems, 2016.
- [14] W.E. Moen, J. Huang, M. McCotter, A. Neill, and J. Best, "Extraction and Parsing of Herbarium Specimen Data: Exploring the Use of the Dublin Core Application Profile Framework," 2010. Available: <http://hdl.handle.net/2142/14920>.
- [15] S. Dhiman, A.J. Singh, "Tesseract Vs GOCR A Comparative Study)," International Journal of Recent Technology and Engineering (IJRTE); Vol 2, Issue 4, 2013.
- [16] T. Breuel, A. Ul-Hasan, M. Al Azawi, and F. Safait, "High-Performance OCR for Printed English and Fraktur using LSTM Networks," 12th International Conference on Document Analysis and Recognition, 2013.
- [17] T. Breuel, "ocropy", GitHub, 2010. Available: <https://github.com/tmbdev/ocropy>. [Accessed: 23- May- 2016].
- [18] R. Smith and Z. Podobny, "Tesseract", GitHub, 2007. Available: <https://github.com/tesseract-ocr/tesseract>. [Accessed: 23- May- 2016].
- [19] R. Smith, "An Overview of the Tesseract OCR Engine," in Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 02, pp. 629-633, 2007.
- [20] R. Minerich, "Record Linkage Algorithms in F# – Extensions to Jaro-Winkler Distance (Part 3) « Inviting Epiphany," richardminerich.com, 2011. Available: <http://richardminerich.com/2011/09/record-linkage-algorithms-in-f-extensions-to-jaro-winkler-distance-part-3/>. [Accessed: 05- May- 2016].
- [21] "Optical character recognition," Wikipedia, 2016. Available: https://en.wikipedia.org/wiki/Optical_character_recognition. [Accessed: 04- May- 2016].
- [22] "Collaborative Data Extraction Scripts," Available: https://github.com/acislab/HuMaIN_Collaborative_Data_Extraction. [Accessed: 19- August- 2016].
- [23] J. Turk and M. Stephens, "jellyfish 0.5.3," pypi.python.org, 2016. Available: <https://pypi.python.org/pypi/jellyfish>. [Accessed: 04- May- 2016].
- [24] "Jaro-Winkler distance", Wikipedia, 2016. Available: https://en.wikipedia.org/wiki/Jaro-Winkler_distance. [Accessed: 04- May- 2016].
- [25] "Darwin Core", rs.tdwg.org, 2015. Available: <http://rs.tdwg.org/dwc/>. [Accessed: 24- May- 2016].
- [26] "Darwin Core Terms: A quick reference guide," rs.tdwg.org, 2015. Available: <http://rs.tdwg.org/dwc/terms/>. [Accessed: 24- May- 2016].
- [27] "label-data", <https://github.com/idigbio-aocr/label-data>. [Accessed: 24- May- 2016].
- [28] J. Leskovec, A. Rajaraman, and J. D. Ullman, "Mining of Massive Datasets", 2nd. ed., 2014, p. 74.
- [29] L. Lafferty, "SALIX 2," 2013. Available: <https://www.idigbio.org/content/idigbio-hackathon-salix-2>. [Accessed: 24- May- 2016].
- [30] B. Heidorn. "LABELX. Label Annotation Through Biodiversity Enhanced Learning," 2014. Available: <http://github.com/BryanHeidorn/LABELX>. [Accessed: 24- May- 2016].
- [31] W. Gomaa and A. Fahmy, "A survey of text similarity approaches. International Journal of Computer Applications," 2013, 68(13) doi:<http://dx.doi.org/10.5120/11638-7118>.
- [32] "HuMain: Human and Machine Intelligent Network," Advanced Computing and Information Systems (ACIS) laboratory. Available: <http://humain.acis.ufl.edu>. [Accessed: 24- May- 2016].
- [33] G. Fairchild, "pyxDamerauLevenshtein," GitHub, 2015. Available: <https://github.com/gfairchild/pyxDamerauLevenshtein>. [Accessed: 04- May- 2016].
- [34] "Symbiota Introduction," Available: <http://symbiota.org/docs/>. [Accessed: 24- May- 2016].
- [35] Barroso, L. A., Clidaras, J., & Hölzle, U., "The datacenter as a computer: An introduction to the design of warehouse-scale machines". Synthesis lectures on computer architecture, 8(3), pp. 1-154, 2013.

Generating Knowledge Networks from Phenotypic Descriptions

Fagner Leal Pantoja, Patrícia Cavoto, Julio Cesar dos Reis, André Santanchè

Institute of Computing

University of Campinas

Campinas, São Paulo, Brazil

{fagner.pantoja, julio.dosreis, santanche}@ic.unicamp.br

patricia.cavoto@gmail.com

Abstract—Several computing systems rely on information about living beings, such as *Identification Keys* – artifacts created by biologists to identify specimens following a flow of questions about their observable characters (phenotype). These questions are described in a free-text format, *e.g.*, “big and black eye”. Free-texts hamper the automatic information interpretation by machines, limiting their ability to perform search and comparison of terms, as well as integration tasks. This paper proposes a method to extract phenotypic information from natural language texts from biology legacy information systems, transforming them in an *Entity-Quality* formalism – a format to represent each phenotype character (*Entity*) and its state (*Quality*). Our approach aligns automatically recognized *Entities* and *Qualities* with domain concepts described in ontologies. It adopts existing Natural Language Processing techniques, adding an extra original step, which exploits intrinsic characteristics of phenotypic descriptions and of the organizational structure of *Identification Keys*. The approach was validated over the *FishBase* data. We conducted extensive experiments based on a manually annotated Gold Standard set to assess the precision and applicability of the proposed extraction method. The obtained results reveal the feasibility of our technique, its benefits and possibilities of scientific studies using the extracted knowledge network.

I. INTRODUCTION

Within the large set of knowledge bases containing information about living beings, phenotype descriptions play a key role, denoting the visible properties of organisms. These descriptions are written in a textual format and are mainly composed by morphological characters, *e.g.*, “eye”, and related qualifiers, *e.g.*, “big”.

Many problems arise when descriptions are written in a free-text format, such as the possibility of writing the same description in different ways. For instance, “*median fin skeleton*” can be written as “*unpaired fin skeleton*” and “*axial fin skeleton*”. It makes difficult the fully semantic interpretation of data by computers and limits their capacity of supporting accurate analyses over the information. The challenge in this scenario is how to distinguish, as automatically as possible, the characters and their states from free-text descriptions.

In this paper, we propose a method to detect the phenotype descriptions expressed in an *Identification Key* (IK), which is a decision tree to identify a specimen based on observed characters [1]. Our proposal transforms the recognized elements into a semantic-based representation aligned to the *Entity-Quality*

(EQ) approach [2]. In an EQ statement, the *Entity* refers to the morphological character (*e.g.*, “eye”) and the *Quality* stands for a qualifier (*e.g.*, “big”) that specifies a given state of the *Entity*.

This investigation defines a two-step method. The first step analyses a sentence using a Natural Language Processing (NLP) technique that produces a *Dependency Tree*, establishing dependency relations between the sentence terms. It extracts EQ elements computing matches between ontology concepts and terms of the tree. We assume that the relations among terms in the *Dependency Tree* have latent *Entity-Quality* statements. They reflect the biologists approach to write phenotype descriptions: a term (or a set of terms) representing a given *Entity* has specific kinds of dependency with a term representing its *Quality*. The second step takes advantage of the way that biologists relate and structure the phenotype descriptions. This step explores the correlations between sentences inside the IK. The identified *Entities* and *Qualities* are connected to domain ontologies to make their semantic explicit.

We conducted an experimental evaluation using data from *FishBase* to validate the proposed method. *FishBase*¹ is a fish knowledge base containing information used by researchers, fishery managers and zoologists. We show how recognized EQs can link descriptions of several species to produce a knowledge network and how this network can be explored for data analysis. The results indicate the adequacy and potentialities of our approach.

The remainder of this article is organized as follows: Section II formulates the research scenario and problem. Section III discusses the related work. Section IV describes the proposed method for the extraction and semantic linking of phenotype EQs. Section V reports on our experimental evaluation and shows potential applications of the generated knowledge network. Finally, Section VI draws conclusions and future work.

II. RESEARCH SCENARIO AND PROBLEM DEFINITION

Among several types of data managed by FishBase, Identification Keys (IKs) consist in artifacts created by biologists to

¹<http://www.fishbase.org>

Couplet	Character	Next	Prev	Link
1 a	One continuous dorsal fin.	2	(1)	
1 b	2 dorsal fins or dorsal fins clearly separated into 3 parts.	4	(1)	
2 a	Spines present in dorsals, sometimes feeble, pelvics present.	3	(1)	
2 b	No spines in dorsal or anal fin; eel-like; pelvics absent.	-	(1)	Apodocreadia, Creediidae
3 a	Mouth extending beyond eye, with elongate maxilla; caudal fin rounded to subtruncate.	-	(2)	Opistognathus, Opistognathidae
3 b	Mouth reaching eye, lower jaw projecting; caudal fin pointed; first 2-3 dorsal rays filamentous, free.	-	(2)	Trichonotidae

Fig. 1: Fragment of *Identification Key* to the *Teleostean families* from East Africa (sub-order *Trachinoidei*). Source: <http://www.fishbase.org/keys/description.php?keycode=799>.

identify species or any other taxonomic group (called taxon) of an observed specimen [1]. An IK denotes a structured set of phenotype descriptions of organisms. To identify a living being using an IK, users might navigate through a series of multiple choice questions about the specimen characteristics. According to the picked answers, the path leads to the respective taxon. Currently, *FishBase* has 1,668 IKs of fishes containing 25,542 phenotype description sentences.

As an example of IK usage, Figure 1 presents an IK to identify the *Teleostean* families, from East Africa (sub-order *Trachinoidei*). The identification process begins with question 1, which has the pair of options 1a and 1b, with their descriptive texts in the *Character* column. According to the picked answer, the user might navigate to question either 2 or 4, indicated in the *Next* column. Each descriptive text inside the *Character* column is called *Key Question* (KQ). This process is repeated until the biologist reaches a row that does not lead to another question. At this stage, the specimen is identified and its respective taxon appears at the *Link* column.

IKs and other data of *FishBase* are stored in a set of relational tables. Handling all these data manually is a huge challenge for scientists, who face difficulties to analyse some scenarios involving the network of relations (links) among taxa and their characteristics. The overwhelming amount of phenotype descriptions is in free-text format. This format is more flexible and easier to produce, having advantages in the narrative structure and providing better expressiveness. However, this free-text format is inappropriate for some computational tasks, mainly when it involves the interpretation and comparison of the content by machines. It hampers tasks involving information retrieval and integration with other sources, since the description components are “locked” within the text.

Therefore, it is necessary to develop methods that can automate the identification, extraction, and integration of phenotype information hidden in descriptive texts [3]. This research faces the problem through a method that automatically recognizes *Entities* and *Qualities* inside phenotype description sentences and link them to other data managed by *FishBase* (e.g., species, genus, country). It produces a knowledge network, making possible:

- **Reuse of EQs:** If EQs are duly unified in a semantic level, it is possible to identify which IKs refer to the same EQs, making explicit the network among IKs and

EQs.

- **No need of previous knowledge:** In *FishBase*, IKs are segmented according to the taxa that they identify. Therefore, users must know beforehand the specimen’s taxon to pick a correct IK. This process is laborious and error-prone; In addition, it limits the use of the system only to expert biologists, who could not have previous clues about the specimen to be identified. An explicit and standard semantic representation might enable to correlate EQ elements of several IKs and combine them in a unified identification tree.
- **Relation between taxa and keys:** Unified and semantic-enriched descriptions will enable to perform analyses to understand facts including: (i) which IKs identify similar taxonomic groups; (ii) which EQ elements are determinant to discriminate a taxon of a specimen; (iii) which EQ elements define a specific taxon.

III. RELATED WORK

There is a huge amount of biological data available in free-text format. As the process of producing biological data is expensive and complex, it is necessary to leverage the capability of automatically computing existing data. Thus, there is a challenge of migrating such vast amount of data into machine-interpretable formats, in order to produce semantically explicit knowledge.

These machine-interpretable data can be used by generic identification systems to improve their process and results. These systems implement different identification processes, such as: by descriptive characteristics, by pictures, by morphological measures, etc. Besides the generic identification systems, some information systems specialized in specific kinds of organisms may also offer support to build and publish IKs. For example, *FishBase* for fishes and *Bird Id* (<http://www.birdid.co.uk>) for birds. In the *FishBase* case, the identification process can be conducted in distinct ways, such as by images, by ecosystems, through descriptive characteristics, etc.

Several systems allow people to digitally create and publish Identification Keys for organisms [4], for example, *Intkey*, *IdentifyIt*, *Linnaeus II*, *Lucid* [5], *MEKA*, *NaviKey*, *PollyClave*, *XID*, *xPer* [6], *ActKey*, *eFloras*, *SLIKS*, and *KeyToNature* [7]. Technical reviews of some of these tools can be found in Dallwitz *et al.* [8].

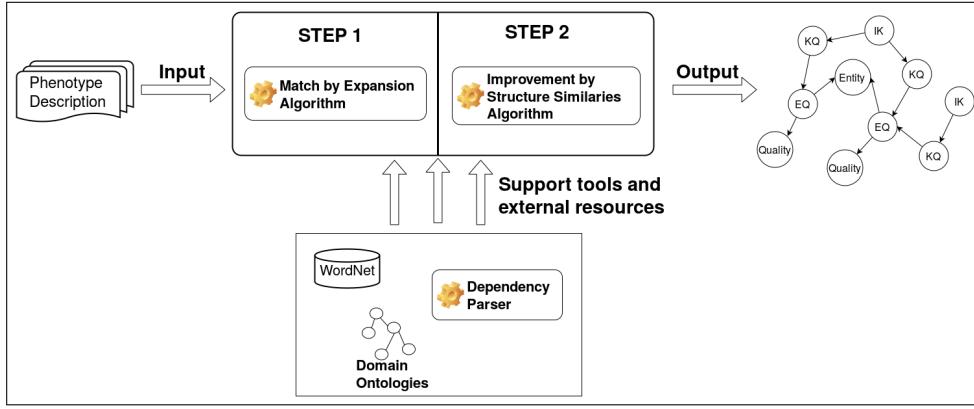


Fig. 2: General view of the proposed approach.

Farnsworth *et al.* [4] give an overview of technical innovations and trends in the area and highlight the importance of ontologies and semantics. They show that there is still space for improvements on the usage of these data, concerning data analysis and the correlation of phenotypes across different taxa and systems.

Existing investigations consider the use of phenotype descriptions in a machine-interpretable format. *Phenoscape*² addresses this issue adopting the *Entity-Quality* (EQ) approach to describe phenotypes and developing a scalable infrastructure that enables linking phenotypes across different fields of biology by the semantic similarity of their descriptions.

Concerning how to make explicit the semantics of biological data, Dahdul *et al.* [9] investigated techniques for transforming descriptive biology texts into a format that enables large-scale computation. Based on a previous study, they claim that large-scale computation can benefit from annotating characters with ontology terms. Therefore, they advocate the need of efficient methods to automatically extract and annotate phenotypes from descriptions and consider that NLP tools can be used in the process.

Related work concerning phenotype extraction are mostly concentrated in: interactions among genes, proteins, drugs, and diseases. This can be seen in Ciaramita *et al.* [10], Song *et al.* [11], Pyysalo and Ananiadou [1], Ramakrishnan *et al.* [12], and Fundel *et al.* [13]. Although the domains are similar to our work, we exploit specific characteristics of organisms morphological descriptions to improve the results of our extraction.

Cui [14] presents a method to extract phenotypes that describe leaves, fruits, and nuts of plants. He uses two key techniques: (a) an unsupervised learning algorithm to annotate descriptions at the sentence level, to build a lexicon; (b) the learned lexicon, enhanced by a human user, feeds a parser that recognizes biological characters in descriptive sentences and annotates them. Our work differs since it does not require human intervention during the process, in such a way that a non-expert can use the system.

²http://phenoscape.org/wiki/Main_Page

Alnazzawi *et al.* [3] compare several statistical learning methods against a curated corpus made by experts, called *PhenoCHF*. This corpus contains annotations about phenotypic information related to Congestive Heart Failure (CHF). One of their objectives is to demonstrate how the well-known methods perform better when a curated corpus is available. However, the creation of a corpus is a hard and expensive task. Our approach was developed to serve in contexts in which such corpus are unavailable.

IV. EXTRACTION OF ENTITY-QUALITY TERMS

This section details our approach to recognize and to make explicit *Entity-Quality* (EQ) elements, which are part of textual descriptions inside semi-structured Identification Keys (IKs). The method involves mapping them to a more formal representation with explicit semantics, based on domain ontologies. The approach departs from natural language text sentences (phenotype descriptions) and produces a graph representation of the recognized EQs. Figure 2 shows the general view of our approach, which encompasses two steps:

- **Step 1:** recognizes EQ elements through an algorithm that analyses the text of the sentence;
- **Step 2:** improves the results of Step 1 recognizing more EQ elements through an algorithm that analyses the relations of sentences according to the structure of the IK.

Both steps rely on external tools and resources throughout the process. This proposal is founded on two assumptions that synthesize the principles behind our method:

Assumption 1: The typical way in which a phenotypic description is written can guide the extraction of EQ elements.

Assumption 2: The way in which a set of phenotype descriptions is organized and structured holds implicit relations that can be exploited to improve the extraction of EQ elements.

Steps 1 and 2 implement algorithms based on Assumptions 1 and 2, respectively (*cf.* Sections IV-A and IV-B). We further define a notation to be used throughout this chapter, which will support the explanation of the method.

$E[e_x]$	= an Entity
$Q[q_y]$	= a Quality
$EQ[e_x, q_1, q_2, \dots, q_n]$	= an Entity-Quality
$S[s_x]$	= sentence in free-text format
$V[v_1, v_2, \dots, v_n]$	= vertexes of a Dependency Tree

A. Step 1: Exploiting the Writing Characteristics of a Phenotypic Description

Following Assumption 1, in order to guide the extraction task, this step exploits the typical approach followed by biologists to write phenotypic descriptions. This principle was previously exploited by other authors like Cui [14], who listed out some writing characteristics observed in Biology description texts:

- 1) Generally, morphological descriptions are constituted by two elements: Characters and Character States (C/CS);
- 2) Omission of Function Words – it is usual the omission of words that do not carry relevant meaning, such as articles and auxiliary verbs (*e.g.*, a, an, the, is, are);
- 3) Characters are often not explicitly stated in the descriptions. For example, in the sentence “*Black and big eyes*”, the characters color and size are not explicitly stated.

To deal with the first item, we have chose to work with Dependency Trees in order to reveal relations between sentence terms reflecting C/CS relations. Dependency Trees are produced by a Dependency Parser, which transforms the sentence in a tree of relationships between words, where each node represents a word and each edge denotes a grammatical dependency. The dependencies are all binary relations [15]. In this work, we use the *Stanford Typed Dependencies Parser*³ (STDP), a Dependency Parser implementation which belongs to the *Stanford Core NLP* toolkit. Figure 3 shows two examples of Dependency Trees generated by the parser.

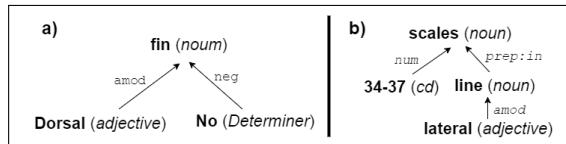


Fig. 3: Dependency trees of the sentences (a) $S[\text{No dorsal fin}]$ and (b) $S[34\text{-}37 \text{ scales in lateral line}]$.

STDP contains approximately 50 grammatical dependencies [16]. We selected a subset which reflects the written characteristics to be analized. For example, *amod* is a dependency that has an adjective qualifying a noun. In the Dependency Tree, Function Words are represented as edges (*e.g.*, the word “*in*” in Figure 3.b) instead of vertexes. Our match algorithm does not consider the edge labels. Therefore, these Function Words are ignored, which is conform the second writing characteristic of phenotype descriptions, observed by Cui [14] since it does not carry relevant meaning.

In order to obtain a semantic description of EQs, the Dependency Relations are matched with domain ontologies.

³<http://nlp.stanford.edu/software/stanford-dependencies.shtml>

We used ontologies widely adopted by the community: (1) *Teleost Anatomy Ontology* (TAO) [17] – an ontology that formalizes the knowledge about teleostean fishes anatomy; (2) *Phenotypic Quality Ontology* (PATO) – an ontology that defines *Qualities* to be related to *Entities* and their respective values.

We applied a recursive match algorithm that performs a search over a domain ontology (TAO or PATO: *Entity* or *Quality*, respectively). The match algorithm returns the concept in the ontology that has the highest similarity with a given subgraph of the input Dependency Tree. Similarity refers to which degree ($similarity \in [0, 1]$) an existing ontology concept is similar to the terms of the given subgraph. Firstly, the algorithm discovers the *Entities* present in the Dependency Tree. Then, it discovers *Qualities* related to the *Entities* already recognized. Figure 4 presents two elements returned by the match algorithm. The subgraph containing $V[\text{fin, dorsal}]$ is matched with $E[\text{Dorsal fin}]$ (a concept of TAO), while the subgraph containing $V[\text{no}]$ is matched with $Q[\text{absent}]$ (a concept of PATO).

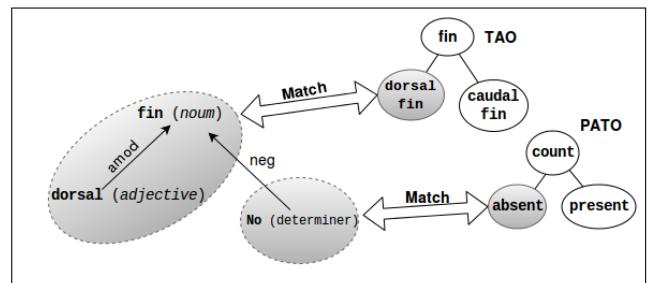


Fig. 4: *Entity* and *Quality* recognized in the sentence $S[\text{No dorsal fin}]$.

Figure 5 illustrates the execution of the recursive match algorithm looking for an *Entity*. At each iteration, the algorithm expands the subgraph adding vertexes connected to the current subgraph (neighbors). After recursively traversing all the Dependency Tree, it looks for the most similar concept $E[\text{dorsal fin}]$.

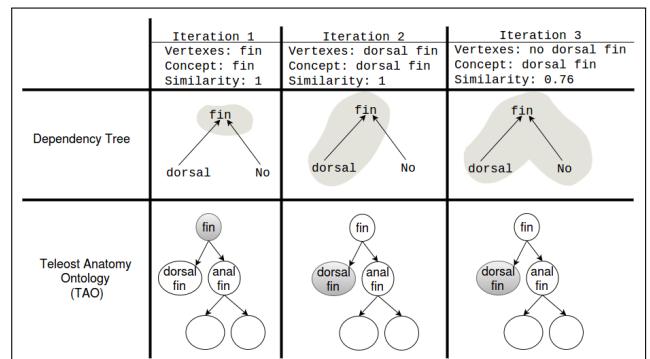


Fig. 5: Match by expansion algorithm over the sentence $S[\text{No dorsal fin}]$.

Figure 6 presents the result of Step 1: a graph where each

Key Question is connected to the respective recognized *Entities* and *Qualities*, e.g., the nodes $E[\text{dorsal fin}]$ and $Q[\text{absent}]$.

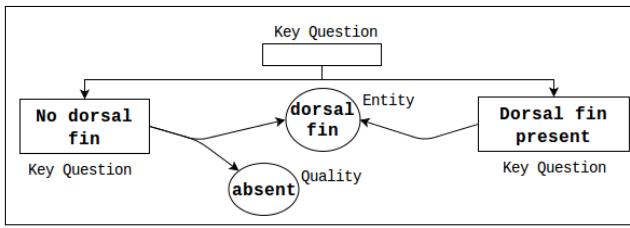


Fig. 6: Step 1 output.

It is possible to observe in Figure 6 that the method fails at recognizing $Q[\text{present}]$ in the sentence $S[\text{Dorsal fin present}]$. This failure is due to the violation of the English language rules in the sentence formulation. There are other cases where Step 1 fails, then the following Step 2 aims to treat these cases.

B. Step 2: Exploring the Structure of Identification Keys

This step explores the structure of IKs to enrich the output graph from Step 1. We assume that the correlation between distinct descriptions might be useful in the extraction of additional EQs. Such correlation is an intrinsic characteristic of IKs, as a result of their organizational structure. This step is based on the previously mentioned Assumption 2: **The way in which a set of phenotype descriptions is organized and structured holds implicit relations that can be exploited to improve the extraction of EQ statements.**

We believe that the principles behind this work could be generalized to other fields in the future. An organizational structure, as we exploit in the IKs, could also be the sessions of a technical report, the structure of legal documents with juridical rules, the layout of a Web site, etc. Wong *et al.* [18] indicate that such noncontent cues may be used to support information extraction tasks. This perspective opens a future wider application scenario for our technique.

IKs are structured in a tree format, in which the alternatives of a given KQ are its sibling nodes containing complementary alternative sentences. This structure offers clues about its content, from which we consider the following characteristics:

- Alternatives of a KQ frequently refer to the same *Entities*. In our previous example, both sibling sentences $S_1[\text{No dorsal fin}]$ and $S_2[\text{Dorsal fin present}]$ refer to the same anatomical character $E[\text{dorsal fin}]$;
- Alternatives of a KQ are frequently complementary, in the sense that they assign complementary states to the described *Entity*. In the same previous example, the *Qualities* $Q_1[\text{absent}]$ and $Q_2[\text{present}]$, assigned to the *Entity* $E[\text{dorsal fin}]$, are opposites, encompassing its possible state values.

In summary, we assume that if an EQ pair is identified in a KQ, it is very likely that the sibling KQs must refer to the same *Entity*, but potentially using complementary *Quality* terms to modify the *Entities*. The challenge here is to verify if the sibling nodes hold this property.

Therefore, we developed an algorithm that measures the similarity between two sentence pieces. It is based on the general principle of Paraphrase Recognition, which is a process to judge if two different sentences convey the same aspect or the same information. Androutsopoulos and Malakasiotis [19] present a survey regarding Paraphrase Recognition techniques. There are techniques that exploit the dependency tree to measure the similarity between the sentences. In general, they assume that if there is a value above a given threshold, the involved sentences are considered paraphrases.

Usually, Paraphrases Recognition algorithms compare the whole trees [19]. We have adapted the principle of Paraphrases Recognition to the problem of recognizing complementary sentences in an IK.

Step 2 acts in cases where Step 1 was successful in one sentence, but failed in recognizing EQ statements in its siblings. It determines if these sentences have complementary *Qualities* for the same *Entities*. It measures the similarity between the subtrees comparing each edge inside them. We aim to verify if they refer to the same *Entity* with complementary *Qualities*, based on a settled threshold.

Figure 7 illustrates the Step 2 input elements. The algorithm receives a pair of *Key Questions*: KQ_{main} and KQ_{sibling} . Inside each KQ node, there are the Dependency Trees of the sentences. The KQ_{main} has a link to an EQ pair E_1Q_1 and the KQ_{sibling} has a link to the same E_1 , but it lacks the Q_2 (dashed element), to be recognized by the algorithm.

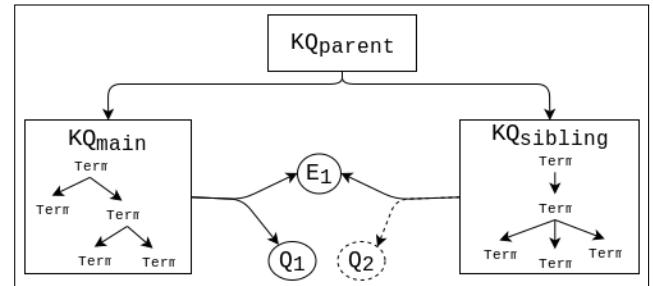


Fig. 7: A generic example of Step 2 input.

The Step 2 algorithm iterates over each EQ pair extracted from the KQ_{main} (E_1 and Q_1 in Figure 7). For each EQ pair, the algorithm gets the corresponding subtrees $Entity_{\text{main_subtree}}$ and $Quality_{\text{main_subtree}}$ that contain the terms that are part of the E_1 and Q_1 , respectively. Figure 8 exemplifies these subtrees, highlighting $Entity_{\text{main_subtree}}$ and $Quality_{\text{main_subtree}}$ inside the DT_{main} .

The algorithm gets the $Edge_{\text{main_2}}$, which links the $Entity_{\text{main_subtree}}$ to the $Quality_{\text{main_subtree}}$. Then, the algorithm fetches the subtrees extracted from sibling KQs (KQ_{sibling} , in this case) and compares the original edge with all edges related to the subtree $Entity_{\text{sibling_subtree}}$ (which represents the same entity of $Entity_{\text{main_subtree}}$): edges $\langle Edge_{\text{sibling_1}}, Edge_{\text{sibling_2}}, Edge_{\text{sibling_3}}, Edge_{\text{sibling_4}} \rangle$. This comparison looks for an edge that connects the $Entity_{\text{sibling_subtree}}$ to the complementary Q_2 . The similarity

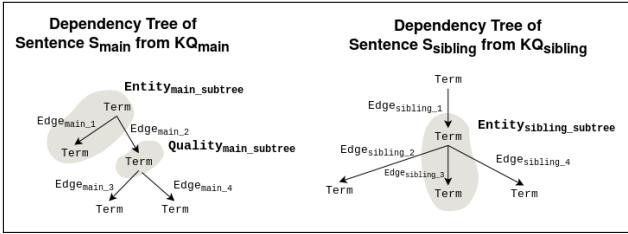


Fig. 8: Generic example of dependency trees of two sentences. *Entity* and *Qualities* recognized in the previous Step 1 are highlighted.

computation between the edges takes into account the following parameters:

- Directions of the dependency relations $edge_{sibling_n}$ and $edge_{main_2}$;
- Grammatical class of Q_1 and Q_2 ;
- Types of the dependency relations of $edge_{sibling_eq}$ and $edge_{main_eq}$;
- Antonymy between Q_1 and Q_2 (the algorithm explores the *WordNet* lexical database [20] to check if two words are antonyms).

These parameters represent to which extent one edge is similar to another. To calculate the degree of similarity, each parameter contributes with a pre-defined value: $v_a = 0.25$; $v_b = 0.50$; $v_c = 0.75$; $v_d = 1$.

We have chose these parameters and estimated their corresponding values based on empirical observations regarding their relevance in Dependency Tree elements (edges and vertexes) concerning phenotype description sentences. For example, we noted that a pair of edges having the same direction is important, but it is less important than the fact that the *Qualities* have antonyms terms since the algorithm is looking for opposite *Qualities*. These parameters and their values can be adapted to the execution of the algorithm in other scenarios.

The similarity between each pair of edges is calculated through a summation of those parameters. The edge with the highest similarity value is selected as the potential Q_2 , if it is equal or higher than a determined *threshold*. In the conducted experiments, we assigned the $threshold = 0.75$ to avoid retrieving edges with low similarity values. Afterward, the recursive match algorithm (the same used in Step 1) performs a search over the *PATO* ontology in order to confirm if the selected edge corresponds to a *Quality* concept.

The *threshold* value can be modified and it affects the behaviour of the algorithm. A high *threshold* value enables to recognize more *Qualities*, but it can increase the rate of false positives. On the other hand, a low value can decrease the number of recognized *Qualities*, but it increases the rate of correct elements. The values of each parameter and threshold have been empirically determined by experimental analyses.

Figure 9 shows the the Step 2 output for the KQs example. Compared to Figure 6, the $Q[\text{present}]$ was inserted as a new node in the graph as a result of the algorithm.

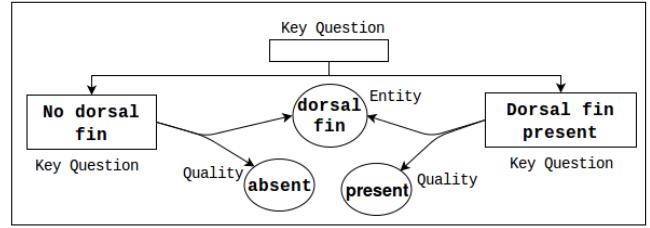


Fig. 9: Step 2 output.

V. EVALUATION AND APPLICATION EXPERIMENTS

This section reports experimental results of this investigation. We rely on the *FishBase* database to conduct the proposed assessments. Section V-A presents an evaluation to assess the viability of our extraction method. The objective is to investigate the effectiveness of the approach considering a gold standard dataset and traditional metrics.

The initial motivation for this research was to obtain a knowledge network correlating and integrating several elements of phenotype descriptions. Therefore, Section V-B presents experiments linking *FishBase* species data through the recognized EQs.

A. Accuracy Assessment

The quality of the recognition and extraction of elements in natural language texts, *i.e.*, entities or relations, can be evaluated by several mechanisms. The most common considers a standard evaluation set generated by either a group of specialists in the domain, or an organizing committee of a competition. A standard evaluation set contains fragments of texts highlighting the elements that are supposed to be recognized. Such kind of evaluation is suitable when there is a mature developed community acting in the area of interest.

However, there is still no standard evaluation set for morphological descriptions, in the context that we are working, *i.e.*, *Entity* and *Quality* linked in an EQ pair. Therefore, this investigation involved the creation of an evaluation dataset to assess the performance of our method. This dataset has the original sentence descriptions where the EQ elements are annotated. A set of 100 KQs have been manually annotated, from the total of 25,542 KQ from *FishBase*.

Figure 10 shows four examples of sentences in our evaluation dataset. The words in bold compose *Entities*, and words in italic compose *Qualities*, while the boxes represent EQ pairs.

1)	Lips <i>not fringed</i> ; mouth <i>horizontal</i> .
2)	No <i>dark longitudinal stripes</i> on head and body .
3)	Total vertebrae <i>119 to 132</i> .
4)	Scattered <i>breast melanophores</i> . One <i>large spot</i> centered at the base of the caudal fin .

Fig. 10: Examples of sentences within our Gold Standard.

Several criteria were explored to create the Gold Standard. First, we considered only Simple EQs, *i.e.*, those composed

strictly by one *Entity* and one *Quality*, such as in the second sentence in Figure 10: *E[stripes]Q[no]*, *E[stripes]Q[dark]* and *E[stripes]Q[longitudinal]*. To save space, we group them as follows: *E[stripes]Q[no]Q[dark]Q[longitudinal]*.

We ignored complex EQs, *i.e.*, those composed by complex *Qualities*, which recursively contain *Qualities* linked to other *Entities*. For example, Sentence 4 in Figure 10 has a complex EQ formed by *E[spot] Q[centered at the base of] E[caudal fin]*. This kind of phenotype construction requires further efforts and expertise to produce annotation. In particular, complex EQs are not treated by our approach and to avoid misinterpretations in the numerical evaluation, they are not computed.

We applied our method to each annotated KQ. We compared the EQs recognized by our method with the annotations of the Gold Standard. The comparison considers four indicators:

- **True Positive (TP):** elements correctly identified. For Example: our method identified in Sentence 1 the following EQs: *E[lips]Q[not fringed]*; *E[mouth]Q[horizontal]*. These elements were actually annotated in Sentence 1 of the Gold Standard;
- **False Positive (FP):** An expression recognized by the method as a phenotype, which does not appear as such in the Gold Standard. Example: in Sentence 3, our approach recognized *E[vertebrae]Q[132]*, which is a *Quality* that slightly differs from the expected one;
- **False Negative (FN):** those phenotypes annotated in the Gold standard that were not detected by the method. Example: *E[breast melanophores]Q[Scattered]* should be identified in Sentence 4 and we failed in recognizing it.

The computation of TP, FP and FN allows calculating the following traditional measures:

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

$$F\text{-measure} = \frac{2 * Precision * Recall}{Precision + Recall} \quad (3)$$

Table I presents the obtained results of Precision, Recall and F-measure. The column “EQ pair” compute the recognition of complete *Entity-Quality* pairs and the column “Entity” computes the recognition of *Entities* alone without related *Qualities*.

TABLE I: Results concerning Perfect Matches.

Measures	Elements	EQ pair	Entity
Recall		0,39	0,69
Precision		0,75	0,85
F-measure		0,51	0,76

The results are influenced by EQs partially recognized. These cases might not be considered a totally wrong result, then we added a more flexible partial match count:

- **Partial Matches (PM):** The cases where the recognized element contains only a part of the expected one. Example: in sentence 3, the *E[vertebrae]Q[132]* was recognized, but the *Quality* is not complete.

As a consequence, the measures are adapted as follows:

$$Total\ Precision = Precision + \frac{PM}{TP + PM + FP} \quad (4)$$

$$Partial\ Recall = Recall + \frac{PM}{TP + PM + FN} \quad (5)$$

$$F\text{-measure} = \frac{2 * Total\ Precision * Total\ Recall}{Total\ Precision + Total\ Recall} \quad (6)$$

Tables II presents the obtained results considering partial matches. As expected, in an overall analysis, results reached with partial matches overcome the results of exact matches. We note that better results yield mostly by the Recall.

TABLE II: Total Results.

Measures	Elements	EQ pair	Entity
Total Recall		0,45	0,76
Total Precision		0,87	0,94
Total F-measure		0,59	0,84

The results are affected by many factors, among them: the range of terms in the domain covered by the ontology. In our case, the universe of *Quality* terms is more vast than those available in PATO.

A study comparing our results with related work is hampered by the unavailability of a Gold Standard set. However, it is possible to compare the proposals conceptually. Among the existing approaches, the most related is the *CharaParser* – part of the *Phenoscope* project – which has a good acceptance in the community. Our work presents more independence of human action in identifying the EQ elements, since *CharaParser* requires some steps of validation by the user over the extracted information, to feed the next steps.

Our approach demands further refinements to identify more EQ elements. Among them, we highlight the need of:

- extending the EQ formalism to handle complex EQs with compound *Entities* and *Qualities*;
- developing a method to perform an Entity Linking task to handle complex cases, *e.g.*, *S[first four dorsal spines prolonged, the second and third longest]*. This sentence requires identifying that the words *second* and *third* implicitly mention the *E[dorsal spine]*;

B. Knowledge Network Analysis

In this section, we present practical applications, which are possible due to the extraction of phenotypes. The objective is demonstrating the usefulness of explicitly recognizing EQs. The knowledge network was created by correlating the detected EQs with other information elements available in *FishBase*. In particular, we correlated EQ pairs with data concerning taxonomic groups of fishes. Afterwards, we generated different information visualizations/perspectives to evaluate the obtained correlations. We selected specific cases to highlight the relevance of considering EQ statements.

Figure 11 shows a graph model derived from *FishBase* (*FishGraph*), created by a previous work of Cavoto *et al.* [21]. It highlights the node types (*class*, *order*, *family*, *species*, *genus*, *country*, *key*, and *ecosystem*) and relationships among them. We have added new nodes to *FishGraph* – *keyQuestion*, *EQ*, *Entity*, and *Quality* – and linked them to the existing ones.

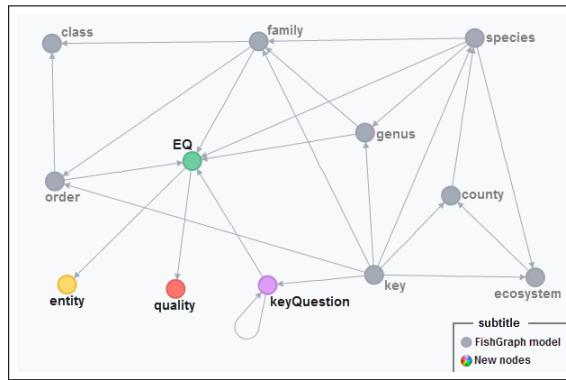


Fig. 11: New nodes added in the *FishGraph* database.

This updated *FishGraph* version models our knowledge network. It allows scientists to perform analyses making use of the new extracted information. We further present examples of possible applications to improve the system usage by the user and analyses to understand facts about living beings.

1) *No need of previous knowledge*: Currently, each IK is represented in *FishBase* as an independent tree, which hampers their usage, as one needs to know beforehand the main taxon (the root of the IK tree) to start the identification process.

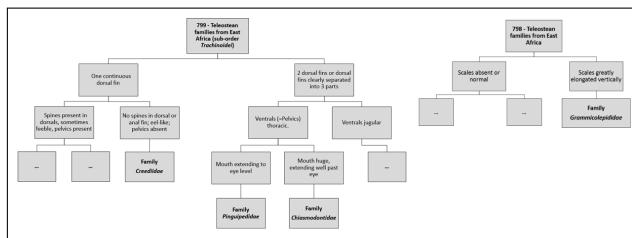


Fig. 12: Part of *Identification Keys*: 799 of Teleostean families from East Africa (sub-order Trachinoidei) and 798 of the Teleostean families from East Africa.

As an example, consider the IK “799 – Teleostean families from East Africa (sub-order Trachinoidei)” (cf. Figure 12).

The identification process using this key requires the following knowledge: the specimen belongs to the teleostean family, sub-order Trachinoidei and it is found in East Africa. The identification process is hampered if the user knows only part of the root – suppose family and geographic location – since *FishBase* has another 6 IKs of the teleostean family distinguished mainly by the sub-order, *e.g.*, IK 798 also in Figure 12. Even with all the required knowledge, it is necessary to follow the proposed path in the IK tree. All these particularities make the identification process only possible to specialists.

The new structure allows starting the identification process from any known characteristic. For instance, we can start the identification process using a known characteristic like *dorsal fin soft*, independently of any IK or other characteristic.

2) *Searching through incremental filtering*: The generated knowledge graph allows searching for specific taxons by applying an incremental filtering process.

Figure 13 shows an example of this incremental filter using *Entities* and *Qualities*, which leads to a family with three specific characteristics. Figure 13.a shows the initial filtered graph with 27 families of species that have the *E[dorsal fin]Q[soft]* (*Entities* and *Qualities* are collapsed in a single node, in order to simplify the view). Adding a second filter of the *E[anal fin]Q[soft]* (Figure 13.b) means to select those species with edges to both EQs. The number of families with both characteristics decreases to 8. A third filter of the *E[body scale]*, results in only 1 family that has the 3 characteristics: *Creediidae* (Figure 13.c).

3) *Relation of taxons and IKs*: One taxon is referred in many IKs in *FishBase* but, since they are independent, each IK has its own set of characteristics. When we analyse IKs referring to the same taxon, there are two possible cases: (i) keys share partially or totally the characteristics of a given taxon; (ii) keys that have complementary information about the taxon.

Our unified graph structure links distinct characteristics of the same taxonomic group, coming from many independent IKs, enriching and facilitating the identification process. Returning to the previous experiment, the *E[body scale]* is a characteristic that belongs to IK 324 but it does not belong to IK 799. Since they refer to the same taxonomic group, it is possible to combine them to achieve a more complete description of the taxons.

4) *Phenotypes distinguishing taxons*: Figure 14.a shows a fragment of the obtained knowledge network highlighting 3 classes of fishes and the EQ elements concerning the *tooth* structure. As can be seen, our approach enabled to unify the *Entities* and it is possible to verify that all 3 classes share the same EQ elements. However, if we drill down to the level of family, it is possible to verify which EQ elements distinguish the two families *Aulopiformes* and *Cetomimiformes* – the size of the tooth: the first one (Figure 14.b) is large and the second (Figure 14.c) is small.

5) *Sharing EQs through taxons*: We built a bipartite network consisting of two different types of nodes: species and

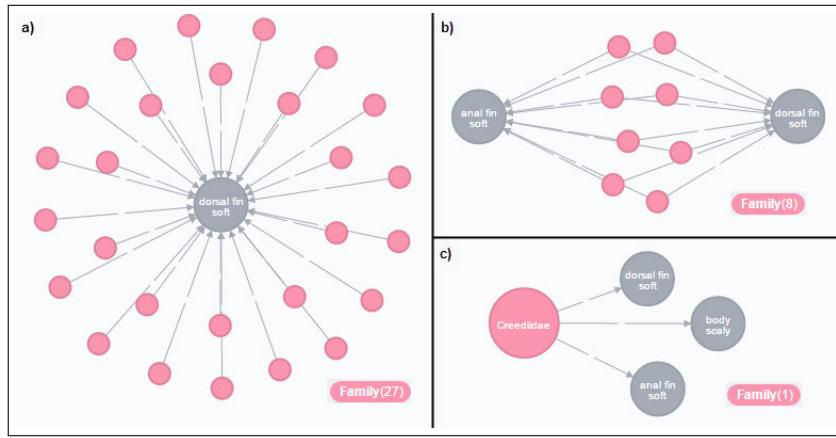


Fig. 13: Filtering families of species by EQ: a) dorsal fin soft; b) dorsal fin soft and anal fin soft; and c) dorsal fin soft, anal fin soft, and body scale.

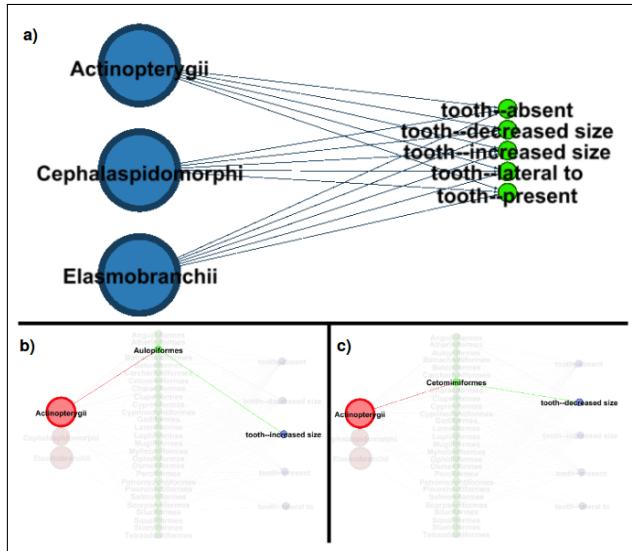


Fig. 14: (a) Relation between a set of EQ elements and classes. (b) EQ element determiner of Aulopiformes Family. (c) EQ element determiner of Cetomimiformes family.

EQ statements. In this network, each EQ element is linked to the species that has it.

Figure 15 shows a small portion of this network, in a synthetic view. Since several EQ elements are shared by a large number of species, the resulting bipartite network is too dense for direct visualization. While 29 species are on the left side, 6 EQ pair elements are on the right side. This network enables visualizing which EQ pairs are the most shared by the species.

In the visualization aspect, the size of the EQ nodes indicates the amount of linked species, e.g., the *E[melanophore spot]Q[low brightness]* is the biggest node, which means that it is an EQ pair present in many species.

Figure 16 shows a projection of the bipartite network. In this visualization, the nodes are EQs and they are connected

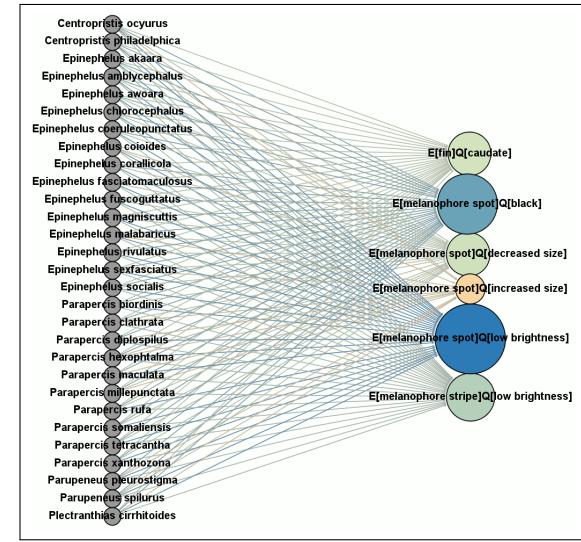


Fig. 15: Bipartite network of Species and EQs elements, showing some of the most present EQ in the species.

if they are present together at least one species. The link width is proportional to the amount of shares. The size of the nodes indicates the prevalence of the EQ elements in species.

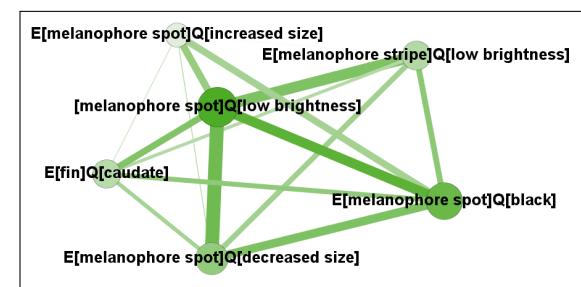


Fig. 16: Projection of the Bipartite network showing the most shared EQ elements by species.

This visualization allows to study which EQ elements frequently occur together. For example, the link width between the nodes *E[melanophore spot]Q[low brightness]* and *E[melanophore spot]Q[decreased size]* indicates that they are EQ elements present together in many species.

VI. CONCLUSION

Phenotype descriptions play a key role in biological knowledge bases, but most of the descriptions remain in a free-textual format, which affects machine interpretation and their applicability in network-driven analyses.

This paper proposed an original approach to recognize *Entities* and *Qualities* connecting them to concepts in ontologies to make their representation semantically interpretable by machines. Our key point, not addressed by related work found in literature, consists in our approach, which explored clues of non-textual information: from the writing characteristics of phenotype descriptions to their organizational structure.

The experimental evaluations revealed encouraging results regarding the assessment against a gold standard set. The experiments point out the contributions of each step to improve the results of the recognition process. The experiments using the EQ elements, extracted from free-text sentences applying our proposal, showed the advantages of bringing these descriptions to a common and formal language. It enables machines better consuming and interpreting the available descriptions.

Future work involves conducting further evaluations to measure and compare the efficiency concerning to other approaches. It also aims at addressing limitations of the recognition of EQ elements.

ACKNOWLEDGMENT

Work partially financed⁴ by CNPq (134205/2015-4), FAPESP (2014/14890-0), FAPESP/Cepid in Computational Engineering and Sciences (2013/08293-7), FAPESP Storage, Modeling and Analysis of Dynamical Systems for e-Science Applications project (2014/08285-7), the Microsoft Research FAPESP Virtual Institute (NavScales project), CNPq (MuZOO Project), CNPq (Descrições Complexas na Web), FAPESP-PRONEX (eScience project), INCT in Web Science, and individual grants from CNPq.

REFERENCES

- [1] S. Pyysalo and S. Ananiadou, “Anatomical entity mention recognition at literature scale,” *Bioinformatics*, vol. 30, no. 6, pp. 868–875, 2014.
- [2] A. Grand, R. V. Lebbe, and A. Santanche, “From Phenotypes to Trees of Life: A Metamodel-Driven Approach for the Integration of Taxonomy Models,” in *IEEE 10th International Conference on e-Science*, vol. 1, 2014, pp. 65–72.
- [3] N. Alnazzawi, P. Thompson, R. Batista-Navarro, and S. Ananiadou, “Using text mining techniques to extract phenotypic information from the phenochf corpus,” *BMC Medical Informatics and Decision Making*, vol. 15, no. Suppl 2, p. S3, 2015.
- [4] E. J. Farnsworth, M. Chu, W. J. Kress, A. K. Neill, J. H. Best, J. Pickering, R. D. Stevenson, G. W. Courtney, J. K. VanDyk, and A. M. Ellison, “Next-generation field guides,” *BioScience*, vol. 63, no. 11, pp. 891–899, 2013.
- [5] Lucid. (2016) Lucid phoenix. [Online]. Available: <http://www.lucidcentral.com/>
- [6] V. Ung, G. Dubus, R. Zaragüeta-Bagils, and R. Vignes-Lebbe, “Xper2: introducing e-taxonomy,” *Bioinformatics*, vol. 26, no. 5, pp. 703–704, 2010.
- [7] S. Martellos and P. Nimis, “Keytonature: teaching and learning biodiversity. dryades, the italian experience,” in *Proceedings of the IASK International Conference Teaching and Learning*, 2008, pp. 863–868.
- [8] M. J. Dallwitz, T. Paine, and E. Zurcher, (2000) Principles of interactive keys. [Online]. Available: <http://biodiversity.uno.edu/delta>
- [9] W. Dahdul, T. A. Dececcchi, N. Ibrahim, H. Lapp, and P. Mabee, “Moving the mountain: analysis of the effort required to transform comparative anatomy into computable anatomy,” *Database*, vol. 2015, p. bav040, 2015.
- [10] M. Ciaramita, A. Gangemi, E. Ratsch, J. Šaric, and I. Rojas, “Unsupervised learning of semantic relations between concepts of a molecular biology ontology,” *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pp. 659–664, 2005.
- [11] M. Song, H. Yu, and W.-S. Han, “Developing a hybrid dictionary-based bio-entity recognition technique,” *BMC Medical Informatics and Decision Making*, vol. 15, no. Suppl 1, p. S9, 2015.
- [12] C. Ramakrishnan, P. N. Mendes, S. Wang, and A. P. Sheth, “Unsupervised discovery of compound entities for relationship extraction,” in *Knowledge Engineering: Practice and Patterns*. Springer, 2008, pp. 146–155.
- [13] K. Fundel, R. Kuffner, and R. Zimmer, “RelEx - Relation extraction using dependency parse trees,” *Bioinformatics*, vol. 23, no. 3, pp. 365–371, 2007.
- [14] H. Cui, “Charaparser for fine-grained semantic annotation of organism morphological descriptions,” *Journal of the American Society for Information Science and Technology*, vol. 63, no. 4, pp. 738–754, 2012.
- [15] M.-C. De Marneffe and C. D. Manning, “Stanford typed dependencies manual,” URL http://nlp.stanford.edu/software/dependencies_manual.pdf, 2008.
- [16] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, “The Stanford CoreNLP natural language processing toolkit,” in *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2014, pp. 55–60. [Online]. Available: <http://www.aclweb.org/anthology/P/P14/P14-5010>
- [17] W. M. Dahdul, J. G. Lundberg, P. E. Midford, J. P. Balhoff, H. Lapp, T. J. Vision, M. A. Haendel, M. Westerfield, and P. M. Mabee, “The teleost anatomy ontology: anatomical representation for the genomics age,” *Systematic Biology*, vol. 59, no. 4, pp. 369–383, 2010.
- [18] T.-L. Wong, W. Lam, and T.-S. Wong, “An unsupervised framework for extracting and normalizing product attributes from multiple web sites,” in *Proceedings of the 31st annual international ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2008, pp. 35–42.
- [19] I. Androutsopoulos and P. Malakasiotis, “A survey of paraphrasing and textual entailment methods,” *Journal of Artificial Intelligence Research*, pp. 135–187, 2010.
- [20] G. A. Miller, “WordNet: a lexical database for English,” *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.
- [21] P. Cavoto, V. Cardoso, R. V. Lebbe, and A. Santanchè, “FishGraph: A Network-Driven Data Analysis,” in *11th IEEE International Conference on e-Science*, 2015.

⁴The opinions expressed in this work do not necessarily reflect those of the funding agencies

PRUNE: A Preserving Run Environment for Reproducible Scientific Computing

Peter Ivie
 University of Notre Dame
 pivie@nd.edu

Douglas Thain
 University of Notre Dame
 dthain@nd.edu

Abstract—Computing as a whole suffers from a crisis of reproducibility. Programs executed in one context are astonishingly hard to reproduce in another context, resulting in wasted effort by people and general distrust of results produced by computer. The root of the problem lies in the fact that every program has implicit dependencies on data and execution environment which are rarely understood by the end user. To address this problem, we present PRUNE, the Preserving Run Environment. In PRUNE, every task to be executed is wrapped in a functional interface and coupled with a strictly defined environment. The task is then executed by PRUNE rather than the user to ensure reproducibility. As a scientific workflow evolves in PRUNE, a growing but immutable tree of derived data is created. The provenance of every item in the system can be precisely described, facilitating sharing and modification between collaborating researchers, along with efficient management of limited storage space. We present the user interface and the initial prototype of PRUNE, and demonstrate its application in matching records and comparing surnames in U.S. Censuses.

I. INTRODUCTION

Reproducibility of results has become a major concern in scientific computing [1], [2], [3]. The challenge lies largely in the complexity of software and environments. A scientist's *productivity* can be reduced due to time spent getting shared software working, and the *integrity* of science becomes suspect due to the difficulty in verifying claims. A large part of the problem is **implicit dependencies**, such as configuration files, supporting libraries, scripting languages, an OS kernel, and suitable hardware [4]. The standard command line interface is not designed to encourage reproducibility.

One common solution is to use virtual machines (or containers) to store the full context available to an application. This can result in excessive overhead, so another common technique is to trace only dependencies that are actually used, such as with CDE [5], PTU [6], ReproZip [7], and Parrot [4]. The next step is to capture the evolution of the application over time, or the data flowing between multiple containers for large scale applications.

PRUNE is a computing system designed to support scientific reproducibility from the ground up.¹ PRUNE expects all data, software, and environment dependencies to be explicitly registered with the system and directly computes the tasks that make up an application, rather than recording the system

¹PRUNE was briefly introduced (along with a few other utilities) in a paper at iPres 2015 [8]. This is the first detailed presentation of the architecture, performance, and applications of the system.

calls or merely the end results of user executed commands. As various execution paths are explored, the user builds up a large graph of data, software, and environments used.

PRUNE bears some similarity to a distributed version control system like Git [9], except that it records a tree of continuous derivation information consisting of immutable files and computations, rather than a tree of periodic commits consisting of line changes in a directory of mainly textual files. This allows the end user to understand the precise provenance of any generated results. PRUNE is also similar to the Nectar [10] datacenter management system, except that it supports distributed de-duplication of storage and computation, rather than relying on a centralized system. This distributed provenance of computation facilitates flexible collaboration using export/import operations directly between individual scientists.

We have implemented a prototype of PRUNE (version 1) that exploits distributed execution, allowing it to scale to a large number of tasks. As a case study, we show how PRUNE is used to manage a data analysis workflow based on US censuses, developing a repository that preserves 34.5 terabytes of workflow data representing 3.9 million executions, 3.9 million files, and 2 different execution environments. To characterize the performance of this prototype, we measure the overhead of system operations in the context of this large repository. We show the wall clock time consumed is 101% compared to native execution (or 1% overhead).

II. A PRESERVE-FIRST STRATEGY

Preservation is often perceived as an activity undertaken after research has been completed. [11] But, by the time the results based on a scientific workflow are accepted for publication, the authors have moved on to other work, students may have graduated, or the environment in which the work was done has been changed, upgraded, or destroyed. The funding that supported the research may have expired, and so it is hard to justify any post-facto effort in preservation. Even when such an effort is made, the focus is often only on repeatability [12], and more work is needed to fill in gaps in the preserved form of the research [13]. This process is shown in Figure 1a.

In contrast, we advocate a **preserve-first strategy** for reproducible computational research as shown in Figure 1b. We argue that researchers should first (before any computation) preserve (at least locally) the components they wish to use. Automated execution based on the preserved components can

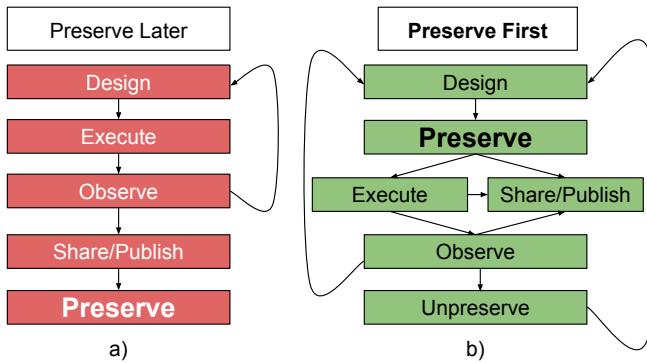


Fig. 1: Preserve-First or Preserve-Later? We propose a preserve-first strategy, where digital items are preserved before use to avoid ambiguity about results.

then ensure all necessary dependencies are included, otherwise the execution fails. Once the desired research results are obtained, it is then trivial to publish them with full provenance in a public repository. Then others can build upon the same work with a high probability of success.

Adopting this strategy requires additional user and computer overhead. But we believe with this approach, PRUNE moves towards greater structure and oversight such as with the adoption of: block-structured programming [14]; graph-structured Make files [15]; and rigorous version control [9].

III. OVERVIEW OF PRUNE

A. User's Perspective

An end user begins by creating their own private PRUNE repository, which may simply exist on their own laptop. The user describes a workflow which explicitly adds (into the repository) any input data and tasks that should be performed to derive some result. When the user submits this description, PRUNE detects portions of the workflow that are already in the repository, and records and then adds the remainder. Observing the results, the user may submit a revised workflow, expanding the graph in the repository. If space consumption becomes a problem, PRUNE will automatically delete derived results, because it retains the ability to re-create them on demand.

Other users or organizations may operate their own repositories. When a user has a result of interest to be shared, he/she asks PRUNE to export the appropriate meta-data into a portable package. The package can contain all the meta-data necessary to describe how the result was obtained, so that a receiving user can examine, re-execute, or build upon that result within their own repository. The most interesting results can be widely disseminated through a public repository.

B. Repository Structure

A PRUNE **repository** contains a graph of immutable objects describing the data and computational elements needed to execute a workflow. The following 4 basic objects constitute the nodes of the graph: Files, Tasks, Results, and Environments. Once a workflow has been described in terms of these objects,

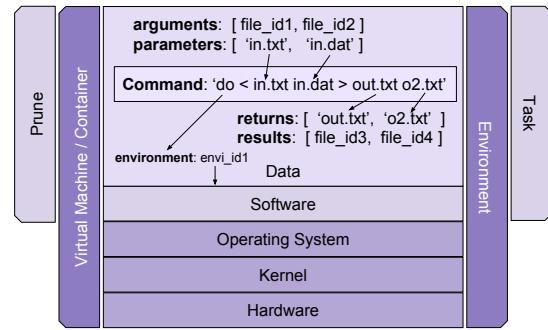


Fig. 2: Tasks and Environments: Data and software can be handled in a Task or Environment. The Environment describes down to the hardware layer.

the objects can be shared with collaborators or published as a complete and reproducible description of the workflow.

A File is an immutable string of bytes, identified by a hash of the content of the File. Any data that the user wishes to use must first exist as a File within a repository.

A Task is a program to be executed, represented as a brief JSON document that describes a command line, the input Files, and the Environment in which the Task should run.

A Result object contains information about the completed execution of a Task, including identifiers for the output files (which were not known until the Task completed) along with the time and resources consumed during execution.

An **Environment** is an explicit statement of the hardware and software needed to execute a Task. An Environment can take many forms, depending on the technology used and the priority placed on reproducibility compared to convenience.

For example, an Environment could be a virtual machine image with instructions for creating and using a virtual machine for executing Tasks. Such an Environment may be more likely to be reproduced in 10 years than an Environment which is a reference to a virtual machine image in Amazon EC2 or a container image in a public Docker Hub. However, a reference to Amazon EC2 might be more convenient in the short term. An Umbrella [16] Environment can choose an efficient mode when available, but include backup modes if needed. However, even a tarball of software needed on top of an assumed operating system can be considered an Environment.

We assume that an Environment is something created infrequently by working closely with a system administrator, in the same way that a physical machine's operating system is infrequently changed and constantly re-used.

Figure 2 illustrates how a Task relates to an Environment. To execute a Task, the Environment is used to create a temporary sandbox. Any input File arguments are mapped to local pathnames within the sandbox ["in.txt", "in.dat"] where they can be accessed via the running command. After the command is executed, the output files are retrieved from their expected location ["out.txt", "o2.txt"] where they can be extracted and stored within the PRUNE repository as Files and a Result. The remainder of the sandbox is discarded.

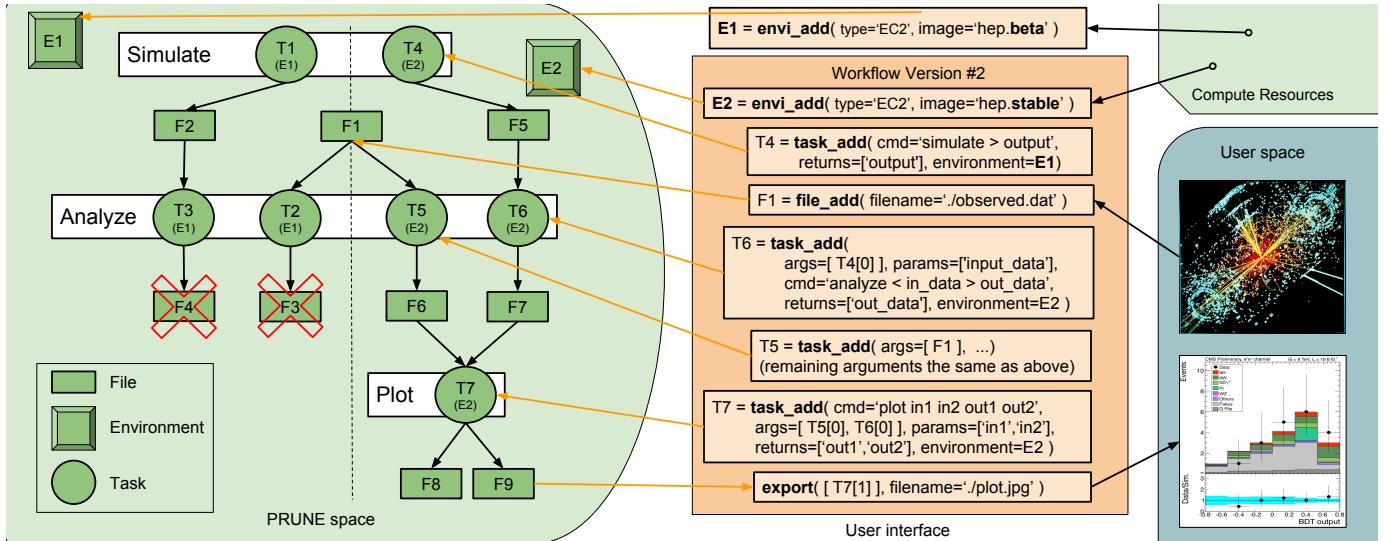


Fig. 3: Overview of a PRUNE repository: Files, Tasks and Environments are used to explicitly describe an evolving workflow from start to the current state.

C. Interface

Prune has six fundamental operations:

```
id = file_add( filename );
id = task_add( task-description );
id = envi_add( type, image );
execute( available_resources );
export( id-list, filename, options );
import( filename );
```

Three operations add to the repository: `file_add` adds a file to the repository from the local filesystem, and returns an identifier for its File object. `task_add` adds a Task to be executed to the repository and then immediately returns an identifier. The Task is queued for execution and the results will become available when time and resources permit. `envi_add` adds a new Environment to the repository, specifying the type of the environment (VMWare, Amazon, Docker, TGZ, etc) and the name of the image.

The `execute` command specifies what resources can be used to execute Tasks, and when they are to be used. The `export` operation creates a package which includes a subgraph of the repository. It expects of a query anchor (a list of ids as a starting point) and options that describe which direction(s) to follow derivation lines and which object types to include in the package. The `import` operation adds new objects into the repository from such a package. Because `task_add` returns an identifier before executing the Task, it is possible that an export will request File objects that do not yet exist. It is a matter of preference whether such a request will block or require the user to poll until objects are available.

D. Example

A snapshot of a workflow is illustrated in Figure 3. Here, we show a common workflow pattern in high energy physics. A researcher runs several simulations that mimic the behavior of a device like the Large Hadron Collider. The behavior

of simulations and observations are analyzed separately. The analyses are then plotted together to produce a publishable graphic. Each step of this process may be repeated many times with different parameters, and then adjusted and retried as the user refines the workflow.

The left side of Figure 3 shows a graph stored in a PRUNE repository, while the middle of the figure shows some operations used to add new items to the repository. The dotted line divides the objects into separate versions of the workflow. F1 (on the dotted line) is the observed data and is used in both revisions of the workflow. Initially, the user ran simulation T1 in Environment E1 to produce synthetic data F2, which was analyzed by T3, producing F4. Then, some actual data was imported as F1 and analyzed by T2, producing F3.

Suppose that the user realizes that the (crossed out) Files F3 and F4 are invalid results, due to some bug in the supporting libraries found in Environment E1. To remedy this, the user prepares a second Environment E2 with new libraries, and then runs T4, T5 and T6 to simulate and analyze the same data again. Finally, the simulated and real data are combined into a single plot (T7) that produces files suitable for graphing.

Figure 4 shows some examples of export being used for collaboration. Export can be used to retrieve how a result was generated (lineage) or what objects were derived from an anchor object (progeny).

E. Naming

The issue of naming in computing has long been a challenge and various approaches have been proposed to resolve the disconnect between computer and human naming. [17] PRUNE uses two types of identifiers for objects: content-based identifiers and derivation-based identifiers.

A content based identifier (CBID) is the fundamental name for all Files, Tasks, and Environments. It is generated by computing a hash function of either the content of the object, which is the binary data of a File, or the JSON document

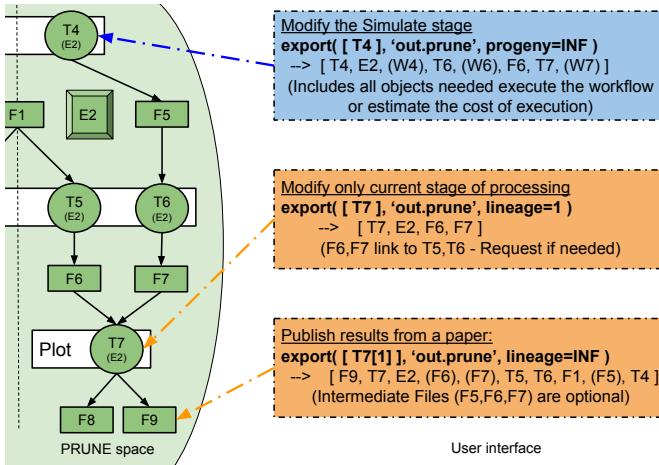


Fig. 4: Export Example: Different options determine which objects are shared.

representing a Task or an Environment. Care must be taken to ensure the ordering of JSON elements (alphanumeric or fixed order keys) so that a CBID does not change as the item is shared among repositories.

PRUNE also stores some auxiliary meta-data about each object type, such as owner, creation time, resources consumed, etc. This meta-data is excluded from the checksum so that the CBID can be used to detect if an object is logically unique.

A derivation-based identifier (DBID) is used to identify files that have not yet been generated. It consists of the CBID of a Task, followed by a subscript that selects one of the results of the Task. DBIDs can be used as arguments to later tasks, so that multiple Tasks can be chained together before the intermediate Files have even been generated.

For example, suppose that Task T consumes files A and B (which exist in the repository) and produces files X and Y. The CBIDs for Files A and B are used in the JSON document that describes Task T. The CBID for Task T is simply the checksum of its JSON document (38b1d). When Files X and Y are produced, they can be addressed using the CBIDs computed from their checksums. But they may also be addressed as 38b1d[0] and 38b1d[1], which indicate they are the first and second output Files of Task T respectively.

Task:	(A,B)→	T	→ (X,Y)
CBID:	18f23, a3f91→	38b1d	→ 93d8a, 413ca
DBID:			→ 38b1d[0], 38b1d[1]

Result objects record the mapping between DBIDs and output CBIDs (once the Task has been executed). Keeping this information separate from the Task allows the Task to remain immutable. Sometimes generated Files are deleted to make room for other Files as mentioned in section III-A. If those Files are needed again, the Task is re-executed, generating an additional Result object for the Task. If derived Files are deleted, the checksums in the Result can be used to validate re-generated output Files.

F. Non-Determinism

If a Task is non-deterministic, multiple executions of the Task can generate Files that are bitwise different, but logically equivalent for a given scientific domain. PRUNE is unable to detect such logical equivalence. For example, this can happen with the Monte Carlo simulations used in high energy physics workflows. In these cases a single DBID can refer to multiple CBIDs. Since the input File identifiers are part of a Task's checksum, equivalent Tasks could end up with (any number of) different Task CBIDs.

In an effort mitigate this issue while still allowing the workflows to be fully specified before execution, PRUNE encourages, when possible, the use of DBIDs throughout. This enhances the ability to effectively collaborate and de-duplicate, which is discussed in later sections, but CBIDs can also be used where the user feels it is more appropriate.

IV. STORAGE MANAGEMENT

One of the challenges with preserving a workflow is the amount of storage space required. We observe (and assume) that, in general, the largest portion of the storage requirement for a scientific workflow consists of Files generated during the execution of a workflow. These **derived** Files can be **leaf** Files (not used as an argument for any Task) or **intermediate** Files (used as an argument in one or more Tasks). We propose treating derived Files as a disposable portion of a workflow as detailed in section IV-A. We assume that the second largest portion of the storage requirement is typically **root** Files (external input data directly imported into a Prune repository). We discuss ways to address this challenge in section IV-B. The smallest portion of the storage requirement is the data describing the Tasks needed to get from the root Files to the leaf Files. Reducing the storage requirements in this category is covered in section IV-C.

A. Derived File Cache

Derived Files can be deleted to save disk space without limiting reproducibility, since all the information needed to recreate them is found in the Tasks, root Files, and Environments. In a sense, these derived Files can be treated as a temporary cache. The Result objects remain in the database for consumed resource statistics and checksum validation.

The priority used to determine which derived Files to evict first could be as simple as evicting the oldest derived File. However, more advanced algorithms could be based on File sizes and their position in the repository graph. The same algorithms used to follow lineage and progeny in the export operation could also be useful in deciding which derived Files are the least likely to be used. The cost (financial or otherwise) of reproducing a File should also be considered.

B. External Objects

Since root Files cannot be re-generated, they must be set apart from the derived Files to prevent the system from disposing of them. An advanced implementation of PRUNE could extend Tasks to allow input files specified as URLs rather

than restricting them to Files only. In such a case, additional rules (based on the bandwidth, reliability and longevity of the external resource) would be needed to determine whether the results of such Tasks could be generated again in the future.

For very large workflows, a smaller repository could treat derived Files from another repository as rooted files, but also include a Task that refers to the full repository for additional lineage. This permits flexibility in constructing repositories appropriate for a given researcher, while still ensuring full preservability (spanning multiple repositories) back to the root Files. In some cases there should be overlap between repositories for added replication and availability, but for others it would be sufficient to simply have a well defined line between repositories.

This is in line with large central data approaches like IVOA [18], IRIS [19], the LHC [20], etc., but any changes to the data by the managing organization must be detectable and/or avoidable in the interest of ensuring reproducibility.

C. Workflow merging

Recording each workflow DAG individually in a PRUNE repository satisfies the need for preservation. However, this can cause unnecessary duplication of Task objects and their executions. Even with the assumption that Task objects are small compared to File objects, eliminating duplication at this level can result in more efficient use of both storage and execution resources.

We observe that as a researcher creates a workflow, there is generally a gradual evolution of that workflow while adjustments are made. Only a portion of the PRUNE objects describing the workflow will change with each evolution. Especially for changes made closer to the leaf Files, or by extending from leaf Files, only a small portion of the objects will differ from a previous version of the workflow. To merge a new workflow into a repository, PRUNE identifies the duplicates and effectively grafts the new objects onto a merged repository graph.

The expanded graph after de-duplication describes both the old and the new workflows simultaneously with shared objects defining the earlier portions of the workflow. As the workflow continues to evolve the graph continues to expand. This expanded graph approach makes up a more efficient PRUNE repository. The ability to detect duplicate Tasks coupled with the ability to treat their generated results as a cache enables memoization. This optimization technique reduces the time it takes to execute a workflow which already includes generated Files in the repository.

In order to support queries (such as those for the export operation) on a merged repository graph, tracing the lineage of the query anchor forward can be enabled by attaching a workflow identifier to each new object added to the graph. However, since any existing duplicate objects are immutable, they cannot be updated with a list of workflows they were used in. When tracing the progeny of the query anchor backwards, there may be multiple paths that could be traversed. This could happen, for example, if two Tasks achieve identical results, but

reached those results using a different approach. In order to ensure that the progeny of a result matches how the result was achieved, an identifier based on the workflow needs to be used in addition to a CBID and DBID.

V. PROTOTYPE

PRUNE v1 is written in Python and uses SQLite3 to keep track of all workflows submitted to it. The user creates a Python script which uses a PRUNE v1 client library to expose PRUNE v1 operations inside of the Python script. The client library translates API commands into SQLite3 queries to preserve new workflow objects and ignore duplicate objects when detected. The client library can also export or import entire workflows or portions of workflows.

A PRUNE v1 repository is a database of workflow objects recorded over time. It is divided into 3 parts; **persistence**, **cache**, and **status**. Both the cache and status portions can be re-created by PRUNE v1, but the persistence portion contains objects that contain irreplaceable information. The cache portion stores generated Files. The persistence portion stores the remaining objects. The status portion tracks the progress of Tasks that still need to be executed and which of those are ready to execute immediately as compared to those that depend on Files which are not yet available in the cache.

SHA1 checksums are computed on object content to create the CBIDs. When the content is in JSON format (Tasks, Environments, and Results), the keys are sorted alphanumerically to keep the CBIDs consistent.

DBIDs use a ‘:’ character after the Task CBID, followed by an index number to distinguish between outputs of a given Task. To encourage meaningful variable names in Python `task_add` returns the list of DBIDs instead of the CBID for the Task. The CBID prefix is still available if needed.

PRUNE v1 treats 2 Tasks which are identical except the Environment, as separate Tasks in the database. Each Task must be executed, and each Result stored, but if the generated Files are identical, they are only stored once (using the CBID and first DBID).

An export in PRUNE v1 creates a single file with all relevant objects embedded. This file can be shared with other users of PRUNE v1 either directly or via the internet.

If any Files requested in the export command have not been generated or were evicted from the cache, the user receives a message indicating that Files are not yet available. The user may then repeat the request until the results are available.

A. Compute Resources

Prune can either spawn **local** worker processes to execute Tasks, or start a Work Queue [21] master to coordinate Task execution on **remote** workers. In local mode, input Files are linked into Task sandboxes, with the assumption that Tasks will “play nice” and not modify those files. This is how files are treated when executing commands outside of PRUNE, and is appropriate for the high energy physics and census workflows we considered. In remote mode, Files must be

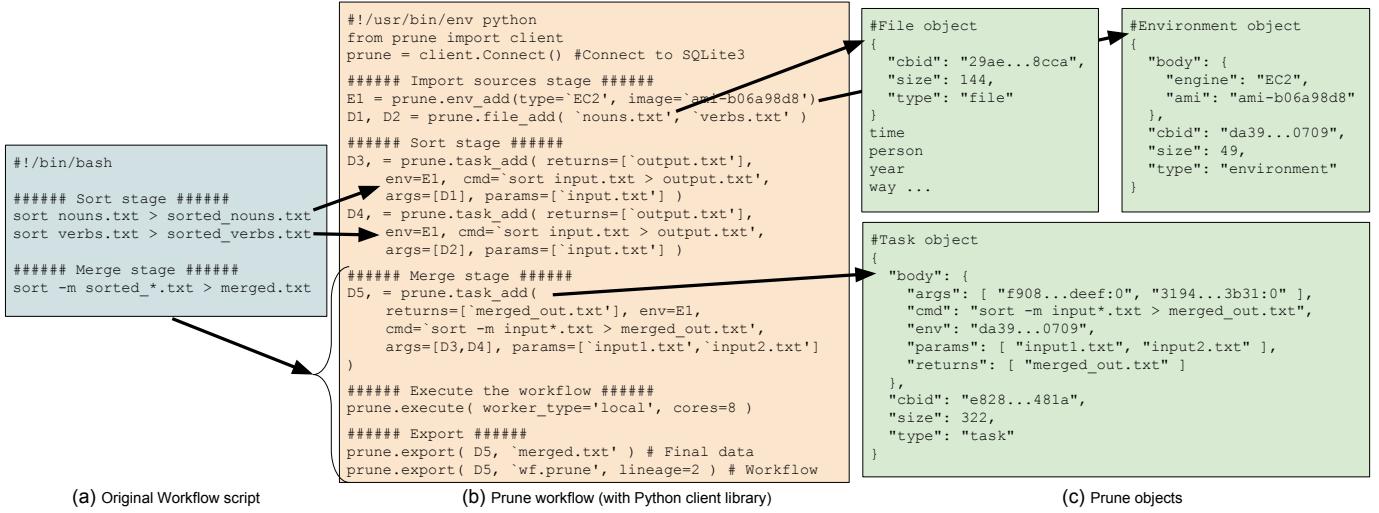


Fig. 5: Example Workflow: An example workflow (a) is shown using PRUNE commands (b), with a few of the individual objects that are recorded (c).

transmitted over the network, making it more appropriate for computationally intensive Tasks with small inputs.

PRUNE v1 puts all submitted Tasks (which don't have their output files in the cache) into the status portion of the database. These Tasks are eagerly evaluated whenever a prune_worker is running. When the command for a locally run Task returns an error code, the sandbox is left intact so the user can see what modifications would be needed to submit a corrected Task.

PRUNE v1 currently allows Tasks to run without a specified environment (meaning that the default available environment should be used), with a Wrap environment, or with a local Umbrella [16] environment. A Wrap environment runs an *open* command to prepare the environment for command execution (then an optional *close* command). A Wrap environment was used to extract a tarball with software needed for the workflows used in evaluating PRUNE v1.

B. Example Workflow

Consider the shell script shown in figure 5a designed to take two input files and efficiently produce a new file with all lines merged and sorted.

The Python script in figure 5b will preserve and execute a workflow equivalent to figure 5a. The last line exports the minimum objects needed to reproduce the workflow, and saves these objects in the “merge_sort.prune” file.

The PRUNE v1 client library converts the script at figure 5b into the PRUNE v1 (slightly abbreviated) objects at figure 5c which are not exposed directly to the user. These objects are what is stored in the PRUNE v1 repository.

This may seem verbose compared to the original workflow. But we claim that the benefits of adopting a preservation-first strategy (beyond just the preservation benefits) can outweigh the added complexity. The following section evaluates some of those benefits.

VI. EVALUATION

In order to evaluate the storage management abilities, computational overhead, and scalability of PRUNE v1, it was used to manage workflows doing some analyses on U.S. Census records. The U.S. Census [22] for years 1850 to 1940 consume 23 GB using 7-Zip compression. In a “Matching” workflow, the censuses are searched for instances where identical attributes occur exactly once in a pair of censuses, indicating a high probability both records refer to the same person in real life. Due to spelling, transcription, and other errors, a workflow with exact matching achieves very few matches.

To improve the matching results, a “Comparison” workflow creates a list of the most frequent surnames in all censuses and compares it against the list of all surnames to obtain lists of possible alternate spellings. The goal is to use these alternate spellings to feed fuzzy matching (rather than exact matching) into the Matching workflow. The Matching workflow is broken down into the following 7 stages:

Matching workflow stages

- 1 Decompress
(7-Zip unpacking)
- 2 Normalize
(Standardize field inclusion, names, and order)
- 3 Map key split
(Split into blocks that will include matches)
- 4 Summarize year
(Merge into 1 file per year per block)
- 5 Merge pairs
(Merge each pair of years together)
- 6 Group by key
(Make groups based on block key)
- 7 Find 1-1 matches
(Find unique matches - exactly 1 entry per year)

Importing original files into PRUNE v1 is more a part of PRUNE v1 behavior than that of the workflow, so we

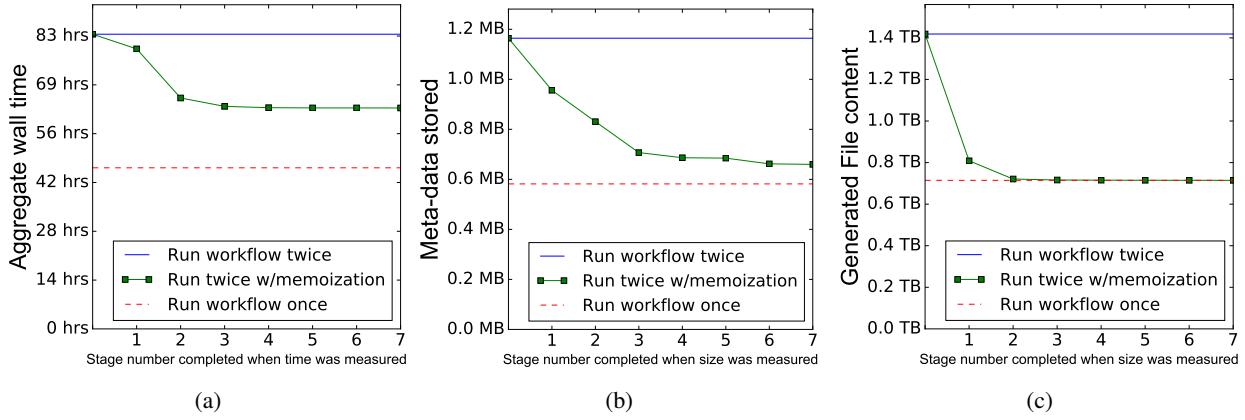


Fig. 6: When changes to a workflow occur in later stages, PRUNE (a) avoids duplicate execution, (b) avoids extra disk space used to specify the workflow, (c) avoids extra disk space used for generated Files.

consider this Stage 0. Stages 0-2 are identical between the two workflows. The “Comparison” workflow has 6 unique stages:

Comparison workflow stages

- 3 Count attributes
(Count appearances of field-attribute pairs)
- 4 Summarize year
(One file per year summarizing pairs in that year)
- 5 Summarize all
(A single file for summarizing pairs across all years)
- 6 Filter by field
(A separate file for each field type)
- 7 Sort by frequency
(Most frequently occurring attribute on top)
- 8 Similar attributes
(Score similar alternates for most frequent surnames)

A. Collaboration

PRUNE v1 can be used to facilitate evolutionary changes by multiple users concurrently. Take for example a situation where one user finds an interesting match and wants to share those results with another user. One of the result files in the full comparison workflow was chosen as an export query anchor. The exported package with all tasks and root and intermediate files resulted in a 1.5TB file and took 1 hour and 25 minutes. But it only took 3 seconds to create a 2.6GB package with only the root Files and the Tasks, and it took 5 minutes and 30 seconds to read the package and recreate the query anchor File on a separate machine. In 4 seconds, another 2.6GB package was created with the Tasks, root Files, and the anchor File. The anchor didn't need to be generated on a separate machine, but all information was available to reproduce the File if desired.

Re-importing any of these exports back into the original repository has no effect as PRUNE v1 detects duplicates and ignores them. However, consider a situation where slight changes are made to the workflow by the collaborator. Importing a new export received from the collaborator, would still result in the detection and ignoring of duplicate objects, and then any new portions of the workflow would be added to the repository.

B. Conservation

A common approach to preservation is to create a separate folder for each snapshot of all scripts and files each time a paper is published or some other milestone. In figures 6a, 6b, and 6c comparisons are made between this situation where two versions of the Comparison workflow are in separate folders (upper line) compared to a situation where only one version of the workflow exists (lower line).

The line in the middle shows the resources consumed by storing both workflow versions in PRUNE v1, after making a change to the workflow stage number indicated on the x-axis.

In figure 6a, the wall time improvements due to memoization are modest in the first stage since it is not very CPU intensive. The normalization stage is more significant computationally. The final stage is the next most significant one in terms of computation. Doing an all-pairs match on surnames using the Jaro-Winkler algorithm [23] is computationally expensive, so even changes to only that final stage still require a significant amount of work.

The measurements in figures 6a, 6b, and 6c were taken after doing comparisons on only 100 of the 11,400,952 unique surnames in the censuses. Executing more comparisons is covered in the following sections.

In figure 6b File content (but not metadata) is ignored. A workflow change in the first 3 stages results in a larger database because of the large number of files generated by those stages. The later stages have a more negligible affect on the database size. This indicates PRUNE v1 might be most effective when evolutionary changes to a workflow are made at the leaves of the workflow rather than at the roots.

Figure 6c shows the intermediate File space. The decompress stage creates large files with duplicate and extraneous (in this context) fields. This data is included in the graph even though it is only stored once in the PRUNE v1 database.

The normalize stage then strips much of that out and produces smaller files. All other stages have comparatively small intermediate Files. This is great for PRUNE v1 because the unpacked data becomes a better candidate for eviction from

TABLE I: WALL CLOCK TIME OVERHEAD

	Preserve workflow	Prepare Execution	Execute Tasks	Checksum results	Preserve executions	Total Time	Wall clock time overhead	Total # of Tasks/Files	Space (MB)
Import sources	1:21	-	-	-	-	1:21	100%	168	24.37
Decompress	~0	0:10	1:41:33	5:19:36	0:25	7:01:43	315%	168	609,984
Normalize	~0	0:12	10:16:11	52:30	1:48	11:10:42	9%	167	86,234
Count attributes	~0	0:01	5:41:12	0:18	~0	5:41:33	~0%	167	4,799
Summarize year	~0	~0	22:05	0:03	~0	22:08	~0%	10	819
Summarize all	~0	~0	4:22	0:01	~0	4:24	~0%	1	407
Filter by field	~0	~0	0:07	0:01	~0	0:09	24%	16	407
Sort by frequency	~0	~0	2:02	0:02	~0	2:04	1%	16	407
Similar attributes	0:25	2:52	544:18:38	8:26	3:00	544:37:39	~0%	10,000	102,689
Total	1:47	7:16	562:26:11	6:20:57	5:13	569:01:43	1%	10,713	830,114

the cache since the normalized data will be used more often than the raw unpacked data.

However, all the data depicted in figure 6c is a candidate for eviction. In extreme cases, intermediate files could be deleted as soon as they are consumed by later tasks.

C. Overhead

To measure the overhead of PRUNE, the Comparison workflow was executed to produce a list of similar surnames for each of the 10,000 most frequent surnames (Stage 8). This workflow was executed using only local workers because the files were large compared to the compute resources needed to process them for these stages. The execution time, wall clock time overhead and data storage requirements for each stage is shown in table I.

Stage 0 (the “import sources” stage) is included here as 100% overhead, since PRUNE v1 must make a copy of all the original data, whereas in preserve later system, the files in user space would be used directly. It is interesting to note that checksumming the files after the decompression stage is more than 3 times more computationally expensive than just decompressing the files. Two options are available to address this issue. Option 1) Skipping a checksum of Files altogether (perhaps when Files are large) would result in less computational time, but the system might have to transfer and store duplicate copies of the data. This might not be bad since this data is intermediate and can be evicted from the cache anyway. Option 2) Checksumming in the background could both avoid the immediate delay and the duplicate storage. However, when Tasks are executed remotely (see the Scaling section below), the data still has to be transferred twice.

However, while this overhead seems significant when looking at that one stage, the overall overhead is only around 1%. The low overhead in the CPU intensive final stage (with a relatively small input file) makes the overhead in the decompress stage much less significant.

PRUNE v1 chooses to always do duplicate elimination as in some cases this can also lead to avoiding the re-execution of later stages if the duplicate is caught early on. Also, this overhead is likely to only occur for the first evolution of the workflow. Only a change in the environment for stage 1 or a

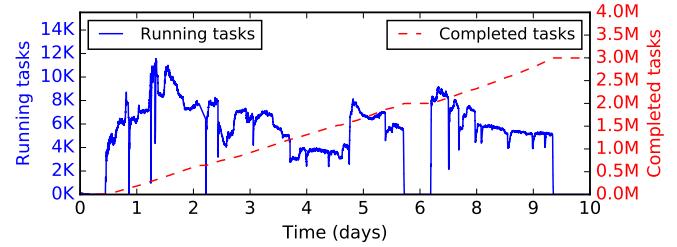


Fig. 7: Tasks running over time: 3 million Tasks were executed within 10 days with a concurrency of O(10k).

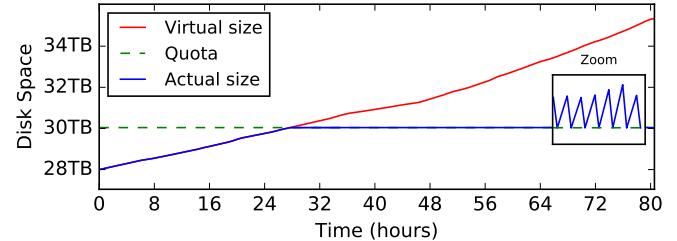


Fig. 8: Virtual vs. actual storage with quota: A storage quota system held disk consumption close to 30TB during an additional ~864k tasks.

change to the files in stage 0 would result in having to perform these checksums again.

D. Scaling

For the scaling evaluation, the earlier stages of the workflow are mostly disk intensive, so they were performed using 16 local processes on the server to avoid network transfer congestion and delays. The final stage is more CPU intensive, so a Work Queue master in Prune with O(10k) remote workers was used to bring the total number of surname comparisons to 3 million. Figure 7 shows the concurrency of Tasks running for about 9 days. The total storage space for the entire workflow after these 3 million+ Tasks was about 28TB.

E. Storage Quota

In any storage-constrained system, it is important to keep the intermediate data within those constraints. While executing an additional ~864k Tasks of the workflow, PRUNE v1 was given a quota of 30TB. Prune v1 appropriately removed Files from

the repository cache whenever it observed that new generated Files caused the repository to go over quota.

Figure 8 shows that Prune stayed within about 700MB of the quota after reaching the quota. This was done in the background to avoid interference with the remote workers.

F. Reproducibility

We do not have permission to share the root Files for the census workflows, so the full workflows cannot be shared in their entirety. However, we simulated some census data so that it could be used as root files for Stage 3 and beyond (no need for unzipping or normalizing the simulated data). PRUNE and example workflows (including the simulated census and a high energy physics workflow using Umbrella [16]) can be downloaded and executed by following the instructions at: <http://ccl.cse.nd.edu/software/prune/>

VII. RELATED WORK

PRUNE builds on, and derives from, many existing technologies. It combines capabilities from revision control systems, workflow management systems, and tools for hardware/software reproducibility. Other systems that combine such capabilities were also considered and drawn from.

Revision control systems like Git[9], CVS[24], and Subversion[25] track changes a user makes to files in a directory tree rather than tracking a graph of task provenance. “Git and Org-Mode”[26] augments Git to be more applicable for workflow preservation. However, there is also normally an underlying assumption that a repository is small enough that making a full copy of both the current state and state history is not a problem. These revision control systems expect the user to decide when something should be committed. This is more efficient and leads to cleaner histories, but runs the risk of the user forgetting to commit important states.

A user might attempt to include portions of the needed environment, such as a compiler, with a repository, but more often than not, the necessary environments are merely implied. In a “preserve-first” source code version control system, every time a user wanted to check if a change to source code is working, the changes would first be committed (just in case the changes are “good”), and the system would automatically compile or test the code, and then present the results (or errors) to the user, with the assumption that “bad” commits can later be removed, if needed.

Management systems such as Pegasus [27], Swift [28], Galaxy [29], and Nextflow [30] are specifically geared towards workflows. They vary widely on their support of reproducibility, collaboration, resource constraints, and preservation. Pegasus focuses on automating compute and storage resource consumption while facilitating recovery and debugging of failures. Swift uses implicit parallelism in its programming model to automatically scale workflows while requiring minimal programming expertise. Galaxy is designed specifically for biomedical research and uses a web interface to facilitate collaboration. Nextflow uses streams (based on Unix pipes) to automate concurrency for any programming language.

A few systems track workflows over time. VisTrails [31] tracks workflows that generate images. An Eidetic System [32] tracks and records the evolution of the file system on a single machine and can replay previous events. Nectar [10] centralizes computations performed in a datacenter in connection with their results and supports treating result data as a cache similar to the data reuse feature in Pegasus. [27]

CDE [5], PTU [6], ReproZip [7], and Parrot [33] are ways of capturing an Environment designed to execute a specific Task (but maybe not a more generic one). Transparent result caching [34] was designed to continuously and automatically record all dependencies over time on a single machine.

Preserving a full Environment (designed for a generic category of Tasks) is made possible with Docker [35], Umbrella [36], and virtual machine images. They can create an instance of an entire software stack for Tasks which have been modified from their original execution.

GridDB [37], Apt [38], Taverna [39], dataref versuchung [40], and Paper Mâché [11] are also related. With GridDB and Apt, the user directly executes operations in the workflow and the system assumes they will preserve how the execution was performed. GridDB includes detailed specifications for intermediate data allowing automatic parallelization between operations. Taverna is designed to assist bioinformaticians using web services to share sequence analysis methods online. Dataref versuchung and Paper Mâché encourage automation of the full publication life-cycle when it can be contained on a single machine.

VIII. CONCLUSION

The PRUNE framework is designed to facilitate reproducible work, by capturing dependencies in a way that is easily shared, but also easily modified, extended, and understood. This seems in line with modern research directions [41] encouraging collaboration across traditional boundaries such as applied vs. basic research and engineers vs. scientists vs. designers.

With a proven concept, a few important qualities in Prune could be applied to other systems. An existing workflow system could adopt a preserve-first strategy by exporting all executed tasks (at any granularity) into a common repository which can then be queried as needed or pruned down to only the most relevant information, if necessary.

If the generated results are included in the exported data, those results can be saved and the workflow system can check the repository for cached results before executing anything. Such a system could use derivation or content based identifiers, or a combination of the two (as in Prune), or a full database implementation might have some advantages also.

Of course, technology is only one piece of the reproducibility puzzle. The problem of reproducibility encompasses publication practices, intellectual property, incentive systems, and many other issues [42]. But we hope that by reducing the technical burden of reproducibility, we can stimulate the erosion of other barriers to scientific progress.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under grants PHY-1247316 and OCI-1148330, and the Department of Education under grant P200A120206.

REFERENCES

- [1] J. P. Ioannidis, "Why most published research findings are false," *PLoS Med.*, vol. 2, no. 8, p. e124, 2005.
- [2] L. P. Freedman, M. C. Gibson, S. P. Ethier, H. R. Soule, R. M. Neve, and Y. A. Reid, "Reproducibility: changing the policies and culture of cell line authentication," *Nature methods*, vol. 12, no. 6, pp. 493–497, 2015.
- [3] B. Nosek, G. Alter, G. Banks, D. Borsboom, S. Bowman, S. Breckler, S. Buck, C. Chambers, G. Chin, G. Christensen, et al., "Promoting an open research culture: Author guidelines for journals could help to promote transparency, openness, and reproducibility," *Science (New York, NY)*, vol. 348, no. 6242, p. 1422, 2015.
- [4] H. Meng, M. Wolf, P. Ivie, A. Woodard, M. Hildreth, and D. Thain, "A Case Study in Preserving a High Energy Physics Application with Parrot," in *Journal of Physics: Conference Series (CHEP 2015)*, 2015.
- [5] P. J. Guo and D. R. Engler, "Cde: Using system call interposition to automatically create portable software packages," in *USENIX Annual Technical Conference*, 2011.
- [6] Q. Pham, T. Malik, and I. Foster, "Using provenance for repeatability," in *Presented as part of the 5th USENIX Workshop on the Theory and Practice of Provenance*, 2013.
- [7] F. Chirigati, D. Shasha, and J. Freire, "Reprozip: Using provenance to support computational reproducibility," in *Presented as part of the 5th USENIX Workshop on the Theory and Practice of Provenance*, 2013.
- [8] D. Thain, P. Ivie, and H. Meng, "Techniques for Preserving Scientific Software Executions: Preserve the Mess or Encourage Cleanliness?," in *12th International Conference on Digital Preservation (iPres)*, 2015.
- [9] J. Loeliger, "Collaborating with git," *Linux Magazine*, June, 2006.
- [10] P. K. Gunda, L. Ravindranath, C. A. Thekkath, Y. Yu, and L. Zhuang, "Nectar: Automatic management of data and computation in datacenters," in *OSDI*, vol. 10, pp. 1–8, 2010.
- [11] G. R. Brammer, R. W. Crosby, S. J. Matthews, and T. L. Williams, "Paper mâché: Creating dynamic reproducible science," *Procedia Computer Science*, vol. 4, pp. 658–667, 2011.
- [12] T. Proebsting and A. M. Warren, "Repeatability and benefaction in computer systems research," 2015.
- [13] K. Belhajjame, C. Goble, S. Soiland-Reyes, and D. De Roure, "Fostering scientific workflow preservation through discovery of substitute services," in *E-Science (e-Science), 2011 IEEE 7th International Conference on*, pp. 97–104, IEEE, 2011.
- [14] E. W. Dijkstra, "Letters to the editor: go to statement considered harmful," *Communications of the ACM*, vol. 11, no. 3, pp. 147–148, 1968.
- [15] S. I. Feldman, "Make – A program for maintaining computer programs," *Software: Practice and experience*, vol. 9, no. 4, pp. 255–265, 1979.
- [16] H. Meng and D. Thain, "Umbrella: A portable environment creator for reproducible computing on clusters, clouds, and grids," in *Proceedings of the 8th International Workshop on Virtualization Technologies in Distributed Computing, VTDC '15*, (New York, NY, USA), ACM, 2015.
- [17] A. Asserson, K. G. Jeffery, and A. Lopatenko, "Cerif: past, present and future: an overview," 2002.
- [18] T. McGlynn, G. Fabbiano, A. Accomazzi, A. Smale, R. L. White, T. Donaldson, A. Aloisi, T. Dower, J. M. Mazzerella, R. Ebert, et al., "Providing comprehensive and consistent access to astronomical observatory archive data: the nasa archive model," in *SPIE Astronomical Telescopes+ Instrumentation*, pp. 99100A–99100A, International Society for Optics and Photonics, 2016.
- [19] M. van Driel, A. Hutko, L. Krischer, C. Trabant, S. Stähler, and T. Nissen-Meyer, "Syngine: On-demand synthetic seismograms from the iris dmc based on axisem & instaseis," in *EGU General Assembly Conference Abstracts*, vol. 18, p. 8190, 2016.
- [20] C. Lynch, "Big data: How do your data grow?," *Nature*, vol. 455, no. 7209, pp. 28–29, 2008.
- [21] P. Bui, D. Rajan, B. Abdul-Wahid, J. Izaguirre, and D. Thain, "Work Queue+Python: A framework for scalable scientific ensemble applications," in *Workshop on Python for High Performance and Scientific Computing at SC11*, 2011.
- [22] FamilySearch.org, "United States Census, 1850-1940." Database. Citing NARA microfilm publication T626. Washington, D.C.: National Archives and Records Administration, 2002.
- [23] W. Cohen, P. Ravikumar, and S. Fienberg, "A comparison of string metrics for matching names and records," in *Kdd workshop on data cleaning and object consolidation*, vol. 3, pp. 73–78, 2003.
- [24] P. Cederqvist, R. Pesch, et al., "Version management with cvs," 1992.
- [25] B. Collins-Sussman, "The subversion project: building a better cvs," *Linux Journal*, vol. 2002, no. 94, p. 3, 2002.
- [26] L. Stanisic, A. Legrand, and V. Danjean, "An effective git and org-mode based workflow for reproducible research," *ACM SIGOPS Operating Systems Review*, vol. 49, no. 1, pp. 61–70, 2015.
- [27] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, et al., "Pegasus: A framework for mapping complex scientific workflows onto distributed systems," *Scientific Programming*, vol. 13, no. 3, pp. 219–237, 2005.
- [28] M. Wilde, M. Hategan, J. M. Wozniak, B. Clifford, D. S. Katz, and I. Foster, "Swift: A language for distributed parallel scripting," *Parallel Computing*, vol. 37, no. 9, pp. 633–652, 2011.
- [29] B. Giardine, C. Riemer, R. C. Hardison, R. Burhans, L. Elnitski, P. Shah, Y. Zhang, D. Blankenberg, I. Albert, J. Taylor, et al., "Galaxy: a platform for interactive large-scale genome analysis," *Genome research*, vol. 15, no. 10, pp. 1451–1455, 2005.
- [30] R. Garcia and M. T. Valente, "Nextflow: Business process meets mapping frameworks,"
- [31] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo, "Managing the evolution of dataflows with vistrails," in *Data Engineering Workshops, 2006. Proceedings. 22nd International Conference on*, pp. 71–71, IEEE, 2006.
- [32] D. Devecsery, M. Chow, X. Dou, J. Flinn, and P. M. Chen, "Eidetic systems," in *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pp. 525–540, 2014.
- [33] D. Thain and M. Livny, "Parrot: An Application Environment for Data-Intensive Computing," *Scalable Computing: Practice and Experience*, vol. 6, no. 3, pp. 9–18, 2005.
- [34] A. Vahdat and T. E. Anderson, "Transparent result caching," in *USENIX Annual Technical Conference*, 1998.
- [35] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux Journal*, vol. 2014, no. 239, p. 2, 2014.
- [36] H. Meng and D. Thain, "Umbrella: A Portable Environment Creator for Reproducible Computing on Clusters, Clouds, and Grids," in *Workshop on Virtualization Technologies in Distributed Computing (VTDC) at HPDC*, 2015.
- [37] D. T. Liu and M. J. Franklin, "Griddb: a data-centric overlay for scientific grids," in *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pp. 600–611, VLDB Endowment, 2004.
- [38] R. Ricci, G. Wong, L. Stoller, K. Webb, J. Duerig, K. Downie, and M. Hibler, "Apt: A platform for repeatable research in computer science," *ACM SIGOPS Operating Systems Review*, vol. 49, no. 1, pp. 100–107, 2015.
- [39] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, and T. Oinn, "Taverna: a tool for building and running workflows of services," *Nucleic acids research*, vol. 34, no. suppl 2, pp. W729–W732, 2006.
- [40] C. Dietrich and D. Lohmann, "The dataref versuchung: Saving time through better internal repeatability," *ACM SIGOPS Operating Systems Review*, vol. 49, no. 1, pp. 51–60, 2015.
- [41] B. Schneiderman, *The New ABCs of Research: Achieving Breakthrough Collaborations*. Oxford University Press, 2016.
- [42] J. Myers, M. Hedstrom, D. Akmon, S. Payette, B. A. Plale, I. Kouper, S. McCaulay, R. McDonald, I. Suriarachchi, A. Varadharaju, et al., "Towards sustainable curation and preservation: The sead project's data services approach," in *e-Science (e-Science), 2015 IEEE 11th International Conference on*, pp. 485–494, IEEE, 2015.

Converting Scripts into Reproducible Workflow Research Objects

Lucas A. M. C. Carvalho

Institute of Computing
University of Campinas
Campinas, Brazil

Email: lucas.carvalho@ic.unicamp.br

Khalid Belhajjame

LAMSADE
Paris-Dauphine University
Paris, France

Email: Khalid.Belhajjame@dauphine.fr

Claudia Bauzer Medeiros

Institute of Computing
University of Campinas
Campinas, Brazil

Email: cmbm@ic.unicamp.br

Abstract—Scientific discovery and analysis are increasingly computational and data-driven. While scripting languages, such as Python, R and Perl, are the means of choice of the majority of scientists to encode and run their data analysis, scripts are generally not amenable to reuse or reproducibility. Scripts do rarely get reused or even shared with third party scientists. We argue in this paper that the reproducibility of scripts can be promoted by converting them into *workflow research objects*. A workflow research object encodes a script into a production (executable) workflow that is accompanied by annotations, example datasets and provenance traces of their execution, thereby allowing third party users to understand the data analysis encoded by the original script, run the associated workflow using the same or different dataset, or even repurpose it for a different analysis. To this end, we present a methodology for converting scripts into workflow research objects in a principled manner, guided by requirements that we elicited for this purpose. The methodology exploits tools and standards that have been developed by the community, in particular YesWorkflow, Research Objects and the W3C PROV. It is showcased using a real world use case from the field of Molecular Dynamics.

I. INTRODUCTION

Scripting languages have gained momentum among scientists as a means for enacting computational data analysis. Scientists in a number of disciplines use scripts written in general-purpose languages such as Python, R and Perl in their daily data analysis and experiments. We note, however, that scripts are difficult to understand by third parties who were not involved in their development (and sometimes even by the same scientists who developed them); they are, as such, not amenable to reuse and reproducibility. This is witnessed by a number of initiatives that were launched to bring some of the rich features that traditionally come with scientific workflow systems to manage scripts, see e.g., [1]–[4]. For example, McPhilips *et al.* [1] developed YesWorkflow, an environment for extracting a workflow-like graph that depicts the main components that compose a script and their data dependencies based on comments that annotate the script. Murta *et al.* [3] proposed noWorkflow, which also captures provenance traces of script executions.

While the above proposals bring new useful functionalities for understanding scripts and their execution traces they do not enable reuse or reproducibility of scripts. For example, the workflow-like graph obtained using YesWorkflow is abstract

(in the sense that it cannot be executed by the scientists). On the other hand, the provenance traces captured using noWorkflow are fine-grained, and therefore cumbersome, for the user who would like to understand the lineage of the script results [2].

To address the above issues and complement the landscape of solutions proposed by the community for promoting the reuse and reproducibility of scripts, we present in this paper a methodology for converting scripts into reproducible Workflow Research Objects [5] (WRO). WRO are Research Objects that encapsulate scientific workflows and additional information regarding the context and resources consumed or produced during the execution. In more detail, given a script, the methodology we propose drives the creation of research objects that contain the scripts that the scientist authored together with executable workflows that embody and refine the computational analyses carried out by these scripts. These WROs also encapsulate provenance traces of the execution of those workflows, as well as auxiliary resources that help third party users to understand and reproduce such analyses. Examples of those resources include annotations that describe workflow steps, the hypotheses investigated by the data analyses that the scripts incarnate and the findings that the scientists may have made. We argue that such Workflow Research Objects provide potential users with the means to understand, replicate and reuse the data analyses carried by the scripts or part thereof – thanks to the executable workflows that embody such analyses and the accompanying auxiliary resources.

While developing the methodology, we strived to exploit tools and standards that were developed by the scientific community, in particular, YesWorkflow, Research Objects, the W3C PROV recommendations¹ as well as the Web Annotation Data Model². To showcase our methodology, we use a real-world case study from the field of Molecular Dynamics.

The paper is organized as follows. Section II presents the case study that we use as a running example throughout the paper. Section III identifies the requirements that guided the development of our methodology, which is overviewed in

¹<https://www.w3.org/TR/prov-overview/>

²<https://www.w3.org/TR/annotation-model>

section IV. Sections V through VIII show in detail each step of our methodology. Section IX briefly discusses related work. Finally, Section X concludes the paper underlining our main contributions and discussing our ongoing work.

Throughout this paper, we differentiate between at least two kinds of experts – *scientists* and *curators*. Scientists are the domain experts who understand the experiment, and the script; this paper also calls them, sometimes, *users*. Curators may be scientists who are also familiar with workflow and script programming, or, alternatively, computer scientists who are familiar enough with the domain to be able to implement our methodology. Curators are moreover responsible for authoring, documenting and publishing workflows and associated resources.

II. CASE STUDY - MOLECULAR DYNAMICS

The motions of individual atoms in a multimolecular physical system can be determined if the forces acting on every atom are known; these forces may result from their mutual interactions or from the action of an external perturbation. Determining such motions is key to understanding the physical and chemical properties of a given system of interest.

Molecular dynamics (MD) simulations consist of a series of algorithms developed to iteratively solve the set of coupled differential equations that determine the trajectories of individual atoms that constitute the particular physical system. This involves a long sequence of scripts and codes.

MD simulations are used in many branches of material sciences, computational engineering, physics and chemistry.

A typical MD simulation experiment receives as input the structure, topology and force fields of the molecular system and produces molecular trajectories as output. Simulations are subject to a suite of parameters, including thermodynamic variables.

Many groups have implemented their specific MD simulations using special purpose scripts. In our case, a suite of scripts was designed by physiochemists [6]; its inputs are the protein structure (obtained from the RCSB PDB protein data bank³), the simulation parameters and force field files.

There are many kinds of input files and variables, and their configuration varies with simulation processes. For instance, the input multimolecular structure contains the initial set of Cartesian coordinates for every atom/particle in the system, which will evolve in time in the MD simulation. This initial structure varies according to the system to be simulated and research area. Our case study (biophysical chemistry) requires immersing proteins in a solvent. Protein Cartesian atomic coordinates are made available in specialized data repositories, most notably the Protein Data Bank (PDB). Typical systems contain from several thousands to millions of covalently bound atoms.

The main raw product of any MD simulation is a large set of interrelated molecular trajectories. Trajectory data is usually stored for subsequent analyses and consists of thousands of

time-series of the Cartesian coordinates of every atom of the system.

In this paper, we will use a script that sets up a MD simulation. The script will be presented later on (see Listing 1), and used as a running example throughout the paper.

III. REQUIREMENTS FOR SCRIPT CONVERSATIONS

This section presents the requirements that guided the development of our solution, and section IV outlines the methodology we designed to meet them.

Since scripts are usually fine-grained, they are hard to understand - sometimes even the script author does not understand a script s/he developed in the past. To facilitate the task of understanding the script, its author may modularize the script by organizing it into functions. While modularity helps, the functions that compose the script are obtained through a refactoring process that primarily aims to promote code *reuse via the reuse script*, as opposed to *reuse via the main (logical) data processing units* that are relevant from the point of view of the computational analysis implemented by the script. This leads us to the first requirement.

Requirement 1: To help the scientist understand a script *S*, s/he needs a view of *S* that identifies the main processing units that are relevant from the point of view of the *in silico* analysis implemented by the script, as well as the dependencies between such processing units.

The idea, here, is to provide curators with automatic means to obtain a workflow-like view of the script, i.e., an abstract workflow revealing the computational processes and data flows otherwise implicit in these scripts, displaying modules, ports, and data links. Though graphical visualizations may be useful to promote understandability of scripts, large scripts may result in very large (abstract) workflows. Thus, curators need to be able to create a multi-level view of scripts (e.g., through encapsulation of sub-workflows into more complex abstract tasks), or to pose queries against this workflow-like view.

An abstract workflow is a preliminary requirement to our end-goal, namely, to provide curators with the means to generate a (concrete) workflow that can be executed using a workflow management system. This, in turn, will bring to the scientists benefits that such systems provide, such as retrospective provenance recording.

Requirement 2: The user should be able to execute the workflow that embodies the script *S*.

Though seemingly obvious, this is far from being a trivial requirement. It is not enough to "be able to execute". This execution should reflect what is done in the script *S*. In other words, not only should the workflow generated be executable; the scientist must be given the means to compare its results to those of script execution. In many cases, results will not be exactly the same, but similar. This also happens with script execution, in which two successive runs with identical inputs will produce non-identical results that are nevertheless valid and compatible. Thus, this requirement involves providing means of comparing the execution of script *S* and the workflow, and validating the workflow as a valid embodiment of the script.

³<http://www.rcsb.org/pdb/>

Requirement 3: The curator should be able to modify the workflow that embodies the script S to use different computational and data resources.

Not only may a scientist want to be able to replicate the computational experiment encoded by S ; s/he may want to repeat the analysis implemented in the script using third party resources – e.g., which implement some activities in the workflow via alternative algorithms and/or different and potentially larger datasets. For example, s/he may want to modify a method call in a bioinformatics script that performs sequence alignment with a call to an EBI⁴ web service that performs a sophisticated sequence alignment using larger and curated proteomic datasets. By the same token, in a Molecular Dynamics simulation, a protein data source may be modified.

The new (modified) workflow(s) correspond to versions of the initial workflow. They will help the user, for example, to inspect if the results obtained by script S can be reproduced using different resources (algorithms and datasets). Scientists will also be able to compare the execution of S with that of the versions (e.g., if web services are invoked instead of a local code implementation).

Experiment reusability demands that the appropriate (pieces of) workflow be identified and selected for reuse. It is not enough to publish these pieces: potential users must be given enough information to understand how the workflow came to be, and how adequate it is to the intended uses. This leads us to Requirements 4 and 5, respectively involving the need for provenance information, and the elements that should be bundled together to ensure full reusability.

Requirement 4: Provenance information should be recorded.

This involves not only the provenance obtained by workflow execution. This requirement also implies recording the transformations carried out to transform the script into a workflow that embodies the script. Moreover, the transformations to workflows that modify the initial workflow using different resources also need to be recorded. As stressed by [7], provenance that is provided by the execution of a workflow corresponds to a workflow trace, which can be processed as an acyclic digraph, in which nodes are activities and/or data, and edges denote relationships among activities and data.

Requirement 5: All elements necessary to reproduce the experiment need to be captured together to promote reproducibility.

We follow the definition of [7]: "reproducibility denotes the ability for a third party who has access to the description of the original experiment and its results to reproduce those results using a possibly different setting, with the goal of confirming or disputing the original experimenter's claims." [7] also differentiates reproducibility from repeatability, for which results must be the same, and no changes are made anywhere.

Full reproducibility and reusability require ensuring that all elements of an experiment are recorded. The script S ,

the initial workflow, and all of its versions should be made available together with auxiliary resources that will allow understanding how these workflows came to be, and where they should be used. Such resources must include, at least, the provenance information documenting the transformation from the script to the workflows, datasets that are used as inputs, execution traces of the script and the workflows, as well as textual annotations provided by the curator.

IV. METHODOLOGY TO ASSIST IN SCRIPT CONVERSIONS

To meet the requirements identified in Section III, we devised a methodology for converting a script into reproducible workflow research objects [5], [8]. As the use of workflow specifications on their own does not guarantee support to reusability, shareability, reproducibility, or better understanding of scientific methods, additional information may be needed. This includes annotations to describe the operations performed by the workflow; links to other resources, such as the provenance of the results obtained by executing the workflow, datasets used as input, etc. These richly annotation objects are called workflow-centric research objects [5]. A Research Object [9] provides the means to specify a kind of container that gathers resources of different types and provides a digital analogue of the 'Materials and Methods' section of a research paper. Workflow Research Objects [5] (WRO) are a specific kind of Research Objects that can be viewed as an aggregation of resources that bundles a workflow specification and additional information to preserve the workflows and their context. Workflow research objects can be used by third parties to understand and run an experiment using the same data inputs used in the original script as well as different ones of her/his choosing.

The methodology is depicted in Figure 1, in which each step corresponds to one requirement. It is composed of five inter-related steps. Given a script S , the first step **Generate abstract workflow** is used to extract from the script an abstract workflow W_a identifying the main processing steps that constitute the data analysis implemented by the script, and their data dependencies. The workflow W_a obtained as a result in Step 1 is abstract in the sense that it cannot be executed.

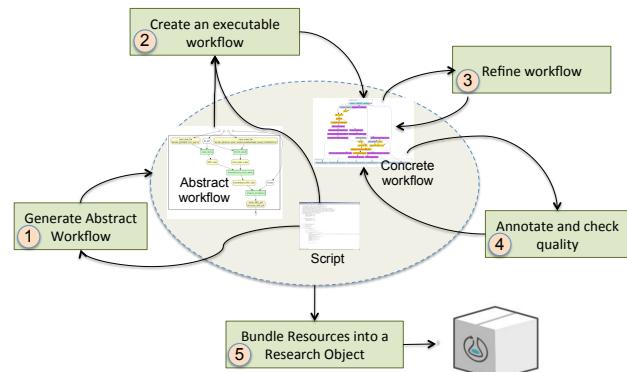


Fig. 1. Methodology for converting scripts into reproducible Workflow Research Objects.

⁴<http://www.ebi.ac.uk>

Given this abstract workflow, the second step **Create an executable workflow** converts the abstract workflow into an executable one W_e by identifying, for each processing step in the abstract workflow, the realization that can be used for its implementation. The executable workflow obtained in step 2 is then **refined** in Step 3 by identifying appropriate third party datasets or processing steps that can, amongst other things, yield better results, generating workflow versions $W_{e1} \dots W_{en}$. For example, the curator may prefer to use human annotated datasets than raw datasets with unknown lineage. In order to help potential users understand the workflow, the curator provides **annotations** describing the workflow, and potentially the resources it utilizes. The curator may also provide examples of provenance traces that have been obtained as a result of the workflow execution. As well as annotating the workflow, the curator should **run a series of checks** to verify the soundness of the workflow. Once tried and tested, the workflow and the auxiliary resources, i.e., annotations, provenance traces, examples of datasets that can be utilized as well as the original script, are **packaged into a Workflow Research Object** (for short, *WRO*) – see section VIII.

We next present the aforementioned steps in detail.

V. GENERATING AN ABSTRACT WORKFLOW

The objective of this phase is to address Requirement 1 by generating an abstract workflow, W_a , given the script S . The generation of W_a entails the analysis of S to identify the main processing units, and their dependencies, that are relevant from the point of view of the scientists, as opposed to a programmer. To do so, we adopt the YesWorkflow tool [1], [4]. It enables scientists to annotate existing scripts with special comments that reveal the computational modules and data flows otherwise implicit in these scripts. YesWorkflow extracts and analyzes these comments, represents the scripts in terms of entities based on the typical scientific workflow model, and provides graphical renderings of this workflow. To the best of our knowledge, YesWorkflow is the only tool that allows generating a graphical representation of a script as a workflow. It does so by processing curator-provided tags of the form $@tag value$, where $@tag$ is a keyword that is recognized by YesWorkflow, and $value$ is an optional value assigned to the tag.

We illustrate the semantics of the tags using Listing 1, which is an excerpt of a script of our use case (see section II) annotated with YesWorkflow tags. We point out that this excerpt shows only annotations, having eliminated most of the code. For the complete code, see⁵, the final WRO. The tags $@begin$ and $@end$ are used to delimit the activities of the workflow, or the workflow itself. The $@begin$ tag is followed by a *name* that identifies the activity in question within the workflow. For example, Listing 1 shows that the overall workflow, named *setup*, is composed of four activities: *split*, *psfgen*, *solvate*, and *ionize*. Those activities represent different parts of the

script. For example, activity *split* corresponds to the code in the script delimited by lines 14 and 27.

The curator can annotate an activity with its description using the $@desc$ tag. An activity may be characterized by a set of input and output ports, defined using the tags $@in$ and $@out$, respectively. For example, activity *split* has one input port named *initial_structure* and three output ports named *protein_pdb*, *bglc_pdb* and *water_pdb*. Note that the names of script variables may not be self explanatory. The curator can associate the input and output ports with more meaningful names using the tag $@as$, which creates an alias name. For example, the output port *gh5_psf* of the *setup* activity (the whole workflow) is associated with the alias *final_structure_psf*. A script may retrieve or store the results used by an input port and generate an output port in a file during the execution. The $@uri$ tag is used in such cases to specify the path of the file within the file system. For example, the output port *protein_pdb* is associated with the URI *protein.pdb*, representing the file where the *split* activity will store the content of the *protein_pdb* output port during the execution.

Just like activities, input and output ports can be annotated with text using the $@desc$ tag. For example, the input port *initial_structure* has a description in line 5. The data dependencies connecting the activities in the workflow are inferred by matching the names of the input and output ports. A data link connecting an output port to an input port is constructed if those ports are associated with the same variable in the script.

Listing 1. Excerpt of an annotated MD script using YesWorkflow tags

```

1  #!/bin/bash
2
3  # @BEGIN setup @DESC setup of a MD simulation
4  # @PARAM directory_path @AS directory
5  # @IN initial_structure @DESC PDB: 8CEL
6  #   @URI file:{directory}/structure.pdb
7  # @IN topology_prot @URI file:top_all22_prot.rtf
8  # @IN topology_carb @URI file:top_all22_prot.rtf
9  # @OUT gh5_psf @AS final_structure_psf
10 #   @URI file:{directory}/gh5.psf
11 # @OUT gh5_pdb @AS final_structure_pdb
12 #   @URI file:{directory}/gh5.pdb
13
14 # @BEGIN split
15 # @IN initial_structure @URI file:structure.pdb
16 # @IN directory_path @AS directory
17 # @OUT protein_pdb @URI file:{directory}/protein.pdb
18 # @OUT bglc_pdb @URI file:{directory}/bglc.pdb
19 # @OUT water_pdb @URI file:{directory}/water.pdb
20 structure = $directory_path"/structure.pdb"
21 protein = $directory_path"/protein.pdb"
22 water = $directory_path"/water.pdb"
23 bglc = $directory_path"/bglc.pdb"
24 egrep -v '(TIP3|BGLC)' $structure > $protein
25 grep TIP3 $structure > $water
26 grep BGLC $structure > $bglc
27 # @END split
28
29 # @BEGIN psfgen @DESC generate the PSF file
30 # @PARAM topology_prot @URI file:top_all22_prot.rtf
31 # @PARAM topology_carb @URI file:top_all36_carb.rtf

```

⁵<http://w3id.org/w2share/s2rwro/>

```

32 # @IN protein_pdb @URI file:protein.pdb
33 # @IN bgc_pdb @URI file:bgc.pdb
34 # @IN water_pdb @URI file:water.pdb
35 # @OUT hyd_pdb @URI file:hyd.pdb
36 # @OUT hyd_psf @URI file:hyd.psf
37
38 ... commands ...
39
40 # @END psfgen
41
42 # @BEGIN solvate
43 # @IN hyd_pdb @URI file:hyd.pdb
44 # @IN hyd_psf @URI file:hyd.psf
45 # @OUT wbox_pdb @URI file:wbox.pdb
46 # @OUT wbox_psf @URI file:wbox.psf
47 echo "
48 package require solvate
49 solvate hyd.psf hyd.pdb -rotate -t 16 -o wbox
50 exit
51 " > solv.tcl
52
53 vmd -dispdev text -e solv.tcl
54 rm solv.tcl
55 # @END solvate
56
57 # @BEGIN ionize
58 # @IN wbox_pdb @URI file:wbox.pdb
59 # @IN wbox_psf @URI file:wbox.psf
60 # @OUT gh5_pdb @AS final_structure_pdb
61 # @URI file:gh5.pdb
62 # @OUT gh5_psf @AS final_structure_psf
63 # @URI file:gh5.psf
64
65 ... commands ...
66
67 # @END ionize
68
69 # @END setup

```

Once the script is annotated, YesWorkflow generates an abstract workflow representation. Figure 2 depicts the abstract workflow generated given the tags provided in Listing 1. It is

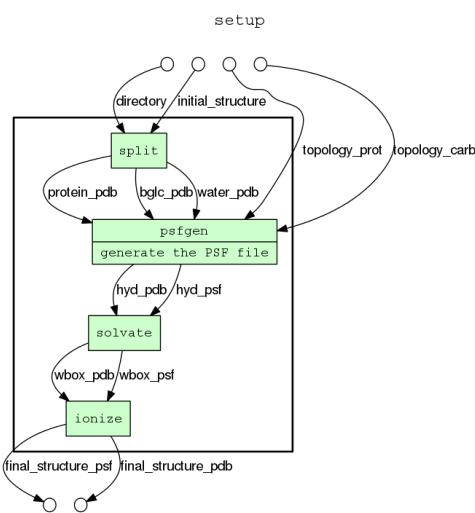


Fig. 2. Abstract workflow representation generated via YesWorkflow.

merely a graphical representation of the script.

Tag recognition is script-language independent, therefore allowing a wide range of script-based experiments to be converted into workflows and consequently a wider adoption of our methodology. The abstract workflow representation is also platform independent. It will be transformed into a platform-specific executable representation in the next step of our methodology.

Furthermore, especially for large, hard-to-read, workflows, we can take advantage of some of the facilities offered by YesWorkflow to help scientists understand a workflow. In particular, YesWorkflow's implementation generates a Datalog file whose facts are constructed from the script tags and follow YesWorkflow's model (e.g., defining that a script is composed of program blocks, ports and channels; or that channels connect ports). We can thus pose Datalog queries against this file to reveal data flow and dependencies within a script. Such queries, as mentioned in [1], can, for instance, allow the user to list the activities defined in the script and their descriptions (when provided by the curator) or the activities that invoke a particular module or external program.

It is worth stressing that the curator needs to respect two constraints when using YesWorkflow in our context. The first constraint concerns **appropriate identification of all processing blocks**. Indeed, YesWorkflow extracts a workflow by processing curator-provided tags; but scientists may not always consider a given piece of script as relevant for tag processing - and thus YesWorkflow will not produce an "appropriate" W_a . However, we are not merely trying to extract an abstract workflow, but to ultimately create an executable workflow that reflects the original script. Thus, the curator annotating the script needs to correctly identify the program blocks that cover the script in its entirety. In other words, taken together, the program blocks that are identified and annotated by the curator need to cover all of the original script.

The second constraint concerns **appropriately tagging all inputs and outputs of each processing block**. In other words, when using YesWorkflow in our context, for each program block identified by the curator, the input and output ports identified and annotated by the curator for that block need to cover all of the inputs needed by that block to be executed, as well as the outputs generated by that block as a result of the execution. Again, in the general case, YesWorkflow does not compel the curator to annotate all the inputs and outputs that are respectively needed and generated by a program block. However, since we are aiming for the creation of an executable workflow, this second constraint needs also to be met.

VI. CREATING AN EXECUTABLE WORKFLOW FROM THE ABSTRACT WORKFLOW

Given the abstract workflow W_a generated previously, the curator needs to create an executable workflow W_e that embodies the data analysis and processes as depicted by W_a – and thus embodies the original script (Requirement 2). Subsequently, the curator may choose to use resources, i.e. datasets and operations, that are different from those used in

the script as s/he sees fit (Requirement 3). Moreover, provenance information identifying how the executable workflow came to be and its relationship with the script need to be recorded (Requirement 4).

A. Step 2: Creating an Initial Executable Workflow

To create the executable workflow W_e , the curator needs to specify for each activity in the abstract workflow, the corresponding concrete activity that implements it.

A simple, yet effective approach to do so consists in exploiting a readily available resource, namely the script code itself.

Given an activity in W_a , the corresponding code in W_e is generated by reusing the chunk (block) of the script that is associated with the abstract workflow activity.

For example, the *split* abstract activity can be implemented by copying the code from the script between the corresponding @begin and @end tags (see lines 20 to 26 in Listing 1); the same would apply to the *solvate* abstract activity (see lines 47 to 54 in Listing 1).

In the implementation of the activity, its input and output ports will be associated with the names of the input and output ports in the abstract workflow. However, they may be different from the corresponding variable names in the script. Therefore it is necessary to check consistency and, when required, change the implementation of the activity, so that the names of the variables are coherent with the port names. To do so, the curator replaces the variable names in the script code with the name used in the tag @as, when defined. This step can be performed in largely automatic fashion. Consider the implementation of the *split* activity, where the *split* program block have a @in tag and an alias name defined via @as. In this implementation, the name of the variable *directory_path* is modified to be *directory*, the name defined via @as (see Listing 2).

Listing 2. Script code - correcting variable name in the implementation of the *split* abstract activity

```

1 structure = %directory%"/structure.pdb"
2 protein = %directory%"/protein.pdb"
3 water = %directory%"/water.pdb"
4 bglc = %directory%"/bglc.pdb"
5 egrep -v '(TIP3|BGLC)' $structure > $protein
6 grep TIP3 $structure > $water
7 grep BGLC $structure > $bglc

```

This approach for conversion comes with two advantages: (i) ease of conversion, since we are using a readily available resource, i.e. the script code, and (ii) the ability to check and debug the execution of W_e against the script execution, to correct eventual mistakes in script-to-workflow conversion.

Once the curator specifies the implementation of each activity in W_a , a concrete workflow specification W_e that is conform to a given scientific workflow system can be created. Without loss of generality, we used the Taverna system [10], although our solution can be adapted to other scientific workflow systems. We chose Taverna as our implementation

platform due to its widespread adoption in several eScience domains and because it supports the script language adopted in our case study.

The workflow curator must be aware of whether the language script is supported by the chosen SWfMS or s/he may assume the risk that the script will not be properly converted into an executable workflow. At this point, the curator will have an executable workflow designed to execute on a specific SWfMS; this workflow can be from now on edited taking advantage of the authoring capabilities of the chosen SWfMS. Figure 3 illustrates the result of this implementation for our case study; it shows a partial MD workflow that was created according to methodology steps 1 and 2, for Taverna.

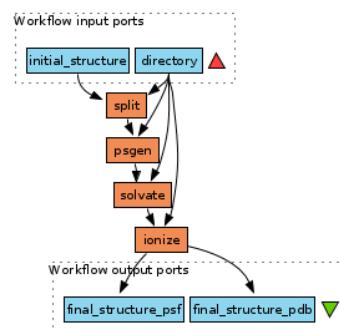


Fig. 3. Partial workflow for an MD script - initial implementation following the first two steps of the methodology.

Once scientists execute this workflow, provenance information regarding execution traces must be collected to serve as input to steps 4 and 5 of our methodology. By executing the workflow, s/he may verify, manually, its results, e.g., checking them against the script results. If this check is not satisfactory, the scientist should identify the problem with help of the execution traces and re-design or re-implement the faulty workflow elements – see more details at section VII.

B. Step 3: Refining the Executable Workflow

Requirement 3 states that the user should be able to modify the workflow to use different computational and data resources. To support this task, a list of available web services and data sets should be shown to the user. For instance, in our case study, scientists' scripts use local data files containing protein coordinates which they download from authoritative web sources. This forces them to download such files from the web, and update them locally whenever they are modified, moreover making them keep track of many file directories, sometimes with redundant information. An example of refinement would be the use of web services to retrieve these files. An even more helpful refinement is, as we did, to reuse workflows that perform this task: we retrieved from the myExperiment repository⁶ a small workflow that fetches a protein structure on Protein Data Bank (PDB) from the RCSB PDB archive⁷. This reused myExperiment workflow was inserted

⁶<http://www.myexperiment.org>

⁷<http://www.rcsb.org/pdb/>

in the beginning of our original workflow, replacing the local PDB file used in the original script (see figure 4).

Here, the *structure_filepath* input parameter of figure 3 was replaced by the sub-workflow within the light blue box, copied from myExperiment workflow repositories.

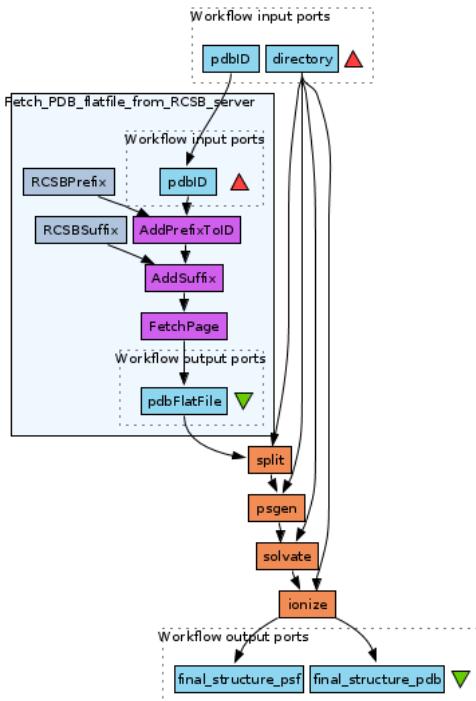


Fig. 4. Workflow refined to use a reusable component that fetches PDB files from the Web.

By the same token, in the life sciences, scientists can invoke web services or reuse data sets listed on portals such as Biocatalogue⁸, which provides a curated catalogue of Web services, and Biotools⁹, which is a tools and data services registry.

C. Recording Provenance Information of the Executable Workflow

Requirement 4 states that provenance information must be recorded to capture the steps performed in the transformation from script to workflow. This transformation is recorded using a provenance model, which allows identifying the correspondence between workflow activiti(es) and script code, and reusable components/web services and script excerpts.

The lineage of versions of the workflow should be stored, as well. It is important to inform to future users that the workflow was curated, and how this curation process occurred.

Listing 3. PROV statements

```

1 @base <http://w3id.org/s2rwro/md-setup/>.
2 @prefix oa: <http://www.w3.org/ns/oa#>.
3 @prefix prov: <http://www.w3.org/ns/prov-o#>.

```

⁸<https://www.biocatalogue.org/>

⁹<https://bio.tools>

```

4 @prefix wfdesc: <http://purl.org/w4ever/wfdesc#>.
5 @prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
6 @prefix wf4ever: <http://purl.org/wf4ever/wf4ever#>.
7
8 <resources/script.sh> a wf4ever:Script, prov:Entity.
9
10 <script/split> a wfdesc:ProcessImplementation;
11     prov:wasDerivedFrom <resources/script.sh>,
12     [
13         a oa:TextPositionSelector;
14         oa:start "674"^^xsd:Integer;
15         oa:end "933"^^xsd:Integer;
16         a prov:Entity
17     ].
18
19 <workflow/we> a wfdesc:Workflow, prov:Entity;
20     prov:wasDerivedFrom <resources/script.sh>;
21     wfdesc:hasSubProcess <workflow/we1/split>.
22
23 <workflow/we/split> a wfdesc:Process;
24     wfdesc:hasImplementation <script/split>.
25
26 <workflow/we1> a wfdesc:Workflow;
27     prov:wasDerivedFrom <workflow/we>;
28     wfdesc:hasSubProcess <workflow/we/split>.

```

Listing 3 shows RDF statements in Turtle syntax wrt the provenance of W_e , the first workflow derived directly from the script S , and the subsequent workflow W_{e1} derived from W_e . Line 8 describes the script resource as a *wf4ever : Script*. To identify the chunk of the script that corresponds to a given (executable) activity in W_e , we utilize the W3C Web Annotation Data Model¹⁰. For example, lines 10 to 17 show that a fragment of the script (delimited using the class *ao : TextPositionSelector* and the position within the script source code) originated the implementation of a process *<script/split>* (as a *wfdesc : ProcessImplementation*). This information was extracted from the program block defined using the YesWorkflow tags *@begin* and *@end*. Lines 19 to 21 show the declaration of W_e ; it was derived from the script and it has a subprocess which was declared in lines 23 and 24. This subprocess (defined as *wfdesc : Process*) is associated with the implementation *<script/split>*. Lines 26 to 28 declared W_{e1} as a derivation of W_e and with a subprocess which is the same one from W_e , identified as *<workflow/we/split>*.

VII. ANNOTATING THE WORKFLOW AND CHECKING ITS QUALITY

It is critical to have a quality check where the scientist explicitly assesses the workflow activities and data flow, comparing them to what was executed by the script.

Throughout the process of workflow creation and modification, the scientist should provide **annotations** describing it (i.e. activities and ports), and potentially the resources it utilizes.

¹⁰<https://www.w3.org/TR/annotation-model>

Part of these annotations can be migrated to the concrete workflow from the YesWorkflow tags - e.g., `@desc` used in the script to describe its program blocks and ports. Most SWfMS, moreover, provide an annotation interface, which can be taken advantage of.

A. Quality dimensions

Quality, here, involves three different quality dimensions¹¹: reproducibility, understandability for reuse, and performance. Reproducibility assesses whether W_e – the first workflow created from the script via conversion of the abstract workflow W_a – and its versions $W_{e1} \dots W_{en}$ reproduce S within some scientist-defined tolerance thresholds. Understandability is promoted with Step 5 (section VIII), by creating Workflow Research Objects with associated annotations. This bundling makes sure that the Research Object is understandable (and thus reusable and reproducible by third parties). Performance concerns the versions $W_{e1} \dots W_{en}$. Assuming that they satisfy the reproducibility criterion, performance provides measurements of the advantages of executing these versions (e.g., faster execution).

These three dimensions make up for a fourth global quality dimension - reliability. In this sense, we state that our methodology ensures reliable results in the transformation of script S into a bundled Workflow Research Object that supports experiment reproducibility, understandability for reuse, and meets performance requirements.

B. Assessing quality of the workflows

Perhaps the main challenge of assessing all the quality dimensions is to define how to compare script and workflow, and the metrics to use to perform this comparison – i.e., how to assess experiment reproducibility.

To check reproducibility, one may compare the script with the workflow code to check if they are equivalent. However, it is known that checking program equivalence is undecidable. Moreover, the refined workflow may use remote programs (e.g., via web services), for which the source code is not available.

A more pragmatic approach to checking reproducibility consists in shepherding the curator in assessing "equivalence", always highly dependent on human expertise. We stipulate that this should be performed in two stages: the first will compare S to W_e , and the second will compare W_e with each of its versions (obtained through refinement), to identify divergences.

a) *Comparing S to W_e :* In more detail, the analysis of differences between S and W_e should be performed in two successive steps: (i) Visual analysis of W_a by the scientist, to check for problems in, e.g., defining activities, or data flow; and (ii) Comparison of execution results, given that W_e uses exactly the same input files as S , and that the script code was copied from S to W_e . Step (i) was mentioned in section VI-A. Step (ii) can be automatic (e.g., for text files, use linux' `diff`),

¹¹A dimension, in quality literature, is a specific quality property that needs to be taken into consideration in assessing quality.

or semi-automatic, combining algorithms and visual checks. Nevertheless, at some point there may be the need to check data semantics; here, annotations (and semantic annotations of data) can help. Our case study basically involves text files (PDB and similar files as inputs, molecular trajectories as output), and thus textual comparison is enough.

In performing these steps, one must keep in mind that comparisons should be done with the help of the human curator. For instance, if the results are different in terms of values, then the curator can be solicited to see if they are scientifically similar or if they are completely different and signal a problem with the workflow.

Common mistakes when converting S to W_e typically include:

- the scientist did not clearly identify the main logical processing units in the script and inserted YesWorkflow tags in the wrong places - in this case, the visualization of the abstract workflow will help identify the problem;
- the scientist made a mistake when migrating script code into the corresponding activity - here, execution traces help since they will show that some data sources are used as inputs to incorrect activities;
- the scientist did not provide the correct input files and parameters - again, traces will help;
- the coding of the workflow itself contained errors - this may be checked with analysis of traces.

Last but not least, additional differences may be introduced by the computing environment itself – e.g., the programming environment used to execute the script wrt the SWfMS environment.

b) *Comparing W_e to its workflow versions:* The rest of the quality check is executed at the same time the scientist improves and/or modifies the workflow through versions (e.g., changing algorithms, or data sets). This comparison can take advantage of the PDIFF algorithm of [7]. PDIFF performs an "equivalence check" of two workflows by comparing the traces of their executions. Trace comparison is based on 4 elements: the workflow graph obtained from the execution trace, the input data, third-party data and processes, and the SWfMS environment each workflow used. Traces become digraphs, and the authors perform comparison of these digraphs to obtain their differences. The specific point(s) of divergence are identified through graph analysis, assisting the workflow user to understand those differences. In our case, we assume that W_e and its versions run in the same SWfMS.

VIII. BUNDLE RESOURCES INTO A WORKFLOW RESEARCH OBJECT

In this step, the curator creates a Workflow Research Object (WRO) that bundles the original script as well as other auxiliary resources obtained in the other steps of the methodology. The creation of the WRO is conducted in parallel with the rest of the methodology steps, in the sense that the curator adds resources to the Workflow Research Object while performing the other steps of the methodology.

The Workflow Research Object model allows curators to aggregate resources and explicitly specify the relationship between these resources and the workflow in a machine-readable format using a suite of ontologies [8].

The result is a WRO that bundles a number of resources that promote the understanding, reproducibility and ultimately the reuse of the workflows obtained through refinement. More specifically, the curator should bundle at least the following resources into the WRO:

- annotated script files (an experiment may involve multiple scripts);
- the workflow W_e (and its versions);
- workflow provenance (documenting the transformation from S to W_e and its versions);
- provenance traces of workflow executions (activities, inputs, outputs, intermediate results);
- research questions and hypotheses;
- output files;

By including these resources, it will be possible for scientists not only to understand how the experiment was conducted, but also its context. Moreover, curators can also bundle additional documents that may help scientists understand the workflow research object, e.g., technical reports and published papers.

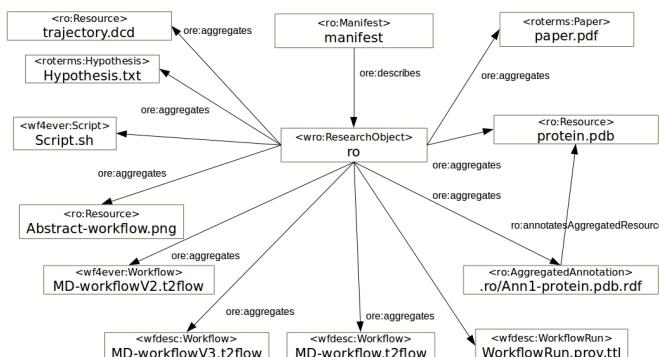


Fig. 5. A graphical example of a WRO bundle derived from our case study.

Figure 5 shows an example of a WRO created for our use case. Arrows denote relationships and boxes denote instances of concepts defined in ontologies. We used the Research Object ontology¹² to define the RDF-based manifest file describing all the resources aggregated in the bundle and to define the relationships (as *ore:aggregates*) with the *wro : ResearchObject* instance. The figure also shows an example of an annotation (*ro : AggregatedAnnotation*), defined in the *.ro/Ann1 – protein.pdb.rdf* file, describing *protein.pdb*. Every file defined in the manifest is a *ro : Resource* and may be also specialized in specific types of resources such as *wfdesc : Workflow*, *wfdesc : WorkflowRun* and *wf4ever : Script*. We used the RO Manager tool¹³ to create the WRO bundle file at the end of our methodological steps.

¹²<http://purl.org/wf4ever/ro>

¹³<https://github.com/wf4ever/ro-manager>

The bundle is available online at <http://w3id.org/w2share/s2rwro/>.

However, it is not enough to create such research objects; they must be made available to the scientific community in a user-friendly manner, so that not only machines, but also scientists can select the most appropriate ones. A possible solution is to make them available by depositing them in a Research Object Portal such as myExperiment and RO Hub¹⁴ which have an interface to search and navigate between resources aggregated in a RO.

IX. RELATED WORK

Part of related work was already discussed in the text, e.g., the work of [7] for workflow equivalence. Here, we present some brief comments on some relevant sources.

Our methodology guides the transformation from script to executable and verifiable workflow. We adopted YesWorkflow [1], [4] to generate the abstract workflow visualizations before creating the executable workflow (step 1 of our methodology). Our choice of YesWorkflow was primarily based on its simplicity of use, script language independence and platform independence, as well as its open code. Moreover, it allows generating a graphical representation of a script as a workflow.

Another (more system-specific) example of the construction of executable workflows from source code is pursued in [11]. It relies on analysis of Abstract Syntax Trees (ASTs) from the source code of Ruby scripts, to convert automatically such scripts into an executable workflow targeted to a specific SWfMS. Our approach differs from this in that we propose a language-independent methodology to assist scientists to convert scripts written in any language into an executable and reproducible workflow.

There are several other tools and systems to create executable workflows. Examples include HyperFlow [12], StarFlow [13] and Swift [14]. These focus on how a declarative language (defining the workflow model) in conjunction with a general-purpose programming language (defining the activities) can be combined to create executable workflows. Our approach differs in the sense that we do not change the way the scripts are developed, and neither is our approach limited to a specific language.

The work of [15] proposes an executable visual-based representation of a workflow. This was extended by [16] to allow scientists two alternative ways of working with workflows: script-based and visual-based representations. A two-way representation translator enables the conversion between representations; workflow execution uses a single enactor, independent from the users' preferred representation. [16] argues that, in some cases, scripts are preferable to specify workflows since scientists may want to look at code. We, instead, go the opposite way, given the need for reusability by third parties: we adopted a tool and a language and platform-independent approach to transform scripts into workflow research objects.

¹⁴<http://www.rohub.org/>

Another important aspect of our work concerns assessment of quality with respect to reproducibility, reuse and understandability. Our preliminary work towards this goal is based on [7], and their PDIFF algorithm that compares workflow traces, produced by their SWFMS environment, e-Science Central. Their framework uses as input the two provenance digraphs, and produces as output the difference graph, in which nodes may represent differences in data sources or outputs, or differences in activities. They also provide algorithms that compute the equivalence of three classes of data: text, XML and models. For the purposes of comparing XML documents, they use XOM¹⁵. To calculate the similarity of mathematical models, they use the Analysis of Covariance test that analyses the predictive performance of two models. Yet another possibility to compare workflows appears in [17]. Here, the technique used is based in detecting plagiarism in text. Though interesting, this is too generic for our goals.

X. CONCLUSIONS AND ONGOING WORK

This paper presented a methodology that guides curators in a principled manner to transform scripts into reproducible and reusable workflow research objects. This addresses an important issue in the area of script provenance – that of providing an executable and understandable provenance representation of domain script runs. The methodology was elaborated based on requirements that we elicited given our experience and collaborations with scientists who use scripts in their data analysis. The methodology was showcased via a real world use case from the field of Molecular Dynamics.

Our ongoing work includes the evaluation of our methodology using other use cases, from fields other than molecular dynamics. We are also considering the problem of synchronizing script changes to updates on the corresponding workflow research objects. Moreover, we are investigating extending YesWorkflow to support the semantic annotation of blocks and using concepts from ontologies and vocabularies, and to support workflow nesting, which is currently not supported by YesWorkflow. Last but not least, there is a need to evaluate the cost of the effectiveness of our proposal, in particular since in some cases it may require extensive involvement of scientists and curators.

ACKNOWLEDGMENTS

Work partially financed by FAPESP (2014/23861-4), FAPESP/CEPID CCES (2013/08293-7), FAPESP-PRONEX (eScience project), INCT in Web Science, and individual grants from CNPq. We thank Prof. Munir Skaf and his group from the Institute of Chemistry at Unicamp for making their scripts and data available and for their valuable feedback.

REFERENCES

- [1] T. McPhillips, T. Song, T. Kolisnik, S. Aulenbach, K. Belhajjame, R. K. Bocinsky, Y. Cao, J. Cheney, F. Chirigati, S. Dey *et al.*, “Yesworkflow: A user-oriented, language-independent tool for recovering workflow information from scripts,” *International Journal of Digital Curation*, vol. 10, no. 1, pp. 298–313, 2015.
- [2] S. Dey, K. Belhajjame, D. Koop, M. Raul, and B. Ludäscher, “Linking prospective and retrospective provenance in scripts,” in *7th USENIX Workshop on the Theory and Practice of Provenance (TaPP 15)*, 2015.
- [3] L. Murta, V. Braganholo, F. Chirigati, D. Koop, and J. Freire, “noworkflow: Capturing and analyzing provenance of scripts,” in *Provenance and Annotation of Data and Processes*. Springer, 2014, pp. 71–83.
- [4] T. McPhillips, S. Bowers, K. Belhajjame, and B. Ludäscher, “Retrospective provenance without a runtime provenance recorder,” in *7th USENIX Workshop on the Theory and Practice of Provenance (TaPP 15)*, 2015.
- [5] K. Belhajjame, O. Corcho, D. Garijo, J. Zhao, P. Missier, D. Newman, R. Palma, S. Bechhofer, E. García Cuesta, J. M. Gómez-Pérez *et al.*, “Workflow-centric research objects: First class citizens in scholarly discourse,” in *Proceedings of Workshop on the Semantic Publishing, (SePublica 2012)*, 2012.
- [6] R. L. Silveira and M. S. Skaf, “Molecular dynamics simulations of family 7 cellobiohydrolase mutants aimed at reducing product inhibition,” *The Journal of Physical Chemistry B*, vol. 119, no. 29, pp. 9295–9303, 2014.
- [7] P. Missier, S. Woodman, H. Hiden, and P. Watson, “Provenance and data differencing for workflow reproducibility analysis,” *Concurrency and Computation: Practice and Experience*, vol. 28, no. 4, pp. 995–1015, 2016.
- [8] K. Belhajjame, J. Zhao, D. Garijo, M. Gamble, K. Hettne, R. Palma, E. Mina, O. Corcho, J. M. Gómez-Pérez, S. Bechhofer *et al.*, “Using a suite of ontologies for preserving workflow-centric research objects,” *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 32, pp. 16–42, 2015.
- [9] S. Bechhofer, D. De Roure, M. Gamble, C. Goble, and I. Buchan, “Research objects: Towards exchange and reuse of digital knowledge,” 2010.
- [10] K. Wolstencroft, R. Haines, D. Fellows, A. Williams, D. Withers, S. Owen, S. Soiland-Reyes, I. Dunlop, A. Nenadic, P. Fisher, J. Bhagat, K. Belhajjame, F. Bacall, A. Hardisty, A. Nieva de la Hidalga, M. P. Balcazar Vargas, S. Sufi, and C. Goble, “The Taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud,” *Nucleic Acids Research*, vol. 41, no. W1, pp. W557–W561, 2013.
- [11] M. Baranowski, A. Belloum, M. Bubak, and M. Malawski, “Constructing workflows from script applications,” *Scientific Programming*, vol. 20, no. 4, pp. 359–377, 2012.
- [12] B. Balis, “Increasing scientific workflow programming productivity with hyperflow,” in *Proceedings of the 9th Workshop on Workflows in Support of Large-Scale Science*. IEEE Press, 2014, pp. 59–69.
- [13] E. Angelino, D. Yamins, and M. Seltzer, “Starflow: A script-centric data analysis environment,” in *Provenance and Annotation of Data and Processes*. Springer, 2010, pp. 236–250.
- [14] M. Wilde, M. Hategan, J. M. Wozniak, B. Clifford, D. S. Katz, and I. Foster, “Swift: A language for distributed parallel scripting,” *Parallel Computing*, vol. 37, no. 9, pp. 633–652, 2011.
- [15] J. Montagnat, B. Isnard, T. Glatard, K. Maheshwari, and M. B. Fornarino, “A data-driven workflow language for grids based on array programming principles,” in *Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science*. ACM, 2009, p. 7.
- [16] K. Maheshwari and J. Montagnat, “Scientific workflow development using both visual and script-based representation,” in *6th World Congress on Services (SERVICES-1)*. IEEE, 2010, pp. 328–335.
- [17] D. d. O. Filipe Tadeu Santiago, “Verificação da Reprodução de Workflows Científicos por meio de Algoritmos de Detecção de Plágio (in Portuguese),” in *X Brazilian e-Science Workshop (BRESCI 2016)*. Sociedade Brasileira de Computação, 2016.

¹⁵<http://xom.nu>

A Framework for Scientific Workflow Reproducibility in the Cloud

Rawaa Qasha
 Newcastle University
 Newcastle upon Tyne, UK
 Mosul University, Iraq
 Email: r.qasha@newcastle.ac.uk

Jacek Cala
 Newcastle University
 Newcastle upon Tyne, UK
 Email: jacek.cala@newcastle.ac.uk

Paul Watson
 Newcastle University
 Newcastle upon Tyne, UK
 Email: paul.watson@newcastle.ac.uk

Abstract—Workflow is a well-established means by which to capture scientific methods in an abstract graph of interrelated processing tasks. The reproducibility of scientific workflows is therefore fundamental to reproducible e-Science. However, the ability to record all the required details so as to make a workflow fully reproducible is a long-standing problem that is very difficult to solve.

In this paper, we introduce an approach that integrates system description, source control, container management and automatic deployment techniques to facilitate workflow reproducibility. We have developed a framework that leverages this integration to support workflow execution, re-execution and reproducibility in the cloud and in a personal computing environment.

We demonstrate the effectiveness of our approach by examining various aspects of repeatability and reproducibility on real scientific workflows. The framework allows workflow and task images to be captured automatically, which improves not only repeatability but also runtime performance. It also gives workflows portability across different cloud environments. Finally, the framework can also track changes in the development of tasks and workflows to protect them from unintentional failures.

I. INTRODUCTION

Workflows have become a valuable mechanism for specifying and automating scientific experiments running on distributed computing infrastructure. Researchers in different disciplines have embraced them to conduct a wide range of analyses and scientific pipelines [1], mainly because a workflow can be considered as a model defining the structure of the computational and/or data processing tasks necessary for the management of a scientific process [2].

However, workflows are not only useful in representing and managing the computation but also as a way of sharing knowledge and experimental methods. When shared, they can help users to understand the overall experiment, or they can become an essential building block in their new experiments. Lastly, workflows can also be used to repeat or reproduce the experiment and replicate the original results [3].

One of the major challenges in achieving workflow reproducibility, however, is the heterogeneity of workflow components which demand different, sometimes conflicting sets of dependencies. Ensuring successful reproducibility of workflows requires more than simply sharing their specifications. It also depends on the ability to isolate necessary and suffi-

cient computational artifacts and preserve them with adequate description for future re-use [4].

A number of analyses and research efforts have already been conducted to determine the salient issues and challenges in workflow reproducibility [5], [6], [7], [8]. In short the issues can be summarized as: insufficient and non-portable description of a workflow including missing details of the processing tools and execution environment, unavailable execution environments, missing third party resources and data, and reliance on external dependencies, such as external web services, which add difficulty to reproducibility at a later time.

Currently, most of the approaches that address reproducibility of scientific workflows have focused either on their *physical preservation*, in which a workflow is conserved by packaging all of its components, so an identical replica is created and can be reused; or on *logical preservation*, in which the workflow and its components are described with enough information for others to reproduce a similar workflow in the future [9].

Although both, packaging and describing, play a vital role in supporting workflow re-use, alone they are not sufficient to effectively maintain reproducibility. On the one hand physical preservation is limited to recreating the packaged components and resources while it lacks a structured description about the workflow. Thus, it makes easy to *repeat* exactly the same execution, yet it is often not enough to *reproduce* the experiment with different parameters or input data. On the other hand logical preservation can provide detailed description of various levels of the workflow. It is still not enough, however, in the absence of the necessary tools and dependencies.

A need to integrate these two forms of preservation becomes increasingly apparent. That, combined with a portable description of the workflow, which can be used in different environments, and an automated workflow deployment mechanism has potential to significantly improve workflow reproducibility.

In this paper we present a framework designed to address the challenges mentioned earlier. The framework integrates features of both logical and physical preservation approaches. Firstly, using OASIS specification “Topology and Orchestration Specification for Cloud Applications” (TOSCA) [10] it allows a workflow description to include the top-level structure of the abstract workflow together with details about its execution environment. The description is portable and may

be used in automated deployment across different execution environments including the Cloud and a local VM.

Secondly, using Docker virtualisation and imaging our framework offers portable packaging of whole workflows and their parts. By integration with TOSCA, the packaging is automated, hence users are free from creating and managing Docker images. Additionally, our framework is built upon code repositories that natively support version control – crucial in tracking the evolution of workflows and their components over time.

We argue that these three elements: portable and comprehensive description, portable packaging and widely applied version control, play a fundamental role in maintaining reproducibility of scientific workflows over longer periods of time. They allowed us to build the framework which we present as the main contribution of this paper. We evaluate the framework using real scientific workflows developed in our previous projects to demonstrate that it can effectively realise its goal.

II. BACKGROUND AND RELATED WORK

Workflow reproducibility and repeatability have been discussed in a number of studies, such as [5], [11], and are considered to be an essential part of the computational scientific method. As our approach to improving reproducibility of workflows is based on the TOSCA specification and Docker technology, in this section we present the three relevant areas.

A. Scientific Workflow Reproducibility

There have been various attempts proposed in the literature or as software tools to address repeatability and reproducibility of scientific workflows. As mentioned earlier, most of them follow one of two directions: (1) packaging the components of a workflow, known as physical preservation/conservation, or (2) describing a workflow and all its components, called logical preservation/conservation.

To implement packaging of workflows Chirigati et al. proposed ReproZip [12]. It tracks system calls during the execution of a workflow to capture the dependencies, data and configuration used at runtime, and to package them all together. Then the package can be used to re-execute the archived workflow invocation.

Other researcher to package workflows have used virtualization mechanisms, specifically the ability to save the state of a virtual machine as an image (VMI) [13], [14], [15]. The main advantage in using VMIs is that they allow the complete experimental workflow and environment to be easily captured and shared with other scientists [16]. However, the resulting images are large in size and costly to be publicly distributed [17]. And despite packaging mechanisms allowing workflows to be re-executed (i.e. allow repeatability), they usually do not convey a detailed and structured description of the entire computation, relevant dependencies and execution environments, which would help in understanding the package contents. Therefore, their ability to reproduce or even reuse a

packaged workflow in other contexts (e.g. using different input data, parameters or execution environments) is often limited.

The logical preservation techniques focus on capturing all the details required to repeat and potentially reproduce scientific workflows. A notable example is myExperiment [18] which offers a web interface to support social sharing of workflows with computational description and visualizations of their components. myExperiment, as a general repository for workflows, contributes to the improvement of workflow reproducibility.

Santana-Perez et al. proposed in [9] a semantic-based approach to preserve workflows with their execution environment. They use a set of semantic vocabularies to specify the resources involved in the execution of a workflow. However, other studies have shown that sharing only the specifications of a workflow is not enough to ensure successful reproducibility [19].

Another technique of logical preservation is capturing the provenance information of the workflow results [20], [21]. Retrospective provenance encapsulates the exact trace of a past workflow execution, which can then help in its re-execution. Nevertheless, provenance usually describes only the abstract layer of a workflow because detailed traces of the use of execution environment (e.g. at the OS level) quickly become overwhelming.

More recently, Hasham et al. [22] presented a framework that captures information about Cloud infrastructure of a workflow execution and interlinks it with data provenance of the workflow. They propose workflow reproducibility by re-provisioning similar execution infrastructure using the Cloud provenance and then re-execution of the workflow. Although the approach enables the re-execution, it is unable to track and address changes to the original workflow.

Belhajjame et al. [19] proposed Research Objects as a preservation approach for scientific workflows. Research Objects can aggregate various types of data to enhance workflow reproducibility like: workflow specifications, description of workflow components and provenance traces. However, they do not include enough technical details about dependencies and the workflow execution environment to easily allow re-enactment.

The specification-based mechanisms provide various details that can help in understanding the workflow and its components. Yet, they are still insufficient when some of the required dependencies change or become unavailable, in which case the ability to reconstruct the same execution environment is lost. Therefore, the integration of workflow specification and description of its components alongside a portable packaging mechanism that facilitates sharing becomes fundamental.

B. Topology and Orchestration Specification for Cloud Applications

TOSCA is an OASIS specification for modeling a complete application stack, and automating its deployment and management in the Cloud. The main intent of TOSCA is to improve

the portability of Cloud applications in the face of the growing diversity of Cloud environments [23].

The specification defines a meta-model for describing both the structure and management of IT applications. The structure of an application, its components and the relationships between them are represented by a *Topology Template*. The components and their relationships are defined as *Node* and *Relationship Templates* instantiated from a predefined *Node* and *Relationship Types*. The types are reusable entities that can be used to construct new Topology Templates for different applications [24].

In our previous work [25] we proposed the use of TOSCA to describe the entire structure of a scientific workflow, together with all its components and specification of a host environment. By adopting TOSCA, we can turn workflows into reusable entities that include not only the description of a scientific experiment but also all details needed to deploy and execute them automatically. Therefore, we use TOSCA as the basis for the framework presented in this paper.

C. Reproducibility using Lightweight Virtualization

Container-based virtualization is a lightweight alternative to Virtual Machines. It is not new but Docker,¹ one of the recently developed tools for Linux systems, established a strong and open ecosystem that several Cloud providers support and promote in their offers. Importantly, Docker containers are portable and can run on different hosts, which makes them a suitable packaging tool to support the reproducibility of applications [26].

Similarly to a VMI, a Docker image is a file that includes an Operating System together with a set of relevant libraries, software packages, configuration and data. And it can later be used to create a container – a running instance of the system/application. That makes containers equally suitable to encapsulate and then re-execute scientific workflows. But the main attraction in using containers, when compared to Virtual Machines, is that images are smaller in size and starting a container is a few orders of magnitude faster than starting an VM. Therefore, in our work we integrated Docker images and containers in the deployment and reproducibly process.

Similarly to Virtual Machine hypervisors, Docker allows workflow applications along with all necessary dependencies to be encapsulated into a container image [27], [28]. But even if these approaches can offer a convenient mechanism to preserve workflows, they still lack a structured description of the aggregate. In addition, they are limited to packaged resources and dependencies, and lack flexibility to change the components or dependencies in an already packaged workflow.

III. IMPROVING WORKFLOW REPRODUCIBILITY

Clearly, the complete reproducibility of a workflow is hard to achieve due to possible changes at various levels of software and hardware platforms used to run it. We can, however, significantly increase the degree of reproducibility by addressing

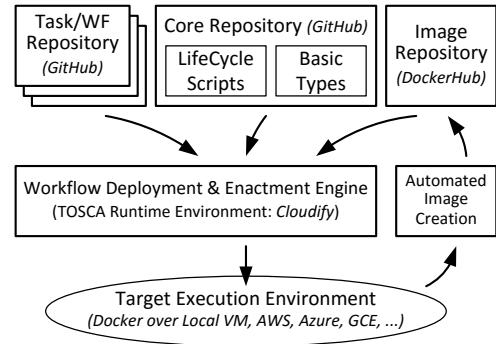


Fig. 1: The architecture of our workflow reproducibility framework.

the challenges discussed earlier. And our goal when designing our workflow reproducibility framework was to find ways in which we can effectively respond to these challenges.

A. The Framework Architecture

The proposed workflow reproducibility framework consists of four main components: the Core repository, a set of Workflow and Task repositories, the Image repository supported by the Automatic Image Creation (AIC) facility, and the workflow enactment engine (Fig. 1). The Core repository includes a set of common and reusable TOSCA elements such as **Node**- and **RelationshipTypes**, and life cycle management scripts. They are a foundation for building tasks and workflows. The Workflow and Task repositories are used to store workflows and their components so they can be accessed during enactment and also shared and reused in designing new workflows. The Image repository contains workflow and task images that are used to improve reproducibility and also performance of workflow enactment. Images are automatically captured by the AIC. Finally, the workflow enactment engine is implemented by a TOSCA-compliant runtime environment.

To implement logical preservation we rely on the TOSCA specification which we previously adopted as a method to model portable workflows [25]. With TOSCA we can describe workflows not only at the abstract level but together with the complete software stack required to deploy and enact them. And it is portable because we can use a TOSCA-compliant runtime environment to automatically deploy and enact our workflows on different Cloud platforms or in a local VM.

To control changes that can affect a workflow and its components we use a version control platform. It gives us the ability to track the complete history of developmental changes of workflows and tasks. The version control platform supports also the Automatic Image Creation facility. The AIC uses Docker to implement physical preservation of workflows and greatly helps in building and management of image libraries.

Moreover, instead of building yet another workflow repository and yet another workflow engine, we define our framework on top of open platforms like GitHub and DockerHub. The former allows workflow and task source code to be stored

¹<https://www.docker.com/>

and maintained under version control, the latter can store workflows and tasks packaged as Docker images. Importantly, both platforms offer mechanisms which promote sharing and reuse.

B. The Framework in Use

To create a workflow the user needs to implement and model its essential components including: **NodeTypes** and tasks code. The **NodeTypes** are used to declare tasks and dependency libraries, they also refer to the task code – the actual software artifacts which will be deployed and executed.

Currently, to facilitate building new tasks and workflows we implement a set of basic **NodeTypes** and tasks which others can reuse. Additionally, our Core repository provides **RelationshipTypes** and life cycle management scripts that are common to all workflows. They define and implement basic workflow functionality like passing data between tasks, configuration of library dependencies, etc. Given all these components, the workflow can be encoded as a TOSCA **ServiceTemplate**. The template includes **Node-** and **RelationshipTemplates** that are instances of the types developed earlier; these templates represent tasks and task links, respectively.

Once the workflow **ServiceTemplate** has been prepared it can be deployed by a TOSCA-compliant runtime environment. Currently, we support Cloudify² but there are other options available like OpenTOSCA³ and Alien4Cloud.⁴ The enactment of workflows follows the structure embedded in the **TopologyTemplate**, a part of the **ServiceTemplate** that in a declarative way combines components and dependencies. Using the **TopologyTemplate** the runtime environment is able to infer the appropriate workflow execution plan.

IV. WORKFLOW AND TASK REPOSITORIES

Since we have been using publicly available platforms like GitHub to maintain the Workflow and Task repositories, these repositories can remain under users' control. We provide our own repositories with a set of basic reusable workflow tasks and example workflows mainly to illustrate how the framework can support reproducibility. But primarily, the ecosystem of workflows and tasks will be grown by researchers and scientists who want to develop their own workflow applications.

The choice of source version control platforms, such as GitHub, to host repositories of workflows and tasks was not accidental. These platforms offer great tools to support sharing and communication. But more importantly, they allow code developers and users to keep track of the developmental changes, and that can directly help to improve repeatability and reproducibility.

Our approach works on the principle that each single workflow and workflow task is maintained in a separate code repository. That brings multiple benefits: repositories mark clear boundaries between components, they offer independent

version control, allow for easy referencing and sharing, and additionally, provide branches and tags to implement strict control of workflow and task interface. With multiple repositories it's also easy to encapsulate auxiliary information, such as sample data and human readable description specific to each workflow and task, which help to maintain long-term reproducibility.

A. Repository Structure

A repository aggregates various artifacts with information and resources related to the workflow or task. These artifacts include: TOSCA-based descriptors, workflow/task-specific life cycle scripts, sample data, human readable description and the *one-click* deployment script. The key and mandatory artifact is a TOSCA-based descriptor. In the case of a workflow, it is a **ServiceTemplate** descriptor that encodes the structure of a workflow and references all the workflow components and life cycle scripts required for enactment. In the case of a task, the descriptor includes TOSCA **NodeType** that defines the task interface and refers to the actual task implementation code.

Other artifacts, although optional, are helpful to maintain reproducibility. For example, provided with sample data and the one-click deployment script users can easily test a workflow or task in their environment. The script starts a multi-step process which deploys the workflow together with basic dependencies such as Docker and Cloudify and then enacts it. Moreover, given a human readable description stored in a repository, users can better understand the purpose of the component and more easily use it. That also helps to recover from failures in the face of changes in the workflow or any of its dependencies.

The structures of a workflow and task repository are very similar to each other. This is because our tasks also include a simple test workflow descriptor and sample data which allow users to easily run a task and test whether it actually meets their requirements.

Usually, our repositories include two workflow descriptors that define the *single-* and *multi-container* configuration. The single-container workflows are executed within one Docker container, whereas in the multi-container configuration *each* task runs in its own container. The use of the single- or multi-container configuration has also impact on the kind of images that will be generated by our Automatic Image Capture facility. We discuss this aspect later in Section VI.

These two default configurations describe, however, only two extremes out of the range of possible workflow deployments. For more specific, advanced scenarios developers can create workflows that include containers which group together a subset of tasks, for example due to security reasons.

B. Interface Control via Branches and Tags

One of the major sources of workflow decay is changes in the components that a workflow is comprised of. In a living system changes are inevitable because the components – tasks, libraries and other dependency workflows – undergo continuous development. Yet to maintain reproducibility we

²<http://getcloudify.org>

³<https://github.com/OpenTOSCA>

⁴<http://alien4cloud.org>

cannot forbid changes at all. Instead, we need to control them, so they do not contribute to the decay.

The changes that occur naturally during workflow and task development can affect two layers: the interface and/or implementation of a component. By the workflow/task interface we consider the contract between the developer and user of a component. Specifically, it is the number and type of input data and properties that the workflow/task uses in processing but also the number and type of output data it produces.

Changes in the interface usually indicate some important modification to a component and need to be followed by changes in its implementation. Conversely, changes to the implementation only, if made carefully, are often merely improvements in the code which can remain unnoticed.

Since in our framework each component has been maintained in a separate repository, we can control these two types of changes effectively. We use repository branches to denote changes in the interface, and tags to indicate significant improvements in the implementation. Minor implementation changes are simple commit events in the repository which do not need any special attention. All that, supported by an effective way to reference a specific branch or tag offered by GitHub are enough to address the problem of changing components.

However, these mechanisms are not only important for our framework to maintain reproducibility of existing workflows but they are also crucial for users in creating new workflows. With repository branches users can easily see different flavours of a specific task or workflow and decide which one to use. On the other hand tags help users to see major improvements of a component or workflow. Tags also indicate to our framework when there is a need to create a new component image.

To illustrate the use of branching and tagging in practice we show later, in the Evaluation section, a development scenario of one of our test workflows.

V. AUTOMATED WORKFLOW DEPLOYMENT AND ENACTMENT

The model of describing workflows using TOSCA proposed in our previous work [25] is important because it not only supports logical preservation but also offers the ability to automatically deploy and enact our workflows. That facilitates repeatability and improves workflow reproducibility.

Currently, as a workflow engine we use Cloudify – a TOSCA-compliant runtime environment. To run a workflow, users need to clone its repository to a target machine in which they are going to run it. The repository includes sample data and the *one-click* deployment script. It is a simple script able to install the software stack required to run the workflow (Cloudify, Docker and some auxiliary tools) and then to submit the workflow to Cloudify with default configuration parameters.

The default configuration and sample data allow users to easily test the workflow. It is also a means to repeat the execution as well as a starting point to reproduce it. To repeat a workflow users can simply switch to a very specific version

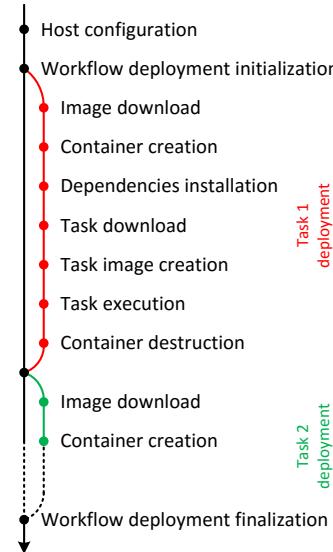


Fig. 2: Steps in automatic workflow deployment using the multi-container configuration.

of the workflow in the repository history and run the *one-click* deployment script. Then, they can modify the default configuration and provide their own data. They can also switch to the latest version of the workflow to validate the output or compare it with output generated by previous versions.

The TOSCA descriptor of a workflow is a declarative specification that includes all tasks, dependency libraries and task links embedded in the workflow ServiceTemplate. The template includes also dependency against the task execution environment which may be composed of one or more Docker containers and VMs. Apart from the declaration of tasks and libraries, the workflow ServiceTemplate encodes also the topology of the workflow. For scientific workflows, usually implemented as directed acyclic graphs, it is enough information so a linear workflow execution plan can be automatically inferred (Fig. 2). Cloudify follows the generated plan, and deploys and runs one task at a time.

Crucial to workflow enactment are life cycle management scripts. They implement deployment operations that each workflow and task needs to go through, such as: initialization of a shared space used to exchange data between tasks, provisioning of the host environment (a container), installation and configuration of library dependencies. As the majority of tasks follow a very similar pattern of deployment, we developed a set of common, reusable life cycle scripts and included them in the Core repository. Developers would refer to these scripts when building their own workflows and tasks.

VI. AUTOMATIC IMAGE CAPTURE

TOSCA-based descriptors are the fundamental element of our framework, partly because they are used to implement logical preservation, and partly because they allow workflows to be automatically deployed and enacted. But running workflows based only on these descriptors would end with significant

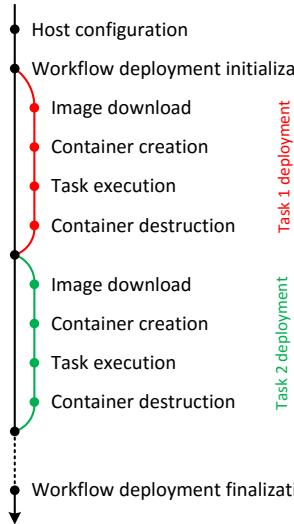


Fig. 3: Steps in automatic workflow deployment using the task images created by the AIC; cf. Fig. 2.

runtime overheads. The framework would repeat the same, sometimes long running, steps to deploy a task every time it was executed.

However, our framework is flexible enough to run the workflow and task deployment process using a variety of Docker images – starting from a pure OS image commonly available from DockerHub, to a specific user-defined image which includes some workflow/task dependencies, to a complete image that contains all of the required dependencies. If the image referred to in the workflow `ServiceTemplate` does not contain all the dependencies, they will be installed by the framework on-demand during workflow enactment. That automation simplifies the development cycle because users are not forced to manually prepare and manage task or workflow images before they can use a workflow.

Yet, to simplify the use of the framework even more we implemented the Automatic Image Capture facility. Using the Docker image manipulation operations, the AIC is able to create workflow and task images for the user automatically, so they can be deposited in a private or public Image Repository. Next time when a task is executed, instead of the complete deployment cycle, the framework will use the images captured earlier (Fig. 3). As shown later in the Evaluation section, that simplification can have very positive impact on runtime performance.

The workflows we implemented are usually described with two configuration options: single- and multi-container. That influences the way in which deployment and enactment of workflows is performed. But it also determines what image the AIC will create for the workflow. If the workflow uses the single-container configuration, the AIC will capture a single image that encapsulates the whole workflow with all its components. Conversely, if the workflow uses the multi-container configuration, many smaller task images will be created. Both options have their advantages: the former imposes

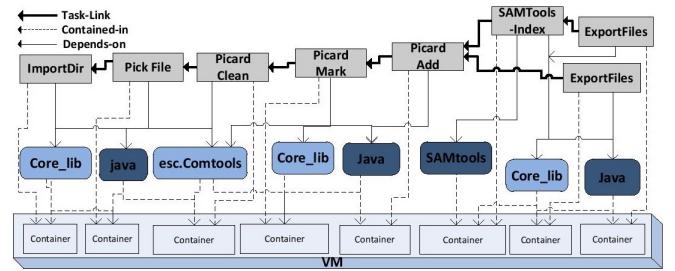


Fig. 4: The structure of the Sequence Cleaning workflow in multi-container configuration described in TOSCA.

less overhead in terms of storage and performance, whereas the latter promotes better reuse of task images and gives more flexibility if the workflow requires updates. Nonetheless, they support repeatability and reproducibility of workflows equally well.

Yet, to realise that goal images must be properly versioned. The AIC uses identifiers from the Image Repository and tags from the Workflow and Task Repositories to address this aspect. The workflow/task image identifier is generated based on the base Docker image identifier and the URL of a branch or tag of a workflow/task for which the image is built. That simple and *unique* mapping between code and image versions allows users to include only the *code* URL in their workflow `ServiceTemplate`, which is enough for the framework to fetch and use the correct image for a task or workflow. And in the case that the image does not yet exist, the workflow enactment will follow the full deployment cycle while the AIC will generate and deposit the relevant images for future use.

VII. EVALUATION AND DISCUSSION

We describe the evaluation of our framework from three different angles. First, we present a set of experiments to show portability of the workflow description, so it can be enacted in different environments. Second, we show the benefit of using the AIC to reduce a workflow’s runtime. Finally, we describe a scenario of workflow and task development to illustrate how the framework can maintain reproducibility in the face of component changes.

A. Repeatability on Different Clouds

The goal of this set of experiments was to re-enact a workflow, initially designed in a local development environment, on three different Clouds and a local VM. We ran the experiment for four different workflows which were previously designed in e-Science Central.⁵ The workflows: *Neighbor Joining* (NJ), *Sequence Cleaning* (SC), *Column Invert* (CI) and *File Zip* (FZ) are different in terms of structure, dependency libraries they require and the number of tasks they include (11, 8, 7 and 3 tasks, respectively). As an example, Fig. 4 depicts the structure of the Sequence Cleaning workflow used in a NGS pipeline [29] and re-implemented using TOSCA.

⁵<http://www.esciencecentral.co.uk>

TABLE I: Basic details about the execution environments.

Environment	CPU Cores	RAM [GiB]	Disk space [GB]	Operating System
Local VM	1	3	13	Ubuntu 14.04
Amazon EC2	1	1	8	Ubuntu Srv 14.04
Google Cloud	1	3.75	10	Ubuntu Srv 14.04
Microsoft Azure	1	3.5	7	Ubuntu Srv 14.04

TABLE II: The average execution time (in minutes) for different workflows executed in different environments.

	Neighbour Join.		Column Invert		File Zip	
	Single	Multi	Single	Multi	Single	Multi
Devel. Env.	2.13	2.54	0.9	1.3	0.6	0.94
Amazon	1.74	2.27	0.66	1.18	0.5	0.84
Azure	2.52	3.86	1.35	2.1	1.23	1.38
Google	1.52	2.48	0.74	1.18	0.5	1.01
Local VM	1.65	2.5	1.03	1.37	0.53	1.03

To illustrate the potential of our framework in supporting repeatability and reproducibility and the value of the proposed workflow representation, each of the selected workflows was first developed, and then deployed and enacted in a local development environment. We recorded the execution time of that initial enactment which also automatically created a workflow or task Docker images.

To conduct the rest of the experiment, we cloned the workflow repositories in four different environments: a local VM, and Amazon AWS, Google Engine and Microsoft Azure Clouds. Finally, we re-executed workflows five times in each VM and collected results. The configuration of the VMs is presented in Table I.

Each workflow was used in two available configurations: single- and multi-container to show the overheads of running multiple task containers. The output data of the workflows were the same in all executions and the average execution times were similar. Fig. 5 shows a chart with the results for the SC workflow, whereas Table II includes the results for the other tested workflows.

The experimental results show that our scientific workflows can be re-enacted, producing the same outputs in similar runtime. They also illustrate a common development pattern in which developers build and test a workflow in their local environment and once it is ready they can share it with others via Workflow, Task and Image Repositories. Both the TOSCA representation and Docker packaging offer significant support for this pattern.

B. Automatic Image Capture for Improved Performance

As mentioned earlier, our framework is flexible enough to allow tasks and workflows to use pure OS images available from DockerHub or custom, predefined task/workflow images created by users or the AIC. By using a predefined image we can avoid the installation of dependency libraries and task artifacts required during workflow execution. And, as shown

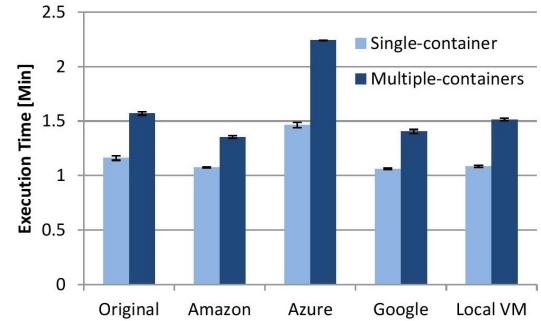


Fig. 5: The average execution time for Sequence Cleaning workflow executed in different environments.

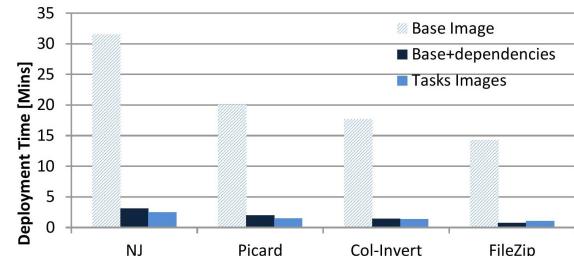


Fig. 6: The average execution time of test workflows using different task images.

previously in Fig. 3, that can reduce the number of deployment steps required in workflow enactment.

However, the elimination of some of the deployment tasks can have very positive impact on the runtime of workflows. To show it we prepared a set of experiments in which we ran our workflows using different images: the base image available on DockerHub, the base image with pre-installed dependency libraries and task images captured by the AIC. Fig. 6 depicts the average workflow execution time for four tested workflows.

Clearly, there was a significant overhead in using the base image from DockerHub. The main reason was the time required to install dependency libraries such as the Java Runtime Environment or, in the case of the NJ workflow, the Wine library.

The second and third option show small differences with slightly shorter execution for experiments which used images created by the AIC. That is because the AIC captures everything the task needs to run (according to the task's TOSCA descriptor), whereas the second option included only dependency libraries while the task artifacts were downloaded and installed on-demand.

The results explicitly show that from the performance perspective the use of pre-packaged images is the most effective option. However, from the user perspective the quickest and easiest is the use of the base images already available on DockerHub instead of building images manually. Our framework supports such flexibility for the cost of some overhead incurred by the initial execution of a workflow. The first run will involve the complete deployment cycle and creation of the

images, whereas any subsequent executions will benefit from that images and will run at full speed.

C. Reproducibility in the Face of Development Changes

One of the key factors that can reduce decay of our workflows is their ability to embrace changes that occur naturally during workflow and task development. These changes may affect mainly two layers: the input/output interface of a workflow or task, and their implementation.

In Fig. 7 we illustrate a *hypothetical* evolution scenario of the Sequence Cleaning workflow shown earlier in Fig. 4. The left side depicts the timeline of development events that occurred in the scenario. It is accompanied by change trees from two repositories: the left tree represents the evolution of the workflow, the one on the right shows the evolution of one of the workflow tasks.

We start the analysis with the version of the SC workflow presented earlier and tagged as v1 in Fig. 7 (event 1). By tagging we acknowledge that this version has been published, advertised and so may be used by others.

Now, let us imagine that a new requirement for our workflow appeared (event 2) – users of the workflow want to save storage space by compressing the workflow output files. In response to that, the developers created a new Zip task (cf. the right version tree) and wanted to add it to the workflow. Note, however, that changing the type of outputs generated by the workflow is a change of its interface. For example, it would likely break any external application that has used uncompressed outputs provided by version v1. Thus, before we can add the Zip task to the workflow we need to create a new branch, named zipped in the figure (event 3).

The zipped branch of the workflow refers to the Zip/master branch of the task. By default such a reference means that the workflow depends on the latest tagged version of the task coming from that branch. This is convenient because as the task implementation is improved over time, the zipped workflow will use a task's latest tagged version (including v1.1). In this way workflows are updated automatically without the need to change them when only implementation improvements are made to the tasks. However, if strict workflow repeatability is required, the reference to the Zip task would include a specific tag. That would prevent the automatic update of such a workflow.

Next, event (4) denotes a new release of the Java library used by some tasks in the workflow. In our hypothetical scenario the new version of the library has improved performance and many errors fixed. Thus, the event is a signal for us to update the workflow as soon as possible. That change is compatible with the previous version of the workflow and so we do not need to create a new branch. Instead, we merge in the changes from master to the zipped branch, so that both branches can benefit from the updated library.

After adding the Zip task and updating the Java library we also tag and announce new, improved versions of our workflow (event 5). Specifically, SampleCleaning/v2 runs faster

and produces smaller outputs which is of great value to the users.

Event (6) marks the arrival of yet another requirement – users want the outputs of the workflow to be encrypted to avoid leakage of patients' raw genomic data. That requires, however, some improvements in the Zip task, including changes to the underlying tool used to compress the data.

After running some tests it appeared that the new zip tool has much better performance, and so we quickly decided to swap the old implementation with the new tool and tag the task v1.1. Note that this simple act of tagging a version causes an automatic update of all workflows that rely on that branch. Therefore, from now on the SampleCleaning.v1.1 and .v2 workflows will use the updated implementation of the Zip task.

Continuing with the task update, we create a new password branch in the task repository (event 7). This new branch is needed due to the changes in the task's input interface – the new version has the extra password input property. But the use of encryption is optional, so to limit the number of branches we decided to discontinue the previous version of the Zip task and tag branch master as deprecated (event 8). That indicates to users that they should use other branches of the task in their new workflows. Nonetheless, the old version will need to remain in the repository because others may still use workflow SampleCleaning.v2 which relies on the Zip/master branch.

As proactive workflow developers we noticed that the master branch of the Zip block has been deprecated and so we decided to update the reference in the zipped version of the workflow to the active password branch. Note that this update does not require a new branch because the use of encryption in the Zip task is optional. Thus, the workflow's input and output interface can remain the same (event 9).

The new branch is created later (event 10) when the password property is exposed to end users as the workflow input property. We want the users to be able to set a custom password for the output data and that requires a change in the workflow interface which, in turn, requires a new branch.

The presented hypothetical evolution shows very common patterns in the development of workflows and their components – changes can occur at different layers of workflow and tasks. However, by means of separate task and workflow repositories, and conscious tagging and branching of their code, we can maintain all workflow versions in the working state and also ensure that their evolution does not break external applications that rely on them.

VIII. CONCLUSIONS AND FUTURE WORK

Reproducibility is a crucial requirement for scientific experiments, enabling them to be verified, shared and further developed. Therefore, workflow reproducibility should be an important requirement in e-Science. In this paper we presented a design and prototype implementation of a framework that supports repeatability and reproducibility of scientific workflows. It combines two well-known techniques: logical and

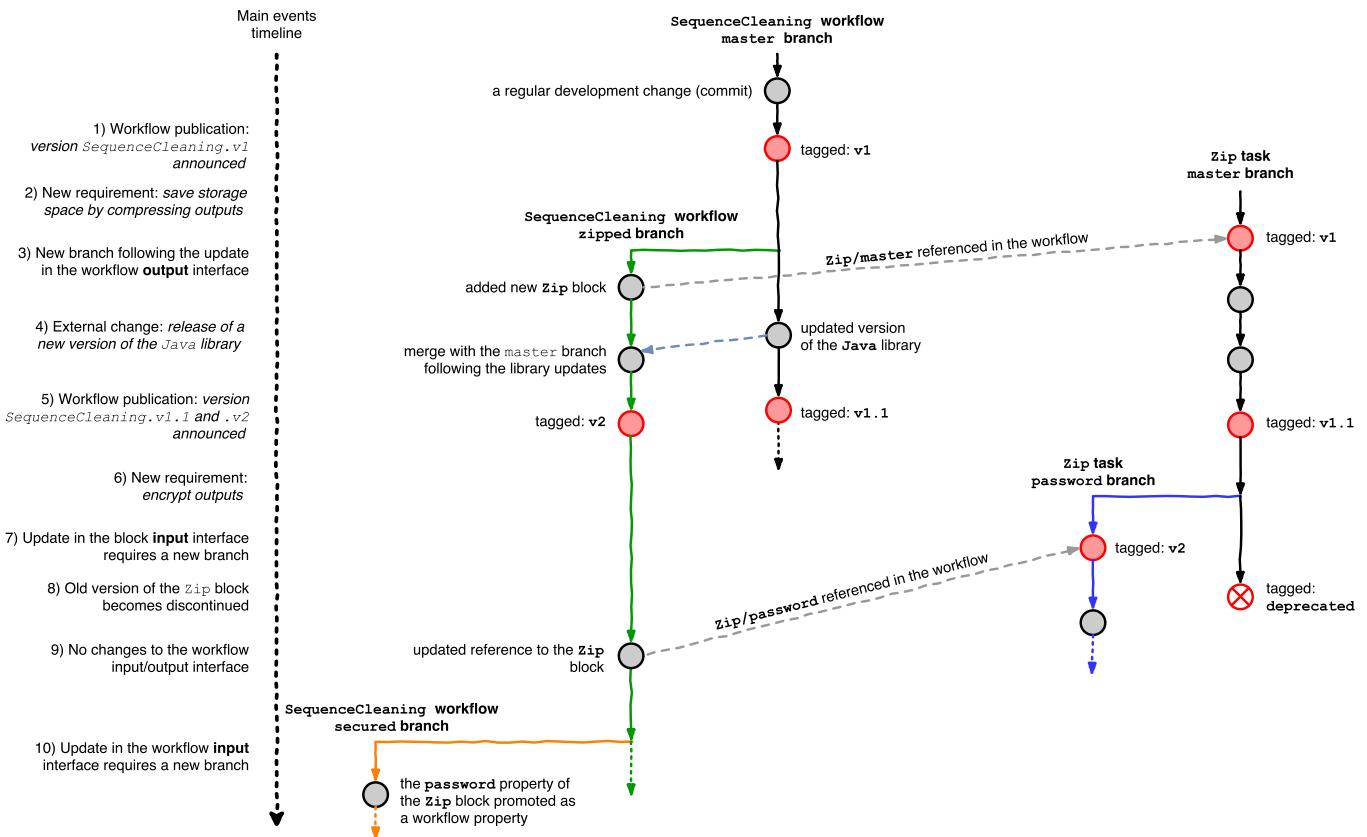


Fig. 7: A hypothetical evolution of the Sequence Cleaning workflow.

physical preservation. To implement the logical preservation technique we use the TOSCA specification as a means to describe workflows in a standardised way. To realise physical preservation we use lightweight virtualisation which allows us to package workflows, tasks and all their dependencies as Docker images.

Moreover, our framework uniquely combines software repositories to manage versioning of source code, an automated workflow deployment tool that facilitates workflow enactment and reuse, and automatic image creation to improve performance. They all significantly increase the degree of workflow reproducibility. And although, currently, our framework does not capture *retrospective* provenance traces, which has been left for future work, the proposed TOSCA-based workflow descriptors may be considered to be a detailed *prospective* provenance document. They describe the high-level structure of the workflow, which might also be encoded using, for example, the ProvONE specification,⁶ together with all details to recreate the complete software stack needed for deployment and enactment.

Still, however, a considerable part of reproducibility is in the hands of workflow developers: scientists and researchers who will use our tools. Only with their help and dedication

⁶The latest draft of the ProvONE specification from May 2016 is available at: <http://jenkins-1.dataone.org/jenkins/view/Documentation%20Projects/job/ProvONE-Documentation-trunk/ws/provenance/ProvONE/v1/provone.html>

can workflows be adequately described, have sample input and configuration data to facilitate testing, and be properly versioned with branches and tags indicating major development events. Our framework merely makes these tasks easier.

As for the future, the presented work opens a variety of interesting research avenues. We plan to add a facility to capture retrospective provenance information for workflows and tasks that could complement the history of their development. We consider implementing support for large-scale, distributed workflow enactment. Finally, we also plan to investigate to what extent our framework can model legacy workflows designed in other scientific workflow management systems like Pegasus and Taverna.

ACKNOWLEDGEMENT

This work was partially supported by EPSRC grant no. EP/N01426X/1 in the UK.

REFERENCES

- [1] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-Science: An overview of workflow system features and capabilities," *Future Generation Computer Systems*, vol. 25, no. 5, pp. 528–540, May 2009.
- [2] B. Liu, B. Sotomayor, R. Madduri, K. Chard, and I. Foster, "Deploying Bioinformatics Workflows on Clouds with Galaxy and Globus Provision," *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*, pp. 1087–1095, Nov. 2012.

- [3] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," *Future Generation Computer Systems*, vol. 29, no. 3, pp. 682–692, Mar. 2013.
- [4] H. Meng, R. Kommineni, Q. Pham, R. Gardner, T. Malik, and D. Thain, "An invariant framework for conducting reproducible computational science," *Journal of Computational Science*, vol. 9, pp. 137–142, 2015.
- [5] J. Zhao, J. M. Gomez-Perez, K. Belhajjame, G. Klyne, E. Garcia-Cuesta, A. Garrido, K. Hettne, M. Roos, D. De Roure, and C. Goble, "Why workflows break: Understanding and combating decay in Taverna workflows," in *2012 IEEE 8th International Conference on E-Science*. IEEE, Oct. 2012, pp. 1–9.
- [6] J. Goeks, A. Nekrutenko, and J. Taylor, "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences," *Genome biology*, vol. 11, p. R86, 2010.
- [7] A. Banati, P. Kacsuk, and M. Kozlovszky, "Four level provenance support to achieve portable reproducibility of scientific workflows," in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, no. May. IEEE, May 2015, pp. 241–244.
- [8] J. Freire, P. Bonnet, and D. Shasha, "Computational reproducibility: state-of-the-art, challenges, and database research opportunities," *Proceedings of the 2012 ACM SIGMOD ...*, pp. 593–596, 2012.
- [9] I. Santana-Perez, R. F. da Silva, M. Ryngé, E. Deelman, M. S. Pérez-Hernández, and O. Corcho, "Reproducibility of execution environments in computational science using Semantics and Clouds," *Future Generation Computer Systems*, 2016.
- [10] O. Standard, "Topology and Orchestration Specification for Cloud Applications version 1.0," pp. 1–114, 2013.
- [11] S. Arabas, M. R. Bareford, L. R. De Silva, I. P. Gent, B. M. Gorman, M. Hajiarabderkani, T. Henderson, L. Hutton, A. Konovalov, L. Kotthoff, C. McCreesh, M. a. Nacenta, R. R. Paul, K. E. J. Petrie, A. Razzaq, D. Reijsbergen, and K. Takeda, "Case Studies and Challenges in Reproducibility in the Computational Sciences," pp. 1–14, 2014.
- [12] F. Chirigati, D. Shasha, and J. Freire, "ReProZip : Using Provenance to Support Computational Reproducibility," *USENIX Workshop on the Theory and Practice of Provenance*, 2013.
- [13] V. Stodden, F. Leisch, and R. D. Peng, *Implementing reproducible research*. CRC Press, 2014.
- [14] B. Howe, "Virtual Appliances, Cloud Computing, and Reproducible Research," *Computing in Science & Engineering*, vol. 14, no. 4, pp. 36–41, Jul. 2012.
- [15] F. Jiang, C. Castillo, C. Schmitt, A. Mandal, P. Ruth, and I. Baldin, "Enabling workflow repeatability with virtualization support," *Proceedings of the 10th Workshop on Workflows in Support of Large-Scale Science - WORKS '15*, pp. 1–10, 2015.
- [16] O. Spjuth, M. Dahlö, F. Haziza, A. Kallio, E. Korpelainen, and E. Bongcam-Rudloff, "BioImg.org: A Catalog of Virtual Machine Images for the Life Sciences," *Bioinformatics and Biology Insights*, no. Vmi, p. 125, Sep. 2015.
- [17] P. Bonnet, S. Manegold, M. Björling, W. Cao, J. Gonzalez, J. Granados, N. Hall, S. Idreos, M. Ivanova, R. Johnson, D. Koop, T. Kraska, R. Müller, D. Olteanu, P. Papotti, C. Reilly, D. Tsirogiannis, C. Yu, J. Freire, and D. Shasha, "Repeatability and workability evaluation of sigmod 2011," *SIGMOD Rec.*, vol. 40, no. 2, pp. 45–48, Sep. 2011.
- [18] C. a. Goble, J. Bhagat, S. Aleksejevs, D. Cruickshank, D. Michaelides, D. Newman, M. Borkum, S. Bechhofer, M. Roos, P. Li, and D. de Roure, "myExperiment: A repository and social network for the sharing of bioinformatics workflows," *Nucleic Acids Research*, vol. 38, no. May, pp. 677–682, 2010.
- [19] K. Belhajjame, J. Zhao, D. Garijo, M. Gamble, K. Hettne, R. Palma, E. Mina, O. Corcho, J. M. Gómez-Pérez, S. Bechhofer, G. Klyne, and C. Goble, "Using a suite of ontologies for preserving workflow-centric research objects," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 32, pp. 16–42, 2015.
- [20] P. Missier, S. Woodman, H. Hiden, and P. Watson, "Provenance and data differencing for workflow reproducibility analysis," *Concurrency Computation Practice and Experience*, no. October, 2013.
- [21] D. McGuinness, J. Michaelis, L. Moreau, O. Hartig, and J. Zhao, "Provenance and Annotation of Data and Processes," *Ipaw*, vol. 5272, pp. 78–90–90, 2008.
- [22] K. Hasham, K. Munir, R. McClatchey, and J. Shamdasani, "Re-provisioning of Cloud-Based Execution Infrastructure Using the Cloud-Aware Provenance to Facilitate Scientific Workflow Execution Reproducibility," in *Cloud Computing and Services Science*, 2016, pp. 74–94.
- [23] Tobias Binz; Gerd Breiter; Frank Leymann; Thomas Spatzier, "Portable Cloud Services Using TOSCA," pp. 80–84, 2012.
- [24] T. Binz, U. Breitenbücher, O. Kopp, and F. Leymann, *TOSCA: Portable Automated Deployment and Management of Cloud Applications*, A. Bouguettaya, Q. Z. Sheng, and F. Daniel, Eds. New York, NY: Springer New York, 2014.
- [25] R. Qasha, J. Cala, and P. Watson, "Towards Automated Workflow Deployment in the Cloud Using TOSCA," in *2015 IEEE 8th International Conference on Cloud Computing*. IEEE, Jun. 2015, pp. 1037–1040.
- [26] D. Merkel, "Docker: lightweight Linux containers for consistent development and deployment," p. 2, 2014.
- [27] R. Chamberlain and J. Schommer, "Using Docker to support Reproducible Research (submission to WSSSPE2)," pp. 1–4, 2014.
- [28] C. Boettiger, "An introduction to Docker for reproducible research," *ACM SIGOPS Operating Systems Review*, vol. 49, no. 1, pp. 71–79, Jan. 2015.
- [29] J. Cala, E. Marei, Y. Xu, K. Takeda, and P. Missier, "Scalable and efficient whole-exome data processing using workflows on the cloud," *Future Generation Computer Systems*, Jan. 2016. [Online]. Available: <http://linkinghub.elsevier.com/retrieve/pii/S0167739X16000030>

Conducting Reproducible Research with Umbrella: Tracking, Creating, and Preserving Execution Environments

Haiyan Meng
and Douglas Thain
Department of Computer
Science and Engineering
University of Notre Dame
Notre Dame, IN 46556, USA
Email: hmeng,dthain@nd.edu

Alexander Vyushkov
Center for Research Computing
University of Notre Dame
Notre Dame, IN 46556, USA
Email: avyushko@nd.edu

Matthias Wolf
and Anna Woodard
Department of Physics
University of Notre Dame
Notre Dame, IN 46556, USA
Email: mwolf3,awoodard@nd.edu

Abstract—Publishing scientific results without the detailed execution environments describing how the results were collected makes it difficult or even impossible for the reader to reproduce the work. However, the configurations of the execution environments are too complex to be described easily by authors. To solve this problem, we propose a framework facilitating the conduct of reproducible research by tracking, creating, and preserving the comprehensive execution environments with Umbrella. The framework includes a lightweight, persistent and deployable execution environment specification, an execution engine which creates the specified execution environments, and an archiver which archives an execution environment into persistent storage services like Amazon S3 and Open Science Framework (OSF). The execution engine utilizes sandbox techniques like virtual machines (VMs), Linux containers and user-space tracers, to create an execution environment, and allows common dependencies like base OS images to be shared by sandboxes for different applications.

We evaluate our framework by utilizing it to reproduce three scientific applications from epidemiology, scene rendering, and high energy physics. We evaluate the time and space overhead of reproducing these applications, and the effectiveness of the chosen archive unit and mounting mechanism for allowing different applications to share dependencies. Our results show that these applications can be reproduced using different sandbox techniques successfully and efficiently, even through the overhead and performance slightly vary.

I. INTRODUCTION

Computational science has accelerated research progress in a broad spectrum of fields and provided the ability to do important research *in silico*. However there have been fewer fundamental advances in our methods for sharing scientific knowledge [1]. Experimental results may be plotted into beautiful figures and presented in an academic conference or published in a journal, however descriptions of the actual procedures by which results were achieved are often superficial and imprecise. Authors may mention the platform configuration used for their experiments in the Evaluation section - CPU, memory, network and disk, but seldom include the

details of the software stack, such as software version, dataset source, and analysis scripts.

Without a complete description of the execution environment used by the original authors, it may be difficult or even impossible to reproduce scientific work. A 2015 study of the repeatability in computer systems research examined 402 papers from ACM conferences and journals whose results were backed by code, and found that only 85 of the papers provided links to their codes in the paper itself. The study also showed that only the codes of 32.3% of the papers can be rebuilt within 30 minutes, the codes of another 16% of the papers can be rebuilt with extra effort, and it was difficult to rebuild the codes of the remaining 51.7% of the papers [23].

The execution environments utilized in computational research are complex, including hardware configuration, network topology, OS, software, data and environment variable settings. There are frequent updates and modifications in both software and hardware. Investigators may not even be aware of all the details of the software stack used for their experiments [24] and documenting the complex web of dependencies that are common in modern software environments would be a daunting task. Unless the full execution environment can be preserved in a timely manner, even the original authors will eventually have no way to reproduce their experiments.

Various attempts have been made to enhance traditional scholarly publication and make scientific results reproducible. Research Objects [2] was proposed to aggregate the data, methods, and people involved in an experiment to facilitate the reproducibility of scientific results. However, Research Objects only bundle together the necessary resources and are not directly executable. Dynamic documents [9], [20] and reproducible papers [8] were designed to integrate the text, code, data and other auxiliary materials to make it easier to reproduce the computations. However, it mainly focuses on software and data dependencies, and does not include the hardware, kernel and OS dependencies. Virtual machines [15] and virtual appliances [11] were used to wrap up the whole

software stack of an experiment into a virtual machine image (VMI), which can then be used to reproduce the experiment. However, this method may fail when the software stack is too large [16] and is not space-efficient, because common dependencies (e.g., shared libraries) are archived into different VMIs redundantly.

In this paper, we extend our previous work Umbrella [17], and propose a framework to help the researchers track, create and preserve their execution environments. Umbrella focuses on the reconstruction of computing execution environments for the purpose of portability and scalability on clusters, clouds and grids. The framework proposed in this paper includes three parts - the Umbrella specification, execution engine and archiver. The Umbrella specification allows the user to specify all the details of a comprehensive execution environment - including hardware, kernel, OS, software, data, environment variables, command, and output - through a lightweight, persistent and deployable JSON-format file. The Umbrella execution engine creates the execution environment specified in an Umbrella specification file using sandbox techniques like virtual machines (VMWare [32]), Linux containers (Docker [19]) and user-space tracers (Parrot [30]), and computing resources from the local machine and cloud computing services like Amazon EC2. The Umbrella execution engine also allows common dependencies like base OS images to be shared by different sandboxes concurrently. The Umbrella archiver allows the users to archive their execution environments into persistent storage services like Amazon S3 and OSF.

It is worth noting that an Umbrella specification says nothing about how to create the specified execution environment - where to create and which sandbox technique to use. Separating specifying execution environments from creating execution environments makes the specification generic, persistent and stable in spite of the evolution of sandbox techniques used to create execution environments.

With this framework, the original author can make an experiment reproducible by archiving its software and data dependencies using the Umbrella archiver (written in Python 2.6), and sharing the Umbrella specification which specifies all the dependencies and how these dependencies should be combined together during runtime. Any other researchers having the Umbrella specification, the proper access permission to the involved resources, and the Umbrella execution engine (written in Python 2.6) can easily reproduce the experiment.

We evaluate our framework by utilizing it to reproduce three applications from epidemiology, scene rendering, and high energy physics. For each application we evaluate the size of the Umbrella specification, the time and space overhead of creating execution environments using Umbrella, and the storage effectiveness of Umbrella for allowing dependencies to be shared by multiple execution environments. We also illustrate how an archival system, curateND, can be used to archive the dependencies of an application, create a DOI for the application, and provide an overview page including references to the Umbrella specification, experiment output and other auxiliary materials.

In summary, our contributions in this paper are two-fold:

- We introduce a lightweight, persistent and deployable specification to specify the execution environment of an experiment from hardware, kernel and OS all the way up to software, data and environment variables. The specification is deployable and can be used to reproduce an experiment easily. The specification is not tied to any specific sandbox technique, and is persistent in spite of the evolution of sandbox techniques used to create execution environments.
- Instead of storing a whole software stack of an experiment including OS, software and data as a single piece, Umbrella expects the archive unit of preserved dependencies to be basic OS image, software and data, and combines all the dependencies of an experiment at runtime using mounting mechanisms. This saves the storage space of both the archival system and the computing node by allowing different experiments to share a single copy of common dependencies.

In this paper we focus on improving the reproducibility of single-machine applications. The framework proposed here is not applicable to distributed applications, which often involve multiple software stacks and the communications between them. However, we plan to extend our framework in the future work to facilitate reproducing distributed applications.

Our framework aims to reproduce an experiment and get the same output, may not be applicable to reproduce performance-focused applications for two reasons: first, the CPU and memory fields in an Umbrella specification specify the minimal requirements, not the exact requirements; second, the Umbrella specification does not cover the parameters vital to performance-focused applications, such as CPU frequency, disk speed, and network performance.

II. WHY IS IT SO DIFFICULT TO REPRODUCE EXPERIMENT RESULTS?

To understand why it is so difficult to reproduce the experiment results published in an academic paper, let us examine the typical workflow of scientific research.

The computing resources used by most research institutions are maintained by professional system administrators, who install and upgrade the OS and system-level software and manage the user accounts. As a researcher, when Alice joined her new research group, she was given access to the computing resource, let us say, a server with the hostname of `lab01.phy.research.org`. As a non-root user, Alice installed software provided by some software community into her home directory, configured some environment variables for her convenience, wrote her own analysis script, named `analysis.py`, using `/usr/bin/python`, which happened to be Python 2.7. Then she downloaded the datasets from the Internet and kept them locally under the directory `/home/alice/data`, ran the experiment and got the experiment results, which were converted into beautiful figures. Finally, these figures were put into her academic paper and submitted to an academic conference.

Once the paper was accepted, Alice was happy and moved on to the next challenge in her research. Everything looked great until three months later another researcher, Bob, emailed her and wanted more instructions of how to reproduce the experiment published in her paper. Telling Bob that she ran the experiment on `lab01.phy.research.org` does not help anything, because it would be unrealistic to give the access to every researcher who wants to reproduce her experiment.

Alice searched her file system for the Python script, `analysis.py`, and was relieved to find it. She shared `analysis.py` with Bob and expected him to tell her that the experiment can be reproduced successfully. However, the news from Bob was both disappointing and surprising. The problems Bob encountered during his attempt of running `analysis.py` are:

- `analysis.py` depends on the setting of the environment variable `SIMCOUNT`;
- `analysis.py` expects an input file located at `/home/alice/data/file1`;
- `analysis.py` attempts to utilize an executable named `sim_sort`;
- the output of running `analysis.py` overflows Bob's memory and disk;
- `/usr/bin/python` on Bob's machine is Python 3.0, which is not backwards compatible with Python 2.7.

Unfortunately, Alice forgot to preserve the `SIMCOUNT` setting used for her paper, and deleted the directory `/home/alice/data` by accident. `sim_sort` is software under version control via Git and can be found, however, Alice forgot the commit id used for her paper. As for the memory and disk overflow, Alice realized she should have told Bob the experiment requires 6GB memory and 20GB disk space.

Although bad enough, these are only the problems relevant to the dependencies directly configured by Alice. In the background, the system administrators need to update, sometimes even upgrade, the kernel, OS and system-level software periodically. Every several years, the hardware equipment may become obsolete enough to be replaced. What's more, Alice's experiment may count on some network resources from third-party websites. Any change about these local and remote dependencies may result in the failure of reproducing Alice's experiment, no matter by herself or others.

Poor Alice! Can we do something to help her?

III. A FRAMEWORK FOR CONDUCTING REPRODUCIBLE RESEARCH

Given the importance of preserving the comprehensive execution environment of an experiment for its reproducibility and the complexity of figuring out all the details about the environment, we propose a framework to help scientific researchers improve the reproducibility of their research. Our framework achieves this through three mechanisms:

- allows the user to specify all the necessary details about a comprehensive execution environment - from the hardware all the way up to software and data (section III-A);

```
{
  "description": "A ray-tracing application which creates video frames.",
  "hardware": {
    "arch": "x86_64",
    "cores": "1",
    "memory": "1GB",
    "disk": "3GB"
  },
  "kernel": {
    "name": "linux",
    "version": ">=2.6.18"
  },
  "os": {
    "name": "redhat",
    "version": "6.5",
    "mountpoint": "/",
    "source": [ "http://ccl.cse.nd.edu/.../redhat-6.5-x86_64.tar.gz" ],
    "format": "tgz",
    "action": "unpack",
    "checksum": "669ab5ef94af84d273f8f92a86b7907a",
    "size": "633848940",
    "uncompressed_size": "1743656960",
    "ec2": {
      "ami": "ami-2cf8901c",
      "region": "us-west-2",
      "user": "ec2-user"
    }
  },
  "software": {
    "povray-3.6.1-redhat6-x86_64": {
      "mountpoint": "/software/povray-3.6.1-redhat6-x86_64",
      "source": [ "http://ccl.cse.nd.edu/.../povray-3.6.1-redhat6-x86_64.tar.gz" ],
      "format": "tgz",
      "action": "unpack",
      "checksum": "b02ba86dd3081a703b4b01dc463e0499",
      "size": "1471452",
      "uncompressed_size": "3010560"
    }
  },
  "data": {
    "4_cubes.pov": {
      "mountpoint": "/tmp/4_cubes.pov",
      "source": [ "http://ccl.cse.nd.edu/.../4_cubes.pov" ],
      "format": "plain",
      "action": "none",
      "checksum": "c65266cd2b672854b821ed93028a877a",
      "size": "1757"
    },
    ...
  },
  "environ": {
    "PWD": "/tmp"
  },
  "cmd": "povray +l/tmp/4_cubes.pov +O/tmp/frame000.png +K.0 -H50 -W50",
  "output": {
    "files": [ "/tmp/frame000.png" ],
    "dirs": [ "/tmp/output" ]
  }
}
```

Fig. 1. Umbrella Specification Example - `povray.umbrella`

- creates the specified execution environment using sandbox techniques like VMs, Linux containers and user-space tracers (section III-B);
- helps the user archive important data and software dependencies in the first place (section III-C).

A. Tracking Execution Environment: Umbrella Specification

Umbrella allows a user to specify a comprehensive execution environment through a JSON-format file independently of any deployment technology. The whole execution environment is specified through multiple sections, each section corresponds to a special aspect of the environment and may further include several subsections. Within each (sub)section, various attributes can be specified through `key:value` pairs. The `hardware` section specifies the requirements about the CPU architecture, the core number, the memory and disk consumption. The `kernel` section specifies the required kernel name and version. The `os` section specifies the name and version

TABLE I
RESOURCE URLs SUPPORTED BY UMBRELLA SPECIFICATIONS

Resource	Example URL
Local Filesystem	/home/hmeng/data/input
HTTP	http://www.data.com/index.html
HTTPS	https://lab.cse.nd.edu/index.html
Amazon S3	s3+https://s3.amazonaws.com/.../cubes.pov
Open Science Framework	osf+https://files.osf.io/v1/.../7559c3a
Git Repository	git+https://github.com/.../cctools.git
CernVM File System	cvmfs://cvmfs/cms.cern.ch

of the required OS image, which includes the basic root filesystem except the kernel. The `software` section and the `data` section specify respectively the required software and data dependencies, which may include multiple subsections, each corresponding to a single dependency. The `environ` section allows the user to specify the environment variable settings. The `cmd` section specifies the command line used to run the experiment. The `output` section specifies the location of the generated experiment results. Figure 1 illustrates the Umbrella specification for a Povray ray tracing application.

The `os` section and each subsection under the `software` and `data` sections provide detailed information about the resource location (`source`), the fixity of the resource (`checksum`, `size`, `uncompressed_size`), how to use the resource (`format`, `action`), and the mountpoint at runtime (`mountpoint`). The `format` attribute specifies whether the resource is a plain file (`plain`) or a gzipped tar file (`tgz`). The `action` attribute specifies how the resource should be used during runtime - used directly (`none`) or uncompressed first (`unpack`).

The `source` attribute provides a list of resource URLs of each dependency. The resources may come from local filesystems, public web servers, Amazon S3, OSF, Git Repositories, or CernVM File System (CVMFS) [3] (Table I). Resources from local filesystems can be utilized directly. Resources from public web servers are downloaded via HTTP or HTTPS protocol. URLs for resources from Amazon S3, OSF and Git Repositories, may have public or private access permissions, and are identified through their special prefixes - `s3+`, `osf+`, and `git+`. When private resources from these three sources are specified in an Umbrella specification, the user needs to provide correct authentication information. Resources from CVMFS can be accessed via the FUSE module or a user-space mount toolkit, Parrot [30].

Not all the sections are required in an Umbrella specification. If an experiment does not require any input data file, the `data` section can be ignored. The author of an Umbrella specification may add new sections, new subsections or new attributes according to his own needs. For example, an `ec2` attribute is added to specify an VMI from Amazon EC2 within the `os` section of Figure 1.

As for the software preservation format, Umbrella expects each software dependency is of binary format, not of source-code format. This decision is based on the following three observations: first, sometimes it is difficult to obtain the source

code, especially of commercial software; second, building software from source code is time-consuming; third, a successful building procedure requires special compilation toolchain and building configuration. In case the software building procedure needs to be reproduced, an Umbrella specification can be composed to track the compilation environment.

To make it easy to compose an Umbrella specification, we implement a web portal (<http://umbrella.basicuserinterface.com/>) which allows the user to specify an execution environment by filling a form online and automatically compute the metadata information of a dependency once its `source` attribute is provided. The portal also provides a specification validator to help diagnose the syntax errors within a specification. Currently, the Umbrella execution engine responds to check the semantic errors within a specification before creating the specified execution environment. In addition, Umbrella specifications are often the collaborative outcome between the researchers and system administrators. System administrators focus on the configurations of hardware, kernel and base OS image. The researchers can focus on the software, data and all the other experiment-specific configurations.

B. Creating Execution Environment: Umbrella Execution Engine

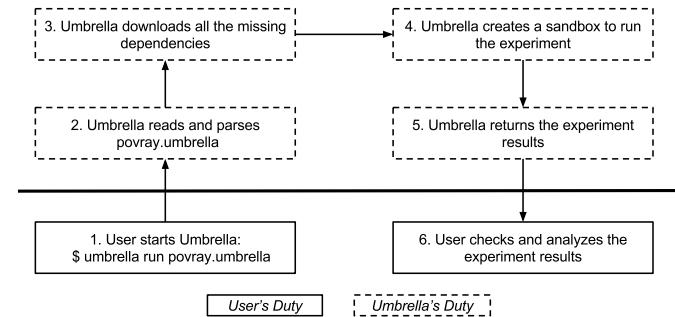


Fig. 2. Workflow of Umbrella Execution Engine

Figure 2 illustrates the workflow of the Umbrella execution engine. Once an Umbrella specification is ready, either composed from scratch or downloaded from the Internet, the user can start an Umbrella job (step 1) and wait for the experiment results to be returned by Umbrella. Umbrella is responsible for parsing the specification (step 2), downloading all the missing dependencies from the locations specified in the `source` attributes (step 3), creating the execution environment by mounting all the dependencies into a unified file system to run the experiment (step 4), and returning the experiment results to the location specified by the user (step 5). After this, the user can check and analyze the experiment results (step 6).

According to the matching degree between the requirements specified in the `hardware`, `kernel` and `os` sections of a specification and the `hardware`, `kernel` and `OS` configurations of an execution node, different sandbox techniques can be used to create execution environments. The higher the matching

TABLE II
SANDBOX TECHNIQUES FOR CREATING EXECUTION ENVIRONMENTS

Hardware	Kernel	OS	Sandbox Techniques
Yes	Yes	Yes	Utilize the current OS directly (section III-B1)
Yes	Yes	No	OS-level Virtualization - Docker, Parrot (section III-B2)
Yes/No	No	No	Hardware Virtualization - VirtualBox, VMWare, EC2 (section III-B3)

degree is, the lighter-weight the employed sandbox techniques can be. Table II shows the available sandbox techniques for each matching degree.

1) *Sandbox Technique - Utilize the Current OS Directly:* When the current node meets the hardware, kernel and OS requirements, it can be utilized directly to create the execution environment and every dependency will be put directly into the path specified by its `mountpoint` attribute. This method is fast because there is no virtualization layer being involved. However, the local filesystem may be polluted in two ways. When the `mountpoint` of a dependency has not existed yet, Umbrella should create the mountpoint before putting the dependency there. If the `mountpoint` already exists, Umbrella should first check whether the existing version is correct. In case the existing version is not the required one, the user needs to be consulted about which version to keep. Due to the lack of isolation mechanisms, this method does not block any damage which may be introduced by the experiment. Therefore, it is only feasible when the behavior of the experiment is safe or the execution node is easy to recover, such as a virtual machine.

2) *Sandbox Technique - OS-Level Virtualization:* If the hardware and kernel configurations of the execution node satisfy the requirements but the OS does not, OS-level virtualization techniques can be utilized to create the execution environment. OS-level virtualization allows multiple OS instances to run simultaneously on top of a single OS kernel. Compared with hardware virtualization, this has lower execution overhead because no hardware-level instruction translation is needed. The implementation of OS-level virtualization may only deploy file system isolation (such as `chroot` and Parrot) or isolate file system and network, and set limits about memory and CPU usage (such as Docker). By isolating the root filesystem of each OS instance, OS-level virtualization avoids the risk of ruining the file system of the execution node.

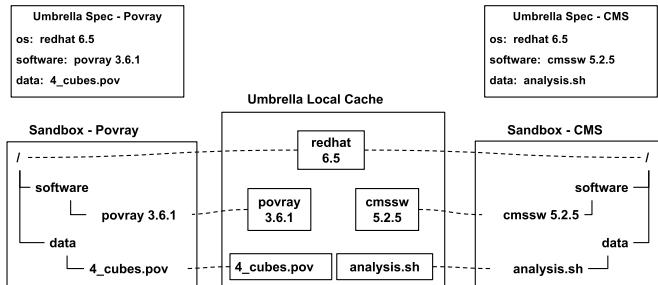


Fig. 3. Umbrella Local Cache - Allowing Dependency Sharing

This isolation also makes it possible to share the common

dependencies between sandboxes for different applications, as illustrated in Figure 3. The two applications - Povray and CMS - share the same OS image (`redhat 6.5`), each has its own software and data dependencies. Umbrella downloads all the distinct dependencies into its own local cache, and mounts the dependencies into each sandbox during runtime. In this scenario, the Umbrella local cache only keeps a single copy of the OS image, which will be shared by the two sandboxes. Within each sandbox, the data mounted from the Umbrella local cache can not be modified. All the modifications during runtime should be written to the mountpoint mounted from a location outside of the Umbrella local cache.

3) *Sandbox Technique - Hardware Virtualization:* When the hardware or kernel configurations of the execution node do not satisfy the requirements, hardware virtualization can be used to simulate the target computing environment in a virtual machine (VM), and create the execution environment within the VM. Compared with OS-level virtualization, Hardware virtualization involves more execution overhead because each instruction within the VM needs to be translated into the instruction on the host via a virtual machine monitor. However, since the virtual machine already satisfies the hardware, kernel and OS requirements, the sandbox can be constructed directly inside it, as discussed in section III-B1.

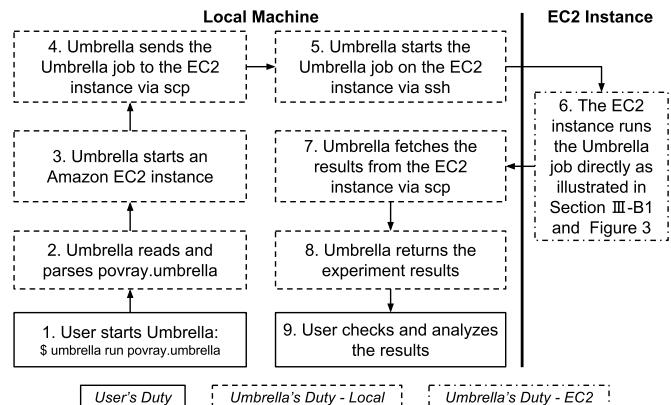


Fig. 4. Workflow of Executing An Umbrella Job via Amazon EC2

This sandbox technique can be implemented in two ways depending on where a virtual machine is hosted. If the current execution node has a hypervisor - such as VirtualBox [33] or VMWare - installed, it can be used directly to launch the required VM and finish the Umbrella job within it. If the current execution node has not installed any hypervisor, or the user prefers not to do the test locally, cloud computing platforms such as Amazon Elastic Computing Cloud (EC2)

and Google Compute Engine (GCE) can be utilized. The composers of Umbrella specifications should specify the required cloud platform and virtual machine image, as illustrated by the `ec2` subsection under the `os` section in Figure 1. Umbrella responds to communicate with the cloud platform to start a VM, send the Umbrella job to the VM, launch the Umbrella job on the VM locally. Then the VM creates the execution environment and finishes the job. Finally, the experiments results will be sent back to the local machine, and then put into the user-specified location. Figure 4 illustrates how Umbrella can utilize Amazon EC2 to finish a job.

C. Preserving Execution Environment - Umbrella Archiver

Once the researchers figure out the final experiment settings, it is time to archive the involved dependencies, especially those dependencies from unreliable sources, such as local disks and some third-party websites. The Umbrella archiver is designed to help the researchers to archive the dependencies specified in a specification into persistent storage services. It accepts the researchers' user credentials for the target storage services from its command line options and communicates with the target storage services via their Python bindings. By default, all the dependencies specified in a specification are archived into the storage system. The researchers may mark off the already-archived dependencies with a JSON field, `upload: false`. Once the archiving process is done, Umbrella will update the resource URLs of all the relevant dependencies to the reliable ones and generate a new Umbrella specification.

Currently, Umbrella supports two persistent storage service: Amazon S3 and OSF. Archiving an execution environment to Amazon S3 creates a new S3 bucket, uploads each unreliable dependency into the bucket, and finally uploads the updated Umbrella specification into the bucket. Then the researcher can publish and share the S3 link of the new Umbrella specification. Archiving an execution environment to OSF creates a new OSF project, archives each unreliable dependency as an OSF file under the OSF project, and finally uploads the updated Umbrella specification into the OSF project. Then the researcher can publish and share the OSF URL of the new Umbrella specification. The archiving procedure also allows the researcher to set the access permission of the uploaded OSF and S3 resources.

IV. EXAMPLE WORKFLOWS

In this section, we describe two typical scenarios where Umbrella can be used to facilitate the reproducibility of scientific research, and provide the detailed workflow of how to use Umbrella to achieve this. In both cases, Alice is the original author of the experiment, and Bob is another researcher who wants to reproduce Alice's work.

In the first scenario, Alice conducts her experiment on her local machine, with all the dependencies from the local machine. Alice first composes an Umbrella specification, `povray_local.umbrella` in Figure 5, to describe the execution environment of her experiment. Then Umbrella is used to create the specified execution environment and execute

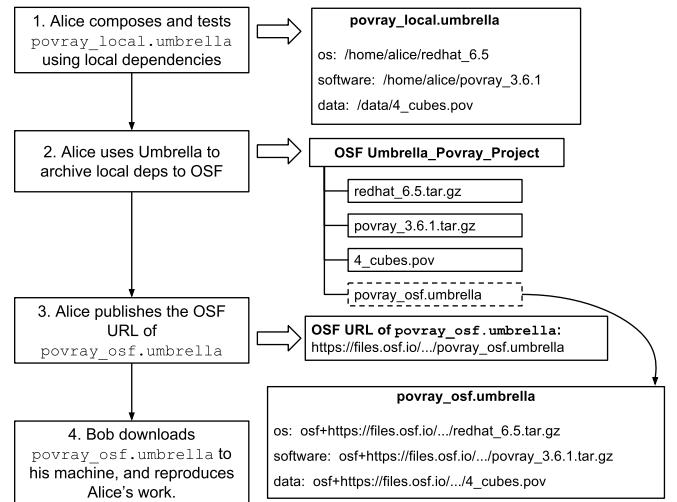


Fig. 5. Conducting Reproducible Research Using Umbrella - Local + OSF

the experiment. Whenever Alice wants to tune the experiment settings, she always first updates the execution environment specification. By tracking the execution environment as the research process goes and even before every real execution starts, Alice is always sure about the environment configurations of a successful execution. After Alice finishes her experiment, she uses Umbrella to creates a new OSF project and upload all the local dependencies into the OSF project. Umbrella also creates a new specification with all the dependencies from OSF, `povray_osf.umbrella`, which is also uploaded into the OSF project. Then Alice attaches the OSF URL of `povray_osf.umbrella` to her paper. If Bob reads Alice's paper and wants to reproduce the experiment, he can download `povray_osf.umbrella` using the OSF URL in the paper and reproduce Alice's work. Figure 5 illustrates the whole workflow.

In the second scenario, Alice wants to conduct her experiment using Amazon EC2. To avoid downloading dependencies from the outside Internet to her EC2 instance, Alice stores the software and data dependencies inside Amazon S3. Alice composes an Umbrella specification, `povray_ec2_s3.umbrella` in Figure 6, to describe the execution environment of her experiment. Then Alice uses Umbrella to communicate with AWS and executes the experiment using an EC2 instance (section III-B3). During runtime, all the dependencies are downloaded from Amazon S3 into the EC2 instance and then used to create the execution environment. When Alice is ready to publish her experiment result, she just needs to put `povray_ec2_s3.umbrella` into Amazon S3, and publishes its S3 link in her paper. By doing so, Bob can obtain a copy of `povray_ec2_s3.umbrella` easily and reproduce Alice's work.

In both scenarios, Bob can first read the Umbrella specification file shared by Alice to understand the hardware and software requirements of the experiment, and then decide where to reproduce it and which sandbox mode to use to

S3 link of povray_ec2_s3.umbrella: https://s3.amazonaws.com/povray/povray_ec2_s3.umbrella

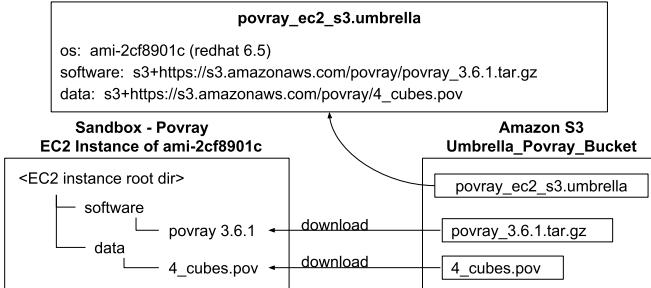


Fig. 6. Conducting Reproducible Research Using Umbrella - EC2 + S3

reproduce it. This helps Bob avoid the time overhead of asking Alice about the environment variable settings and input files, running out of memory and disk space by accident, and the failure caused by incompatible versions of Python.

V. EVALUATION

We have implemented Umbrella supporting the features described above - tracking, creating, and preserving execution environments of scientific research - using Python2.6. As for creating execution environments, four sandbox techniques are currently supported by our implementation: *destructive*, which utilizes the current OS directly; *parrot* and *docker*, which are two examples of OS-level virtualization; *ec2*, which is an example of hardware virtualization utilizing Amazon EC2 (*ec2* and *destructive* are used together: the local Umbrella uses *ec2* to start an VM and the remote Umbrella running on the VM uses *destructive* to run the experiment).

We evaluate our framework by utilizing it to reproduce three scientific applications from epidemiology, scene rendering, and high energy physics (section V-A). We evaluate the Umbrella specification file sizes (section V-B), the space and time overhead of creating execution environments using different sandbox techniques (section V-C), and the effectiveness of the Umbrella local cache for allowing dependencies to be shared by multiple execution environments (section V-D). Our evaluation results show that, with the help of Umbrella, an experiment can be reproduced successfully by the original author and others with different sandbox techniques, even though the overhead and performance may vary slightly.

A. Applications Evaluated

Three applications were used to evaluate our framework: OpenMalaria from epidemiology, Povray ray tracing application, and CMS from high energy physics.

OpenMalaria aims to develop a model to simulate the potential effects of the introduction of pre-erythrocytic malaria vaccines [26]. It uses individual-based stochastic simulations of malaria epidemiology to predict the impacts of interventions on infection, morbidity, mortality, health services use and costs. The OpenMalaria application used here does a VecNet baseline simulation with increased population size for more

TABLE III
SIZES OF APPLICATION DEPENDENCIES

Application	OS Deps	Software Deps	Data Deps
OpenMalaria	CentOS 6.6 (69MB/218MB)	openMalaria (2.9MB/13MB) .rpm packages (209MB) epel.repo (<1KB)	.xml (28KB) .csv (<1KB) .xsd (196KB)
Povray	RedHat 6.5 (605MB/1.8GB)	povray (1.5MB/2.9MB)	.pov (1.8KB) .inc (28KB)
CMS	RedHat 6.5 (605MB/1.8GB)	cmssw (1.3GB) parrot (23MB/71MB)	.sh (<1KB)

The size info of a plain-format dependency is in the format of `(size)`. The size info of a gzipped tar file dependency is in the format of `(compressed_size/uncompressed_size)`.

precise results. The application takes an xml-format input file which describes the place being simulated - human population distribution, entomology of vectors, effectiveness of health system in the area and optionally interventions applied to control malaria. The outputs of the application include a file with the size of 39KB capturing every timestep of a simulation (`ctsout.txt`) and a file with the size of 51KB aggregating data into configurable-size lumps (`output.txt`).

Povray is a ray tracing program which renders 3D graphics from text-based scene descriptions that describes all the details of a scene, including the camera, lights, plane and objects. All povray objects are described by mathematical functions and represented internally using their mathematical definitions. The povray application used here has two data dependencies: a scene description file (`.pov`) and a include file containing the mathematical functions of Rubik's Cube (`.inc`). The output of the application is a `.png` file with the size of 16MB.

The Compact Muon Solenoid (CMS) experiment investigates the most basic building blocks of matter by observing collisions of protons at near light-speed. Large numbers of collisions must be simulated and analyzed statistically to develop accurate models of the underlying physical processes. We applied the Umbrella framework to a step in this simulation process, in which collisions between protons are simulated. The results of each collision are generated, based on the model under test, and recorded. The input consists of a python configuration file describing the process to be investigated, and the output is a 96MB enhanced ROOT file containing the recorded particle descriptions. Due to metadata recorded at runtime which includes timestamps, all outputs will differ in a non-deterministic way. Because the psuedorandom seed is fixed, however, the physics content of the generated output of a given configuration will always be identical.

Table III illustrates the dependencies of each application and the size of every dependency. Two archive formats are used for the preservation of these dependencies: plain (e.g., `epel.repo`) and gzipped tar file (e.g., `CentOS 6.6`). Both the compressed

TABLE IV
UMBRELLA SPECIFICATION FILE SIZES

Application	OpenMalaria	Povray	CMS
Umbrella Spec Size	3.3KB	2.4KB	1.9KB

and uncompressed size of a gzipped tar file dependency are listed in Table III. All the dependencies are archived in a campus archival system, curateND.

Although only the evaluation results for three applications are shown in this paper, the framework does not bind itself with any specific field, and is applicable to the applications from other fields.

B. Umbrella Specification File Sizes

The execution environment for each application is tracked via an Umbrella specification file, which specifies the hardware, kernel, OS, software dependencies, data dependencies, environment variables, analysis command, and application output. Table IV illustrates the Umbrella specification file sizes for these three applications. An Umbrella specification only includes resource identifiers and other metadata information, rather than the real contents of any dependencies. Therefore, in contrast with the dependency sizes shown in Table III, an Umbrella specification file itself is a few kilobytes. To allow these applications to be reproduced with different sandbox techniques, the `os` section of each specification specifies an Amazon Machine Image (AMI) for the `ec2` sandbox mode and an OS image in the format of gzip tar file for the `parrot` and `docker` sandbox modes.

C. Overheads of Creating Execution Environments

Umbrella can create the execution environment specified in an Umbrella specification file with different sandbox techniques. Table V illustrates the time and space overheads of reproducing these three applications using three sandbox techniques - `parrot`, `docker` and `ec2`. Even if the time and space overheads of reproducing each application vary according to the sandbox techniques and computing resources used, they all generate the correct output.

The time overheads of different sandbox modes vary for two main reasons. First, sandbox techniques based on hardware virtualization (`ec2`) involve higher overheads than sandbox techniques based on OS-level virtualization (`parrot` and `docker`). Furthermore, the overheads of different OS-level virtualization techniques vary according to the isolation degree and the implementation details. Second, Umbrella allows the hardware and kernel requirements to be specified as a range, not limited as a single value. For example, the user can specify the CPU core number to be at least 2, the memory space to be at least 3GB. This feature makes it possible for an AWS user to reproduce an application using multiple instance types. We made this design decision because, from a long-term perspective, the aim of reproducibility is first to get the correct result, then to care about the performance.

The space overhead of the `parrot` sandbox mode covers the total size of plain-format dependencies and gzipped tar file dependencies (both compressed version and uncompressed version). This should be identical to the summary of the dependency sizes listed in Table III. The space overhead of the `docker` sandbox mode covers all the dependencies listed in Table III and an extra hard copy of the OS image dependency, which needs to be copied from the Umbrella local cache into the storage backend of Docker. The `ec2` sandbox mode only needs to download software and data dependencies into the Umbrella local cache on the EC2 instance, whose OS image already satisfies the requirement.

Comparing with Povray and CMS, creating the execution environment of OpenMalaria has a special step of installing several rpm packages via the package manager, yum, which requires the root authority. The `parrot` sandbox mode is based on a user-space mount toolkit, Parrot, therefore can not be used to create the execution environment of OpenMalaria. On the contrary, both `docker` and `ec2` sandbox modes can gain the root authority and can be used here.

In summary, the decision of choosing the right sandbox technique to reproduce an application depends on the following three factors - whether the user has root authority on an execution node, where the user wants to reproduce an application (locally or remotely), and whether the application itself involves privileged operations.

D. Effectiveness of Umbrella Local Cache

Umbrella tries to cache a dependency into its local cache when it is first required, so that the following experiments can reuse the local copy inside the local cache without downloading it repeatedly from the Internet. This saves both the time and economic cost of reproducing an experiment. Table VI shows the changes of the Umbrella local cache size as different experiments or different software/data dependencies are tested. The CMS experiment and the Povray experiment are used here, which share the same OS image dependency.

Originally, the Umbrella local cache is empty. When the CMS experiment is tested, all its dependencies, totally 2.39GB, are downloaded into the Umbrella local cache. Next, when the Povray experiment is tested, only its software and data dependencies are missing and added into the cache, totally 4.4MB. When the researcher wants to rerun any of these two experiments, all the required dependencies have existed in the local cache, and no new dependencies are needed. If the user wants to test a new software or data dependency, only the new dependencies will be downloaded from the Internet and added into the cache. Rerunning these experiments does not greatly reduce the execution time, because the execution node and the curateND archival system are on the same campus network, and the downloading speed from curateND to the execution node is about 30MB/s.

E. Last Step to Enhance Reproducibility

Our framework described above facilitates the reproducibility of scientific research. To go further, we created a DOI for

TABLE V
TIME AND SPACE OVERHEADS OF CREATING EXECUTION ENVIRONMENTS

Application	OpenMalaria	Povray	CMS	Permission	Location
Parrot	N/A	65min (2.40GB)	79min (2.39GB)	non-root	locally
Docker	57min (1.53GB)	68min (4.11GB)	82min (4.19GB)	root	locally
ec2 - m3.medium	113min (225MB)	130min (4.4MB)	211min (94MB)	non-root	remotely
ec2 - m3.large	58min (225MB)	65min (4.4MB)	108min (94MB)	non-root	remotely

The parrot and docker sandbox modes are tested on the same machine (hardware: x86_64, kernel: Linux 2.6.32, OS: redhat 6.7).

TABLE VI
EFFECTIVENESS OF UMBRELLA LOCAL CACHE

Application (Deps Size)	Cache Size	Delta (Newly Added Deps)	Time
CMS (2.39GB)	2.39GB	2.39GB (all deps)	79min
CMS - rerun	2.39GB	0	78min
Povray (2.40GB)	2.40GB	4.4MB (software & data)	64min
Povray - rerun	2.40GB	0	64min
Povray - new software deps	2.40GB	4.4MB (software)	64min
Povray - new data deps	2.40GB	28KB (data)	64min

The initial size of the Umbrella local cache is 0. All the tests here were done with the parrot sandbox mode on the same machine (hardware: x86_64, kernel: Linux 2.6.32, OS: redhat 6.7).

TABLE VII
DOIs FOR THE EVALUATED APPLICATIONS TO ENHANCE
REPRODUCIBILITY

Application	DOI URL
OpenMalaria	http://dx.doi.org/doi:10.7274/R03F4MH3
Povray	http://dx.doi.org/doi:10.7274/R0BZ63ZT
CMS	http://dx.doi.org/doi:10.7274/R0765C7T

each evaluated application, as shown in Table VII, using the campus archival service provided by our university library, curateND. Each DOI points to an overview page hosted by curateND, which includes the Umbrella specification file, the links to the Umbrella installation documentation and user manual, all the dependencies, and the experiment results. The original authors can decide when to publish and share their work by setting the proper access rights to the resources referred in the overview pages. Using curateND, anyone having the proper access rights can download the Umbrella specification for an experiment, install the Umbrella binary, reproduce the experiment, verify the published experiment results, and even extend the original experiment.

VI. RELATED WORK

In general, there are two approaches of preserving scientific software executions: preserving the mess and encouraging cleanliness [29].

Preserving the mess allows an application to be preserved, distributed, and shared in a self-contained package. Disk cloning [13] and virtual machine image [4], [14] are two common techniques to achieve this, which are easy to reuse, but are less feasible for complex applications with large software stacks with the size of TB level. To decrease the space overhead of preserved applications, Parrot-package toolkit [18], CDE [10], PTU [22] and ReproZip [6] trap the system calls of an application and only preserve the actually used files.

However, these solutions focus only on the local dependencies and do not preserve any network dependency. Furthermore, there are two main drawbacks of *preserving the mess*. First, the context and structure of the preserved applications are not usually clear enough for the new user to repurpose the application, therefore limiting their potential usage. Second, shared dependencies are redundantly preserved into multiple packages, which wastes the storage space of archival systems.

Encouraging cleanliness tries to specify an execution environment in a structured and cleanly way, archive dependencies of an application separately, and recreate the execution environment by mounting all the dependencies together into a unified sandbox. Software configuration management systems [7], like Puppet [31], Chef [27] and CFEngine [5], allow system administrators to specify the desired configuration for each managed device without considering the heterogeneity and complexity of the devices. This increases the scalability of system configuration by avoiding repetitive tasks, and reduces errors introduced by manual configuration through automation. However, software configuration systems do not consider the key problems of reproducible research, like data provenance, context description and access right [28]. Research Objects [2] can aggregate the data, methods and people involved in an experiment to facilitate the reproducibility of scientific results. However, Research Objects only bundle together all the necessary resource references, but are not deployable. Dynamic documents [9], [20] and reproducible papers [8] were designed to integrate the text, code, data and other auxiliary materials to make it easier to reproduce the computations. However, it mainly focuses on software and data dependencies, and does not include the hardware, kernel and OS dependencies.

Our work in this paper focuses on how to reproduce a single task. Researches focusing on improving the reproducibility of Scientific workflows, such as Prune [12], can utilize the framework proposed in this paper to reproduce each single

task in a scientific workflow.

Researches on data provenance [25] and semantic web [21] have been active to facilitate the management and reuse of scientific data. Our work focuses on how to specify and create the comprehensive execution environments of scientific applications. We have started working together with data archive services, such as curateND and OSF, to further improve the reproducibility of scientific applications.

VII. CONCLUSION AND FUTURE WORK

In this paper, we propose a framework facilitating the conduct of reproducible research by tracking, creating and preserving the comprehensive execution environments for their experiments with Umbrella. Using this framework, the researchers can make their experiments reproducible by specifying their execution environments in a lightweight, persistent and deployable execution environment specification, which can then be easily shared, expanded or repurposed. The researchers can also utilize the framework to archive their execution environments into persistent storage services like Amazon S3 and OSF to facilitate the sharing and publication of their experiments. The framework also provides an execution engine which allows the original researchers and any other researchers to (re)create the specified execution environments using sandbox techniques like VM, Docker and Parrot.

In the future work, we plan to explore the challenges involved in reproducing distributed applications, especially scientific workflows, compare different preservation degrees of workflow management systems, and extend our framework to facilitate the reproducibility of scientific workflows.

ACKNOWLEDGMENTS

This work was supported in part by National Science Foundation grants PHY-1247316 (DASPOS), OCI-1148330 (SI2) and PHY-1312842. The University of Notre Dame Center for Research Computing scientists provided critical technical assistance throughout this research effort.

REFERENCES

- [1] S. Bechhofer, J. Ainsworth, J. Bhagat, et al. Why Linked Data is Not Enough for Scientists. In *e-Science (e-Science), 2010 IEEE Sixth International Conference on*, pages 300–307. IEEE, 2010.
- [2] S. Bechhofer, D. De Roure, M. Gamble, et al. Research objects: Towards exchange and reuse of digital knowledge. *Nature Precedings* (2010), 2010.
- [3] J. Blomer, P. Buncic, and T. Fuhrmann. CernVM-FS: delivering scientific software to globally distributed computing resources. In *Proceedings of the first international workshop on Network-aware data management*, pages 49–56, New York, NY, USA, 2011. ACM.
- [4] P. Buncic, C. A. Sanchez, J. Blomer, et al. CernVM—a virtual software appliance for LHC applications. In *Journal of Physics: Conference Series*, volume 219, page 042003. IOP Publishing, 2010.
- [5] M. Burgess and O. College. Cfengine: a site configuration engine. *USENIX Computing systems*, 8(3):309–337, 1995.
- [6] F. Chirigati, D. Shasha, and J. Freire. ReproZip: Using Provenance to Support Computational Reproducibility. In *Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance*, Berkeley, CA, 2013. USENIX Association.
- [7] T. Delaet, W. Joosen, and B. Van Brabant. A Survey of System Configuration Tools. In *Proceedings of the 24th International Conference on Large Installation System Administration*, pages 1–8, Berkeley, CA, USA, 2010. USENIX Association.
- [8] J. Freire. Making computations and publications reproducible with vistrails. *Computing in Science & Engineering*, 14(4):18–25, 2012.
- [9] R. Gentleman and D. T. Lang. Statistical analyses and reproducible research. *Journal of Computational and Graphical Statistics*, pages 1–23, 2012.
- [10] P. J. Guo and D. R. Engler. CDE: Using System Call Interposition to Automatically Create Portable Software Packages. In *Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference*, pages 21–21, Berkeley, CA, USA, 2011. USENIX Association.
- [11] B. Howe. Virtual Appliances, Cloud Computing, and Reproducible Research. *Computing in Science Engineering*, 14(4):36–41, 2012.
- [12] P. Ivie and D. Thain. Prune: A preserving run environment for reproducible scientific computing. In *e-Science (e-Science), 2016 IEEE 12th International Conference on*, 2016.
- [13] E. Jeanvoine, L. Sarzyniec, and L. Nussbaum. Kadeploy3: Efficient and Scalable Operating System Provisioning for Clusters. *USENIX; login:*, 38(1):38–44, 2013.
- [14] G. Juve, E. Deelman, K. Vahi, et al. Scientific workflow applications on Amazon EC2. In *E-Science Workshops, 2009 5th IEEE International Conference on*, pages 59–66. IEEE, 2009.
- [15] I. Krsul, A. Ganguly, J. Zhang, et al. VMPlants: Providing and Managing Virtual Machine Execution Environments for Grid Computing. In *Supercomputing, 2004. Proceedings of the ACM/IEEE SC2004 Conference*, pages 7–7. IEEE, 2004.
- [16] H. Meng, R. Kommineni, Q. Pham, et al. An invariant framework for conducting reproducible computational science. *Journal of Computational Science*, 9:137–142, 2015.
- [17] H. Meng and D. Thain. Umbrella: A portable environment creator for reproducible computing on clusters, clouds, and grids. In *Proceedings of the 8th International Workshop on Virtualization Technologies in Distributed Computing*, pages 23–30. ACM, 2015.
- [18] H. Meng, M. Wolf, P. Ivie, et al. A case study in preserving a high energy physics application with Parrot. volume 664, page 032022. IOP Publishing, 2015.
- [19] D. Merkel. Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux Journal*, 2014(239):2, 2014.
- [20] J. P. Mesirov. Computer science. Accessible reproducible research. *Science*, 327(5964):415–416, 2010.
- [21] J. Myers, M. Hedstrom, D. Akmon, et al. Towards sustainable curation and preservation: The sead project’s data services approach. In *e-Science (e-Science), 2015 IEEE 11th International Conference on*, pages 485–494, Aug 2015.
- [22] Q. T. Pham. *A framework for reproducible computational research*. PhD thesis, The University of Chicago, 2014.
- [23] T. Proebsting and A. M. Warren. Repeatability and Benefaction in Computer Systems Research. Technical report, 2015.
- [24] C. Ruiz, O. Richard, and J. Emeras. *Reproducible software appliances for experimentation*, pages 33–42. Springer, 2014.
- [25] Y. L. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science. *SIGMOD Rec.*, 34(3):31–36, Sept. 2005.
- [26] T. Smith, N. Maire, A. Ross, et al. Towards a comprehensive simulation model of malaria epidemiology and control. *Parasitology*, 135(13):1507–1516, 2008.
- [27] D. Spinellis. Don’t Install Software by Hand. *IEEE Software*, 29(4):86–87, 2012.
- [28] R. Tansley, M. Bass, and M. Smith. DSpace as an open archival information system: Current status and future directions. In *Research and advanced technology for digital libraries*, pages 446–460. Springer, 2003.
- [29] D. Thain, P. Ivie, and H. Meng. Techniques for Preserving Scientific Software Executions: Preserve the Mess or Encourage Cleanliness? *Proceedings of the 12th International Conference on Digital Preservation (iPres 2015)*, 2015.
- [30] D. Thain and M. Livny. Parrot: An application environment for data-intensive computing. *Scalable Computing: Practice and Experience*, 6(3):9–18, 2005.
- [31] S. Walberg. Automate system administration tasks with puppet. *Linux Journal*, 2008(176):5, 2008.
- [32] C. A. Waldspurger. Memory resource management in VMware ESX server. *ACM SIGOPS Operating Systems Review*, 36(SI):181–194, 2002.
- [33] J. Watson. Virtualbox: bits and bytes masquerading as machines. *Linux Journal*, 2008(166):1, 2008.

Fast Window Aggregate on Array Database by Recursive Incremental Computation

Li Jiang
University of Tsukuba

Hideyuki Kawashima
University of Tsukuba

Osamu Tatebe
University of Tsukuba

Abstract—An array database is effective for managing and analyzing multidimensional scientific big data, and the window aggregate is an important operator in array databases. This paper proposes a method that exploits the scheme of incremental computation to accelerate the execution of window aggregates considerably. Six types of aggregate are improved using different designs of buffer tools to eliminate redundant computation. Our proposed recursive incremental computation method completely eliminates all redundant computation and achieves an improvement factor of the total window size compared with the naive method. This proposed method is fully implemented in SciDB. It improved performance by a factor of 10 on an earth science benchmark and by a factor of 64 on synthetic workloads with a certain data setting when compared with SciDB’s built-in window operator.

I. INTRODUCTION

A. Background

Scientific fields are growing increasingly data intensive. Spatiotemporal data from satellite imagery are very common in several fields, such as earth science, meteorology, and geography. Furthermore, telescope images and array sensing data generally have at least three dimensions. An example of a satellite data collection is MODIS [1], which is a data product provided by NASA. MODIS data are those gathered by two NASA satellites: Terra and Aqua [2]. The data schema has three dimensions: longitude, latitude, and time. They are provided to the scientific community free of charge, and some research projects, such as MODBASE [3], use scientific databases to handle the MODIS data.

For the management of array data, adoption of the relational data model remains difficult. In theory, a relational database can store multidimensional arrays with n -ary relations. However, it incurs a high cost in analytical tasks and in data management because of impedance mismatching [4]. To efficiently store and analyze such multidimensional data, array database systems, such as SciDB [4], [5], SciQL [6], and RasDaMan [7], have been developed. These systems use an array model as the basic data model to overcome the impedance mismatch problem. Array database systems have been adopted in some scientific applications that process multidimensional arrays, such as in cosmology [8] and earth science [3]. Among the current array database systems that have been developed, SciDB is open source and has been actively developed. It divides array data into small subsets, referred to as chunks, to deal with large quantities of data that do not fit in physical memory.

Query 1: $\text{window}(\text{arr}, 0, 1, 0, 2, \text{sum}(v))$

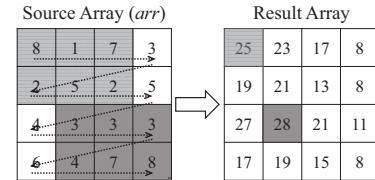


Fig. 1: Window Aggregate (Sum)

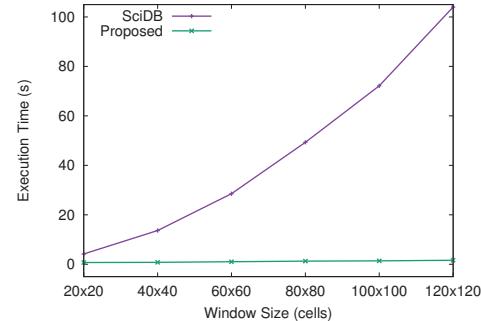


Fig. 2: Window Aggregate Performance

B. Problem: Slow Window Aggregate

One of the important array-oriented queries in such array database systems is called **window aggregate**, which is the query we target for acceleration in this study. An example of a window aggregate query is shown in Query 1 and Figure 1. The statement in Query 1 follows the syntax of SciDB. Here, arr is the name of the source array, which has two dimensions and an attribute named v . The aggregate function to compute is summation. The window area is set so as to expand the central cell one step higher in dimension 1 and two steps higher in dimension 2, making the window scope 2×3 . In Figure 1, the source array is shown on the left; the array on the right is the result. The dotted lines in the source array indicate how the window moves. Each cell in the result array contains the summation result for its corresponding window. Two cells in the result array and their corresponding windows are respectively marked with different shades of gray.

In current array databases, the straightforward method is used to compute this important operator. It is time consuming because a considerable amount of computation is performed

during the process. To process a window aggregate over multi-dimensional array data, each window is handled independently. The naive method scans every window, accumulates the values of all cells inside, and calculates the aggregate result. Thus, the performance of SciDB's built-in operator rapidly degrades as window size increases, as shown by the line labeled "SciDB" in Figure 2. The method ignores the common information shared between adjacent windows and calculates the overlapping areas repeatedly. With larger windows, such redundant calculation becomes significant and makes the whole process time consuming.

C. Contribution: Fast Window Aggregate

In this paper, we address the performance issue of window aggregate operators performed on multidimensional array data in array databases. We adopt a scheme of incremental computation to reduce the unnecessary calculation that exists in the naive method. The central idea is to buffer the intermediate aggregate results calculated for previous windows and reuse them when performing computations for a new window. We refer to the proposed method as the **recursive incremental computation (recursive IC)** method; it completely eliminates all redundant computation and thus greatly reduces the time cost of window aggregate tasks. Incremental computation was previously described in our earlier paper on the computation of window aggregate queries, as the "basic incremental computation" (basic IC) method [9]. However, the basic IC method achieves incremental computation in only a single dimension, whereas the proposal in this paper achieves incremental computation in every dimension, by using a recursive dimensionality-reduction process. This is a breakthrough, causing acceleration in all dimensions instead of only a single dimension, and significantly improves the processing of multidimensional data. The improvement factor of the proposed method against the naive method is proportional to the total window size as shown by the analysis given in Section V.

We exploit the plugin mechanism of SciDB to incorporate the proposed method into the system. The implementation of the recursive IC method demonstrates more efficient performance than the built-in window aggregate operator in SciDB. Our code is available on GitHub [10] so that all SciDB users can employ it. The slow window aggregate problem is solved by our proposed method, labeled as "Proposed" in Figure 2¹. We describe the details in this paper.

D. Organization

The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 presents preliminary concepts. Section 4 presents the proposal: the recursive incremental computation method. Section 5 explains the time and space complexity. Section 6 describes the evaluation and presents the results. Conclusions are given in Section 7.

¹The same result is also shown as Figure 9a in Section V, where the experimental settings are given.

II. RELATED WORK

The window aggregate is an important class of operators whose acceleration has been well studied. SAGA [11] offers efficient approaches for structural aggregates of array data. That proposal also deals with multidimensional data and the window aggregate; however, its focus is the reduction of disk I/O cost. That proposal is at the I/O level, reducing redundant data access in the storage hierarchy, whereas our work is at the algorithm level, reducing redundant computation in the query execution. Our proposal is outside the scope of SAGA; in fact, the improvement achieved by SAGA is orthogonal to our proposal, and thus both can be adopted together.

Incremental computation has been studied in the context of stream data processing [12]. While the performance improvements achieved in these studies are meaningful, they focus on 1-D data, and multidimensional problems are out of their scope. In addition, stream data are different from array data, which are prefetched and unchanged during query processing. Therefore, the "window aggregates" in stream data processing and in array data processing are actually two different types of query.

For temporal aggregates of interval data, important work has been done, including studies on balanced-tree [13] and SB-tree [14], in the incremental computation of the query. These concepts are designed to deal specifically with interval data and are not suitable for multidimensional array data, on which we focus herein; the underlying data models are different.

There is a set of graphic processing studies related to our work. The convolution filter exploits incremental computation and is implemented in OpenCV [15]; however, it does not address large arrays that do not fit in memory, and it works only in 2-D cases. Although some contest problems [16] are more complicated than our problem, they completely differ from the scenario of scientific data analysis tasks we address in this paper. Diamond-, hexagon-, and polygon-shaped window aggregates are discussed in [17]; however, these algorithms target 2-D images and are not applied to n -D scientific data; thus, such shapes are not supported by any array database systems, including SciDB. Furthermore, all the above studies focus only on algorithms but lack a system design perspective and an efficient solution for extremely large quantities of data that cannot fit in memory. In this paper, we provide an implementation of the proposed method on SciDB.

Regarding array databases, there have been studies extending scientific features such as data versioning [18] and uncertain data [19]. Additionally, efficient distributed storage and parallel processing of array data are discussed in [20]. Whereas these papers focus on architecture design and low-level array storage, our work is toward the improvement of a specific query on an array database.

III. INCREMENTAL AGGREGATES

Our technique is based on a shared module referred to as buffer tools. Therefore, we first describe the design of buffer tools and how they help achieve the efficient incremental computation of aggregates in the one-dimensional case.

A. Central Data Structure: Buffer Tools

In order to reduce redundant calculation, intermediate information from the calculated windows is stored and reused when computing other windows with the help of certain tools. In this paper, we refer to such tools as **buffer tools**. Buffer tools are responsible for maintaining and reusing intermediate results during the incremental computation process. A buffer tool has a data structure as the “buffer” to maintain information and provides interfaces that can update the buffer and, most importantly, fetch aggregate results on demand efficiently using the information stored in the buffer.

The first element of the buffer tool is efficient updating operators. The second element of the buffer tool is a fast operator for fetching the aggregate result using the buffered data. For them, different data structures are designed to achieve incremental computation for different aggregate operators. In this paper, we describe improved versions of six fundamental aggregate operators widely used in data analysis tasks: summation, average, variance, standard deviation, minimum, and maximum. In the following subsection, design details of buffer tools for each aggregate are introduced using a 1-D window aggregate scenario for ease of understanding.

B. Six Aggregate Operators

1) Summation and Average: The summation and the average share almost the same computing process. Thus, this pair is discussed using the sum as being representative. A circle list-like structure is used to buffer the data in the current window, along with an extra temporary **SUM** value maintained, being the summation value of all elements in the list; this structure is the buffer tool designed for summation, referred to as “sum list.” The **SUM** value serves to execute the result fetch efficiently in $O(1)$ time as it can be directly returned.

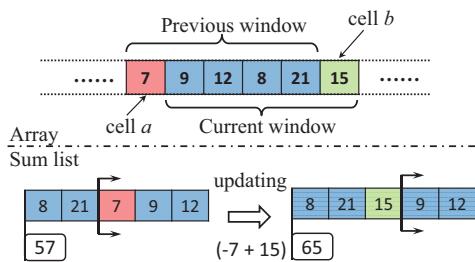


Fig. 3: Updating Details of Sum Aggregate

Let us consider how to compute a new window using the sum list being maintained. As shown in Figure 3, the oldest cell *a* is no longer in the current window, so it is removed from the buffer and subtracted from the **SUM** value; meanwhile, a new cell *b* has entered the window, so it is inserted into the buffer (replacing cell *a*) and is added to the **SUM** value. The update is then finished, and **SUM** is exactly the summation result for the new window.

2) Variance and Standard Deviation: Standard deviation is the square root of variance, so we discuss variance as being representative for the pair. The variance *Var* of a sample data set *X* is computed as given by Equation (1), where μ is the average value of the data set: $\mu = \frac{1}{n} \sum x_i$. Using this equation, the variance can be computed in constant time throughout the window aggregate process. The buffer tool is similar to sum list, with the same buffer structure as a circle list maintaining all of the current window’s cells. In addition, like the **SUM** value in the summation case, two values $\sum x_i$ and $\sum x_i^2$ are maintained. We call this buffer tool “var list.” When it comes to a new window, the buffer is updated in the same way as in sum list. Along with the updating, values $\sum x_i$ and $\sum x_i^2$ are also renewed in constant time.

$$Var = \frac{1}{n-1} \sum_{i=1}^n (x_i - \mu)^2 = \frac{1}{n-1} (\sum x_i^2 - \frac{1}{n} (\sum x_i)^2) \quad (1)$$

3) Minimum and Maximum: We take the “min” aggregate as being representative of this pair. We design an increasing circle queue as the buffer structure, “min queue.” Similar to sum list, it has a head pointer and a tail pointer for circulating the queue to save memory. Within the queue, it is assured that every element’s value is larger than its previous elements. When the window moves to a new position, updates need to be done. To delete the oldest cell, the position record of the queue’s head element is checked. If the head’s cell is no longer in the current window, it is removed by moving the head pointer one step forward. Then, to insert the new cell, the new value is compared with the tail element in the queue. If the tail is larger, it is removed by subtracting the tail pointer, and the process is repeated by comparing with the new tail element. When the process stops, the tail element is deemed to be smaller than the new cell’s value, and we insert the new cell into the buffer queue. In this way, the elements inside the buffer are assured to be in increasing order. For the result fetch operator, the head element of the queue is the smallest one in the buffer; thus it is the minimum result for the current window as well. Figure 4 shows a detailed example of the min window aggregate and how it operates incrementally with the assistance of the min queue.

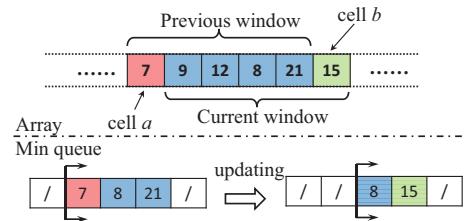


Fig. 4: Updating Details of Min Aggregate

IV. RECURSIVE DIMENSION REDUCTION

In this section, our proposal exploiting the incremental computation scheme is explained in detail. In the rest of this

paper, incremental computation is also expressed as “**IC**” for short.

A. Preparation: Basic IC Method

Before presenting our proposed method for multidimensional arrays, we first introduce a simple method that was proposed in our previous work [9] to make this paper self-contained. It is referred to as the “basic incremental computation” method or “**Basic IC**” in this paper. This method is an n -dimensional (n -D) solution, but its scale of improvement is limited to a single dimension.

1) Processing Details: With the above description in mind, we first explain three important concepts needed in order to understand the n -D scenario. These are: the **basic window**, the **computation round**, and the **window unit**.

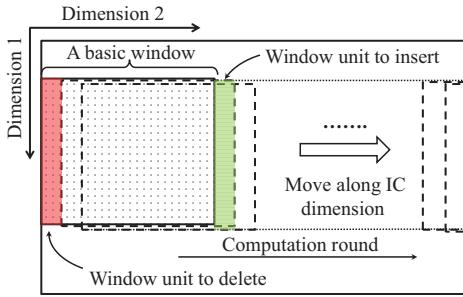


Fig. 5: Basic Window and Computation Round and Window Units

As shown in Figure 5, a basic window is the starting point of its corresponding computation round. During this computation round, the window slides along the IC dimension until all windows derived from this basic window are computed. Considering an n -dimensional source array, there are $\prod_{i=1}^n D_i$ windows to be computed, where D_i is the size of dimension i . These windows are divided into $\prod_{i=1}^{n-1} D_i$ groups. Each group of windows can be computed incrementally as a 1-D sub-task; the basic window is the first window in each 1-D incremental computation sub-task.

In Figure 5, dimension 2 is the IC dimension, and dimension 1 is used to generate the basic windows. Thus there are a total of D_1 basic windows’ computation rounds to be processed. Each computation round is a 1-D sub-task, and the windows’ aggregate results are computed incrementally with the help of buffer tools.

The last term is “window unit.” When the window moves to a new position, in this n -D case multiple cells are required to be updated instead of only two cells as in the 1-D IC process. Still, these cells have a certain positional pattern, and we use the term “window unit” to describe this group of cells.

When the window moves to a new position, one new window unit’s aggregate value is to be inserted and one old window unit’s aggregate value is to be deleted from the buffer tool. Whereas the naive method needs to scan the whole window, the basic IC method only needs to scan the new window unit; the other portion of the current window

can be directly reused from the buffer tool. Herein lies the improvement.

2) Weakness: Remaining Redundant Calculation: The weakness of this solution, the basic IC method, is that it adopts the IC scheme in only one dimension despite the fact that redundant work exists in all dimensions. When computing the aggregate value of the new window unit, every cell inside it needs to be scanned. Consider two neighboring basic windows and their computation rounds: overlapping areas are processed multiple times.

B. Proposal: Recursive IC Method

As discussed in the previous subsection, the remaining redundant calculation of the basic IC method lies in the scanning and computing of new window units when updating the buffer tools; this occurs because only a single dimension is processed incrementally. To completely eliminate such repeated work, our solution is to execute the multidimensional window aggregate **recursively**. The recursive incremental computation method we propose is the main contribution of this paper. By implementing incremental computation in every dimension, the method completely eliminates all redundant calculation. This is a solution targeted for the multidimensional scenario.

We design the recursive IC method to work at multiple levels. For an n -D window aggregate task, it is solved at n levels, different levels handling tasks of different size. The first level handles the original n -D task. At this level, a single dimension is selected as the IC dimension of this particular level. As in the basic IC method, at each step, when processing moves along the selected IC dimension, new window units need to be calculated. Calculation of the window units is equivalent to a window aggregate task with one fewer dimension at the next level. Thus, level 2 handles such $(n-1)$ -D tasks and pushes the task of computing its new window units to level 3. Finally, at the last level, there are the 1-D window aggregate tasks. Because the results of the window aggregate for each level are equivalent to the aggregate values of the new window units required to be updated at the previous level, once the current level’s IC process is finished, the required window units’ aggregate values for the previous level are immediately achieved and there is no need to scan them.

Therefore, the window aggregate can be performed recursively. Each level has its unique incremental computation dimension and specifically focuses on the incremental improvement in that IC dimension. An n -dimensional window aggregate’s incremental computation is achieved with the help of multiple $(n-1)$ -dimensional window aggregates, and these $(n-1)$ -dimensional window aggregates in turn are incrementally solved, exploiting the results of smaller $(n-2)$ -dimensional window aggregates at the next level. This recursive dimensionality reduction process continues until the number of dimensions is reduced to 1.

1) Processing Details for 3-D Case: Figure 6 shows a 3-D example of how a multidimensional window aggregate task is processed incrementally in all dimensions recursively.

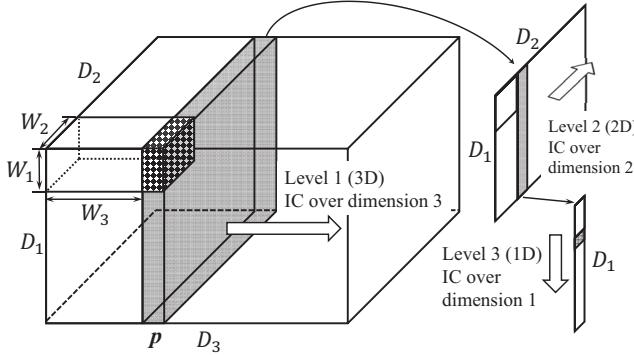


Fig. 6: Recursive IC of a 3-D Array

At level 1, a window has three dimensions, and its corresponding window units can be treated as two-dimensional windows. In each step, as processing moves along the IC dimension, new window units' aggregate values are needed to update the buffer tools and compute current windows' results. Again, a 2-D slice containing all these window units can be cut out, and a corresponding 2-D window aggregate can compute exactly these window units. In the figure, one new window unit is marked with a dotted background, and the dimensionality-reduced slice is marked in gray. The 2-D window aggregate over this slice is processed at level 2. Again, as shown above, computing the new window units in this 2-D window aggregate can be further broken down into 1-D window aggregate tasks, which are handled at level 3.

In brief, the 3-D window aggregate task is broken down into D_3 2-D window aggregate tasks. Each 2-D window aggregate task is further broken down into D_2 1-D window aggregate tasks. The recursive dimensionality reduction ends at this point because the incremental computation of a 1-D window aggregate is perfectly solved (as shown in Section II), as window units of a 1-D window are just single cells, whose values can be accessed directly as aggregate values.

For window aggregates with greater numbers of dimensions, the process is analogous. Any number of dimensions can be recursively broken down and eventually reduced to window aggregate tasks with a single dimension. It should be noted that the selection of the IC dimension at each level has no specific order. Any order will work as long as each dimension is selected once and only once during the entire process.

2) Weakness: Cost of Extra Space: Although the recursive incremental computation method greatly reduces the number of computations, it is obvious that it consumes a certain amount of extra memory for the buffer tools. This is a trade-off, consuming more space to gain time efficiency. However, the extra space cost is actually acceptable because of SciDB's chunking storage mechanism. We discuss this in further detail in the space complexity analysis given in Subsections V-E and V-F.

V. COMPLEXITY ANALYSIS

In this section, time and space complexities are analyzed and compared among the methods for computing window aggregates. In preparation for describing the analysis clearly and consistently, let us first define some parameters involved in the window aggregate operator. For a multidimensional array, its number of dimensions is denoted by n , and its dimension sizes are denoted by D_1, D_2, \dots, D_n . For a window aggregate query over such an array, the window sizes in each dimension are specified as W_1, W_2, \dots, W_n . According to this definition, the array size, which is the total number of cells, is $\prod_{i=1}^n D_i$, and the total number of cells within a normal complete window is $\prod_{i=1}^n W_i$.

A. Overview of Methods

In addition to the recursive IC method, three other methods are also evaluated in this study to compare their performance. These are the naive method, the materialized naive method, and the basic IC method.

- The **naive method** is the most straightforward method; it was described in Section I. It is the method currently used by most array databases, including SciDB.
- The **materialized naive method** is another method used in SciDB to compute window aggregates. The basic algorithm is the same as in the naive method but it materializes the data first, thus greatly reducing the data access cost. Before executing the window aggregate operator, it first scans every cell of the array and builds an index tree that maps multidimensional coordinate to the attribute value stored in the corresponding cell. With the help of this multidimensional index, accessing a cell's attribute value is efficient, whereas in the naive method, accessing a cell's value requires a computation of the location using coordinate offsets in each dimension.
- The **basic IC method** is that proposed in our earlier work [9]. It only reduces the redundant calculation in one dimension despite the fact that redundant calculation exists in every dimension. Therefore it is not as efficient as it could be, especially in cases with multidimensional data, which are the norm in scientific fields.
- The **recursive IC method** is the method proposed in this paper. It eliminates redundant calculation in every dimension in an effort to substantially improve performance relative to the basic IC method, particularly for multidimensional window aggregate tasks.

B. Naive Method

The naive method is quite straightforward. Each window is processed independently; therefore each one can be considered separately. For a given window, all cells inside it need to be scanned and accumulated into the aggregate result. Because there are $\prod_{i=1}^n W_i$ cells within a window, the scanning and aggregate-accumulating process of a single window obviously costs $O(\prod_{i=1}^n W_i)$. Meanwhile, a window aggregate operator has $\prod_{i=1}^n D_i$ windows to be executed because the windows are in a one-to-one mapping with the cells in the source array.

Multiplying them together, we obtain the total time complexity of the naive method: $O(\prod_{i=1}^n D_i \prod_{i=1}^n W_i)$.

C. Materialized Naive Method

For the materialized naive method, the time complexity is the same as that of the naive method. Differences in the data access costs only affect the constant part and will not change the dominant complexity. However, because each individual cell is accessed many times during processing when using the naive method, the materialized naive method shows great improvement in actual practice.

Regarding space cost, the naive method does not buffer any extra information and only processes original data from the source array; therefore it has no extra space cost. The materialized naive method, on the other hand, generates a one-to-one mapping index before processing the window aggregate operator, which incurs the cost of extra space for the size of the whole array, $O(\prod_{i=1}^n D_i)$. Because the array is generally extremely large in real-world scientific applications, this cost for extra space may seem too high. However, with SciDB's chunking storage mechanism, the actual extra memory cost at any given moment is acceptable.

D. Recursive IC Method

1) Time Complexity: For an n -dimensional window aggregate, the recursive incremental computation method executes in n levels. At the first level, with dimension 1 selected as the IC dimension, all of the basic windows move on their own computation rounds. For each position in dimension 1, corresponding new window units' aggregate values are required so that each round's buffer tool can be updated and aggregate results for new windows can be calculated incrementally. To compute these window units' aggregate values, an $(n - 1)$ -dimensional window aggregate task at level 2 is invoked. Thus, there are a total of D_1 $(n - 1)$ -dimensional window aggregate tasks to process at level 2. Similarly, each task at level 2 invokes D_2 $(n - 2)$ -dimensional window aggregate tasks at level 3, and so on in a recursive way.

Finally comes the one-dimensional task at the lowest level with dimension n selected as the IC dimension. The time complexity of this 1-D IC process is $O(D_n)$. Then the aggregate results for windows in this 1-D task help compute the higher level's window aggregate that invokes it as these windows are the window units required for that two-dimensional task. Again this is a recursive updating process from level n back to level 1, costing $O(1)$ at every level with updating and result-fetching methods of the buffer tools. For each window at level n , only one corresponding buffer tool is recursively updated at every level; thus the time cost for this recursive updating process is $O(n)$.

From the analysis above, there are a total of $\prod_{i=1}^{n-1} D_i$ 1-D window aggregate tasks invoked at level n during the entire recursive computation process; each such task has D_n windows, whose results need to be updated recursively through n levels. Therefore the time complexity for the recursive incremental computation method is $O(n \prod_{i=1}^n D_i)$.

2) Spatial Complexity: At level 1, there are $\prod_{i=2}^n D_i$ basic windows, and during each basic window's computation round, one buffer tool with size $O(W_1)$ is used to maintain and reuse intermediate data. Then at level 2, there are $\prod_{i=3}^n D_i$ basic windows, and each maintains one buffer tool with size $O(W_2)$. Finally, in the last dimension, there is one basic window, which maintains one buffer tool with size $O(W_n)$. Therefore, the total space cost is computed as follows:

$$W_1 \prod_{i=2}^n D_i + W_2 \prod_{i=3}^n D_i + \dots + W_n = O(W_1 \prod_{i=2}^n D_i) \quad (2)$$

On the left side is the accurate expression for the complexity, and the first term is dominant. Here we assure that at level 1 the IC dimension is the first dimension, as our implementation in SciDB is so designed. As in the materialized naive method, this extra memory cost seems expensive for very large arrays but thanks to the chunking storage mechanism of SciDB, the actual space cost is much smaller.

E. Chunking Mechanism in SciDB

As discussed in Subsections V-C and V-D2, the materialized naive method and the recursive IC method both have high costs for extra space when the scope of the whole array is considered. However, in SciDB, array data are divided into small subsets, referred to as chunks, to deal with large quantities of data that do not fit in physical memory. Chunks are the units for data storage and query processing. The window aggregate, the target operator in this study, can be processed chunk by chunk. Each time a chunk is loaded into main memory, the required data processing task over this portion of the data is executed. Therefore, when considering the extra space cost, it should be based on the scope of a chunk instead of the entire array.

Define the chunk sizes to be C_1, C_2, \dots, C_n for each dimension, respectively. It should be noted that C_i is much smaller than D_i . Then the actual space cost at any moment will be smaller than that given by the analysis above. For the materialized naive method, it is $O(\prod_{i=1}^n C_i)$, and for the recursive IC method, it is $O(W_1 \prod_{i=2}^n C_i)$.

F. Overall Comparison

We summarize the results of the time and space complexity analysis in Table I. “Basic IC” refers to the simpler incremental method we proposed in a previous paper [9]. From the analysis, it is obvious that in theory, the recursive incremental computation method is the fastest one and that the naive method is the most time-consuming one. Additionally, the materialized naive method seems to have the same order of time complexity as the naive one, but it has a smaller constant term reflecting the lower cost of data access.

When considering the space cost aspect, it can be observed that the extra space costs of the materialized naive method and the recursive IC method are relatively large. However, because the extra space cost is based on chunk size, it should be considered to be completely acceptable. Moreover, it should be noted that SciDB assures that a proper window aggregate

TABLE I: Time and Space Complexity Analysis Summary

Method	Time Complexity	Space Complexity
Naive	$O(\prod_{i=1}^n D_i \prod_{i=1}^n W_i)$	none
Materialized Naive	$O(\prod_{i=1}^n D_i \prod_{i=1}^n W_i)$	$O(\prod_{i=1}^n C_i)$
Basic IC	$O(\prod_{i=1}^n D_i \prod_{i=1}^{n-1} W_i)$	$O(W_n)$
Recursive IC	$O(n \prod_{i=1}^n D_i)$	$O(W_1 \prod_{i=2}^n C_i)$

n stands for the number of dimensions of the array; D_i stands for the array size in dimension i ; W_i stands for the window size in dimension i ; C_i stands for the chunk size in dimension i .

query's window size is always smaller than the source array's chunk size in every dimension, as $W_i < C_i$. Therefore, the extra cost of the recursive IC method is smaller than that of the materialized naive method.

VI. EVALUATION

We evaluated the proposed recursive IC method against SciDB's built-in naive method and the materialized naive method. Another method, the basic IC method [9] proposed in our earlier work, was also evaluated to find out whether the recursive IC method further improved the efficiency. Two series of experiments were conducted, one tested using actual earth science data and the other using synthetic data.

A. Implementation

To evaluate the performance of the recursive IC method against the naive method, we implemented it on SciDB using the plugin mechanism, which supports implementation of a user-defined operator (UDO). There is one rule when implementing a UDO plugin into SciDB. If it is possible for an operator's output array to be large, the operator should work as an iterative pipeline. To execute such a UDO, SciDB iterates every cell in the result array. When iteration proceeds to a new cell, a method for computing this new cell's result needs to have been implemented. The whole process is like a pipeline, and results are computed online and returned (i.e., output to the screen or saved to a file or saved to an array in storage) cell by cell. The window aggregate is exactly such an operator. The result array has the same size as the source array, which may reasonably be expected to be extremely large. Because of this special requirement of SciDB, when the recursive IC method was implemented in SciDB, the order of computation of the result cells in the output array needed to be reorganized.

As mentioned in the background (Subsection I-A), SciDB divides large arrays into small chunks and distributes them within a cluster. Operators can be efficiently executed chunk by chunk without extra disk I/O. When processing a chunk, a window aggregate operator works as an iterative pipeline. Therefore, the method for computing the result for the current

cell when iterating the chunks in the result array is the most important part of the implementation. The details can be found in our source code [10].

B. Experiment Cluster Specifications

We built a SciDB cluster consisting of 4 nodes to perform all the experiments in this study. Table II shows the cluster specifications. All of the servers shared the same machine configurations.

TABLE II: Experiment Configurations

Array Database System	SciDB 14.12
Operating System	CentOS 6.5
Processors	Intel(R) Xeon(R) E5620, 2.4 GHz
RAM	24 GB

C. Overall Comparison

Because the naive method and the basic IC method become quite time consuming when dealing with large arrays and large windows, we first show a comparison of the four methods' performance as tested using a relatively small array so that the performance comparison can be easily shown on a single graph.

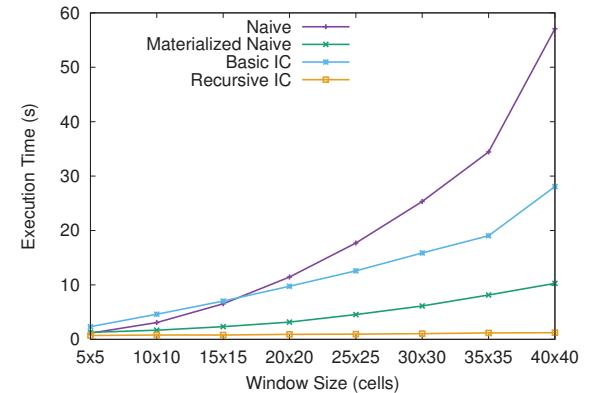


Fig. 7: Overall Comparison: Varying Window Size

Figure 7 shows the result. This experiment was conducted over a two-dimensional 1000×1000 array, and the window size was varied from 5×5 to 40×40 . Window size is an important parameter because the workload of naive methods is strongly related to it, whereas our proposed recursive method eliminates all redundant computation between windows, and thus its performance is almost unaffected by window size. The result confirms this feature, and the recursive IC method achieved the best performance. After the proposed method, the materialized naive method showed the best performance; the other two methods became more time consuming with higher total workloads.

In our further evaluation below, only results for the proposed method and for the materialized naive method are shown in the graphs. The other two methods consumed quite large amounts

of time, and including them in the performance graphs would have made the time scales improper and the graphs difficult to view.

D. Earth Science Benchmark

In this series of experiments, an earth science benchmark with a real-world scientific application was used. A related study, MODBASE [3], [21], attempted to exploit SciDB to manage the earth science data and process analysis tasks. In the earth science benchmark [21] proposed in the MODBASE study, the window aggregate operator is frequently used, especially for execution over the result arrays of other analysis tasks in order to produce arrays with lower resolution for purposes of visualization, comparison, and further analysis. For our experiment, we uploaded MODIS data into SciDB, following the MODBASE benchmark workflow, and executed the “Gridding Data” task from the benchmark, which is a window aggregate query.

1) Data Preprocessing and Upload: 45 MODIS files were downloaded with the area of interest set as the U.S. west coast and with a collection time frame of the year 2012. All of these settings follow the data used in the MODBASE benchmark. Each MODIS file is about 160 MB in size. These files are in HDF format, and the sensing data are stored in the satellite scanning order. Therefore, extra preprocessing work was required to map the sensing data into corresponding cells located by longitude–latitude coordinates before loading them into SciDB. Because this preprocessing work is not closely related to the topic of this paper, further details are not included here.

After preprocessing, CSV files were produced, which were loaded into SciDB. After uploaded, the earth data were in the schema of a three-dimensional array. The values in the longitude and latitude dimensions were scaled by 1000 to avoid different sensing samples overlapping in the same cell in the array. Thus, corresponding to the global area, the longitude dimension was in the range [-180,000, 180,000] and the latitude dimension in the range [-90,000, 90,000].

2) Gridding Task: In the earth science benchmark, the gridding task was executed on the results from other analysis tasks in order to down-sample the array so that the analysis results could be visualized. The gridding task actually consisted of two steps. The first step was a window aggregate query to compute the average value of every cell on earth with the window scope being $0.05^\circ \times 0.05^\circ$. In the second step, the down-sampling was conducted by selecting cells with a fixed skip length. Of course, the gridding task consumed the most time on the window aggregate step because it is computation intensive, whereas the second step was fast and can be ignored relative to the time consumed in the first step.

We selected the normalized difference vegetation index (NDVI) as the analysis task to produce an input array for the gridding task. We executed the NDVI calculation following the source code given in the MODBASE paper [1] as an NDVI task was also included in the MODBASE benchmark. Three sets of area parameters were designed, which are $10^\circ \times 10^\circ$,

$20^\circ \times 20^\circ$, and $30^\circ \times 30^\circ$, corresponding to the sizes of the area on the earth to be analyzed. The output result of an NDVI analysis is a 2-D array with the same size as the area size settings. After the NDVI analysis task was finished, the window aggregate operator was executed over the result array. The result array had two dimensions, and the following Query 2 is the window aggregate query statement performed over a 10° granule size of the NDVI result:

Query 2: `window(ndvi10, 25, 25, 25, 25, avg(ndvi))`

In this window aggregate query, for each cell, its window scope was set so as to expand in both dimensions and in both directions by 25 cells. Thus the total window size was 51×51 cells as the central cell also counts; this is the same window setting as in the MODBASE benchmark mentioned previously. The window scope was about $0.05^\circ \times 0.05^\circ$, corresponding to the 50×50 window on data after being loading into SciDB and with longitude and latitude scaled by 1000.

The experiment had three groups, corresponding to different sizes of areas on earth to be processed in the NDVI task. Window sizes were varied from 11×11 to 51×51 in order to determine how each method behaved. It should be noted that only the 51×51 window size is needed in the actual earth science application.

TABLE III: Metrics of NDVI Arrays

Area Size	Array Size	Density	Data Set Size
$10^\circ \times 10^\circ$	$10,000 \times 10,000$	28.79%	559 MB
$20^\circ \times 20^\circ$	$20,000 \times 20,000$	22.63%	1.78 GB
$30^\circ \times 30^\circ$	$30,000 \times 30,000$	26.75%	4.32 GB

Table III gives the details of the three source arrays used in the experiment. Because in the actual MODIS data not every cell is filled after the data mapping, there were empty cells in the array. In the table, “Array Size” shows the total array scope and “Density” shows how dense the array was.

3) Result: Figure 8a shows the experiment results for the $10^\circ \times 10^\circ$ area. The improvement is clear. Another interesting feature is that no matter how large the window was, the proposed recursive IC method’s execution time remained nearly constant. This is because in the recursive IC method, all redundant calculations are eliminated; thus every cell in the array is processed only once. Therefore, the time consumed is not related to the size of window. The materialized naive method, on the other hand, travels every cell in all windows, and its execution time increased as the total window size increased.

Figure 8b and Figure 8c show the $20^\circ \times 20^\circ$ and $30^\circ \times 30^\circ$ cases. The results show that considerable acceleration was achieved for the recursive IC method against SciDB’s built-in materialized naive method. With the window size at 51×51 as in the actual application, the proposed method achieved a speedup of about 10. In the case with the largest area ($30^\circ \times 30^\circ$), the materialized naive method needed more than

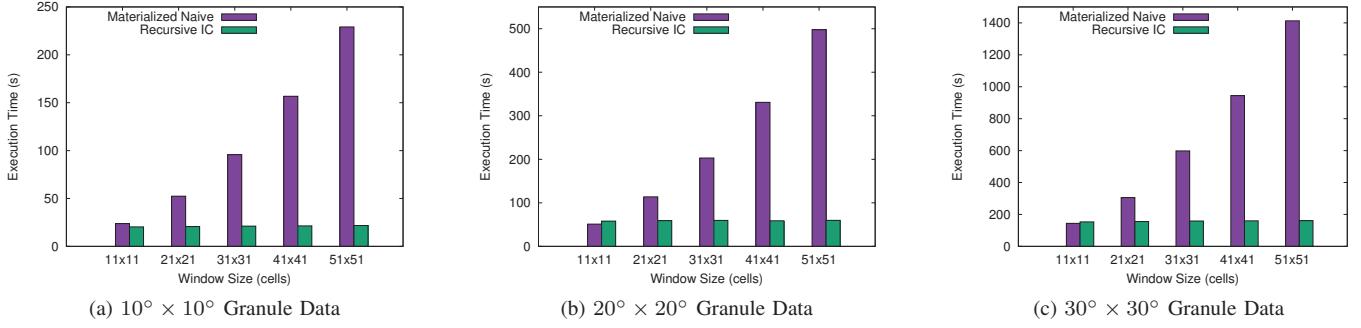


Fig. 8: Earth Science Benchmark Result

23 minutes, whereas the recursive IC method cost less than 3 minutes.

Compared with the time complexity analysis given in Section V, the improvement seems to be not as large as predicted in theory. This is because the result arrays of the NDVI tasks were not very dense. (The densities are shown in Table III.) Our proposed method will achieve the best performance improvement when dealing with dense arrays because with sparse arrays, the naive method can skip the empty cells, whereas with the incremental computation methods, during the IC computation round every step is necessary in order to update the buffer tools. Therefore, for a sparse array, the performance improvement shown by our proposed method will be somewhat less.

E. Synthetic Benchmark

In order to more thoroughly evaluate the performance of the methods, we generated several series of synthetic data to observe how the methods performed with various settings for different parameters. The attribute values of the arrays were randomly generated in the range [0, 100,000].

It should be noted that the aggregate function used for all performance results shown for this experiment was the **variance**. All three aggregate function pairs were tested, but it turned out that using different aggregate functions made only tiny differences in performance for the same method. Thus, we take the variance as being representative of all six aggregates and proceed to focus on the other parameters of interest.

1) Parameter 1 - Window Size: This evaluation was designed to observe the effect of the window size parameter. Because the amount of redundant computation is directly related to the window size, this parameter greatly affects the magnitude of the improvement factor of our proposed method against the naive method.

The result is shown in Figure 9a. From the result, it is clear that, with a larger window, the proposed method achieved greater improvement. A larger window means that more redundant calculation exists in the naive method, and our proposed recursive IC method completely eliminates the unnecessary calculation in all dimensions, greatly improving the performance. With the largest 120×120 window shown

in Figure 9a, the recursive IC method achieved a speedup of 64.07, finishing in less than 2 seconds, while it took the naive method almost 2 minutes.

2) Parameter 2 - Array Size: This evaluation was designed to observe the effect of the size of the total array on the performance of the window aggregate operator. Six 2-D arrays of different sizes were generated. For this evaluation, the window size was fixed at 10×50 .

Figure 9b shows the result. As the array size increased, both methods required more execution time. This is reasonable because larger array sizes lead to heavier workloads for computing the window aggregate operator; the total number of windows to compute is equal to the size of the array. It is also consistent with the results of the complexity analysis as the time complexities for all methods were found to be proportional to the total array size.

3) Parameter 3 - Dimensionality: This evaluation was designed to observe how the methods performed when arrays with more than three dimensions were processed since arrays of such high dimensionality were not covered in the previous experiment using actual earth science application data. In this evaluation, array sizes were fixed at 1,048,576, and window sizes were fixed at 1024 in all cases. The number of dimensions increased from two to six. Array sizes and window sizes were designed to be fixed so that the overall workload was the same for the naive method in all cases. In addition, the array and window schemas were designed to be as evenly distributed as possible so that the importance of each dimension was nearly the same. This was to ensure that the evaluation could expose the behaviors of the methods when dealing with high-dimensional arrays. For example, a 2-D array whose schema is $10,000 \times 2$ will show characteristics similar to a 1-D array with dimension size 10,000, whereas a 2-D array with schema 100×100 is sure to behave differently.

Figure 9c shows how the methods behaved as the data dimensionality increased. The recursive IC method gave excellent performance even when the dimensionality was high. With the highest dimensionality, six, recursive IC achieved a speedup of 225 over the materialized naive method. On the other hand, an unexpected phenomenon was observed: As the number of dimensions increased, the materialized naive

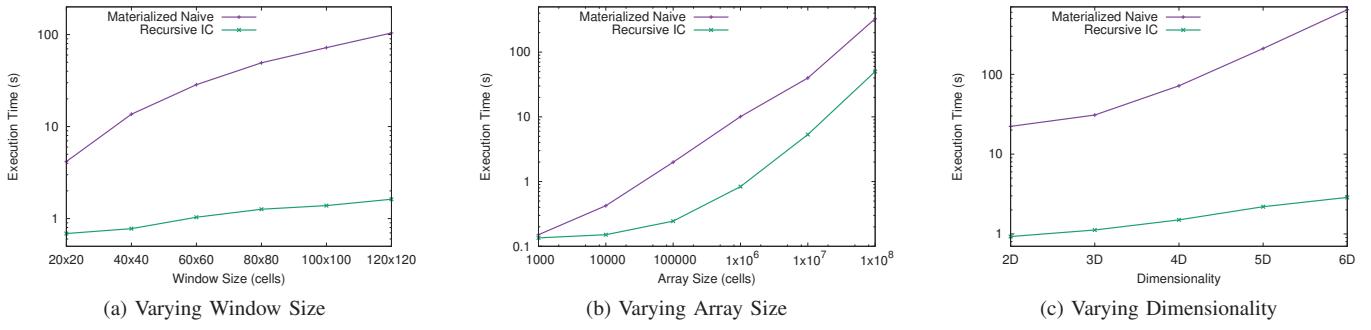


Fig. 9: Synthetic Benchmark Result

method became much slower than expected. According to the data design, the execution times of the materialized naive method were expected to be almost constant. The reason for this phenomenon is not clear. We have checked the source code of SciDB, and we found developers' comments implying that the current implementation of the materialized naive method was optimized for 2-D cases and was not so efficient for higher-dimensional cases [5].

VII. CONCLUSION

The window aggregate is an important operator in array-oriented processing tasks and is widely used in many scientific fields. However, this operator is computationally intensive, and its execution consumes large amounts of time.

We developed acceleration techniques for six aggregate functions, exploiting different designs of buffer tools to efficiently eliminate redundant calculation: a “circle-list” for summation, average, variance, and standard deviation, and a “increasing/decreasing queue” for maximum and minimum.

Analyses of time and space complexity are presented in this paper. To evaluate the proposed method, all methods were fully implemented in SciDB, a popular array database system; experiments included real-world applications in earth science as well as synthetic data. The experiments demonstrated the improvement achieved by the method. Compared with SciDB’s built-in operator, which is an implementation of the naive method, the proposed method improved performance by a factor of 10 for the MODIS earth science benchmark, and produced speedups of 64 and 225 on synthetic data for a large-window case and a high-dimensionality case, respectively. The improvement factor of the proposed method against the naive method was proportional to the total window size.

We believe this method is of benefit to all users of window aggregate operators in SciDB. Our source code is available on GitHub [10] so that any SciDB user can employ it.

ACKNOWLEDGEMENT

This work is supported by JST CREST “System Software for Post Petascale Data Intenseive Science”, “Extreme Big Data (EBD) Next Generation Big Data Infrastructure Technologies Towards Yottabyte / Year”, and “Statistical Computational Comsmology with Big Astronomical Imaging Data”.

REFERENCES

- [1] NASA, “Moderate resolution imaging spectroradiometer,” <http://modis.gsfc.nasa.gov/>, [accessed 12-January-2016].
- [2] ———, “Terra earth-observing satellite mission,” <http://terra.nasa.gov/>, [accessed 12-January-2016].
- [3] G. L. Planthaber Jr, “Modbase: A scidb-powered system for large-scale distributed storage and analysis of modis earth remote sensing data,” Ph.D. dissertation, Massachusetts Institute of Technology, 2012.
- [4] M. Stonebraker, J. Becla, D. J. DeWitt, K.-T. Lim, D. Maier, O. Ratzeberger, and S. B. Zdonik, “Requirements for science data bases and scidb,” in *CIDR*, 2009, pp. 173–184.
- [5] SciDB Developing Team, “Scidb v14.12 source code,” <http://forum.paradigm4.com/t/release-14-12/709>, 2015, [accessed 12-January-2016].
- [6] M. Kersten, Y. Zhang, M. Ivanova, and N. Nes, “Sciql, a query language for science applications,” in *EDBT/ICDT Workshop on Array Databases*. ACM, 2011, pp. 1–12.
- [7] P. Baumann, A. Dehmel, P. Furtado, R. Ritsch, and N. Widmann, “The multidimensional database system rasdaman,” in *SIGMOD Record*, vol. 27, no. 2. ACM, 1998, pp. 575–577.
- [8] J. Vanderplas, E. Soroush, K. S. Krughoff, M. Balazinska, and A. Connolly, “Squeezing a big orange into little boxes: The ascotdb system for parallel processing of data on a sphere.” *IEEE Data Eng. Bull.*, vol. 36, no. 4, pp. 11–20, 2013.
- [9] L. Jiang, H. Kawashima, and O. Tatebe, “Incremental window aggregates over array database,” in *BigData*. IEEE, 2014, pp. 183–188.
- [10] L. Jiang, “Source codes of the proposed incremental computation methods,” <https://github.com/ljiangjl/Recursive-IC-Window>, [accessed 12-January-2016].
- [11] Y. Wang, A. Nandi, and G. Agrawal, “Saga: array storage as a db with support for structural aggregations,” in *SSDBM*, no. 9. ACM, 2014.
- [12] J. Li, D. Maier, K. Tufte, V. Papadimos, and P. A. Tucker, “No pane, no gain: efficient evaluation of sliding-window aggregates over data streams,” *SIGMOD Record*, vol. 34, no. 1, pp. 39–44, 2005.
- [13] B. Moon, I. F. V. López, and V. Immanuel, “Scalable algorithms for large temporal aggregation,” in *ICDE*. IEEE, 2000, pp. 145–154.
- [14] J. Yang and J. Widom, “Incremental computation and maintenance of temporal aggregates,” *VLDB Journal*, vol. 12, no. 3, pp. 262–283, 2003.
- [15] G. B. García, O. D. Suárez, J. L. E. Aranda, J. S. Tercero, I. S. Gracia, and N. V. Enano, “Learning image processing with opencv,” 2015.
- [16] ACM-ICPC, “Square carpets,” <http://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=1128>, 2003, [accessed 12-January-2016].
- [17] C. Sun, “Diamond, hexagon, and general polygonal shaped window smoothing,” *Digital Image Computing*, 2003.
- [18] A. Seering, P. Cudre-Mauroux, S. Madden, and M. Stonebraker, “Efficient versioning for scientific array databases,” in *ICDE*. IEEE, 2012, pp. 1013–1024.
- [19] T. Ge and S. Zdonik, “Handling uncertain data in array database systems,” in *ICDE*. IEEE, 2008, pp. 1140–1149.
- [20] A. van Ballegooij, R. Cornacchia, A. P. de Vries, and M. Kersten, “Distribution rules for array database queries,” in *DEXA*. Springer, 2005, pp. 55–64.
- [21] M. Stonebraker, J. Duggan, L. Battle, and O. Papaemmanoil, “Earth science benchmark over modis data,” http://people.csail.mit.edu/jennie/elastictiy_benchmarks.html, [accessed 12-January-2016].

Representation of continuously changing data over time and space

Modeling the shape of spatiotemporal phenomena

José Moreira, Paulo Dias

DETI / IEETA

Universidade de Aveiro

Aveiro, Portugal

{jose.moreira, paulo.dias}@ua.pt

Pedro Amaral

DETI / Retail Consult

Universidade de Aveiro, Retail Consult

Portugal

apamaral@ua.pt

Abstract—There are numerous technologies and tools to acquire data related to the evolution of spatial phenomena over time. These data are typically organized as sequences of 2D geometric shapes obtained from observations taken at different times. The transformation of such sequences of 2D geometric shapes into spatiotemporal data representations, which can be easily processed and interpreted, has the potential to enable novel applications in fields as diverse as environmental sciences, climate sciences, biology or medicine.

This paper focuses on the representation of moving 2D geometric shapes acquired at discrete times using continuous models of time and space. Using morphing techniques based on compatible triangulations, issues regarding the representation of spatiotemporal data in databases, as well as the influence of different design strategies on the fidelity of the approximations with respect to the modelled phenomena, are investigated. An experimental study using synthetic and real data was performed. The findings show that the use of triangulation based interpolation is a promising approach, because it allows creating continuous spatiotemporal representations that are more realistic than those obtained using the solutions proposed in previous work. Open issues regarding the representation of spatiotemporal data in information systems are also highlighted.

Keywords—spatiotemporal databases; tracking spatial phenomena; 2D planar shape morphing; triangulation;

I. INTRODUCTION

The widespread use of novel technologies allowing to log large amounts of data concerning the location and shape of real-world phenomena over time is motivating the development of automated processes to extract knowledge from these data in several domains (e.g., earth, environmental or climate science) and applications (e.g., land use management, traffic control, urban planning or biomedical imaging). Nowadays, numerous methods and tools are available to deal with spatial data efficiently, namely spatial databases and GIS [1,2]. However, the representation and processing of spatial data changing over time is more complex, and there are several open-issues to overcome.

As for spatial data, spatiotemporal data can be represented by (1) approximating a continuous space by a discrete one, usually referred to as raster or tessellation mode, or (2) using a continuous representation, usually referred to as vector mode or

half-plane representation. The representation modes for the time and space dimensions are independent, and the choice strongly depends of the applications' requirements [3]. The focus of this work is on creating spatiotemporal data using continuous representations (vector mode) over space and time dimensions that might be used to model phenomena such as the movement, melting and collapsing of icebergs, wildfire propagation, siltation in rivers, the morphological behavior of cells and particles in biology, tracking of the left ventricle in medical imaging or the shape of galaxies in cosmology [4-14].

The evolution of spatiotemporal phenomena is typically acquired as discrete time-ordered sequences of observations in the form of images and represented using abstractions often referred to as moving or spatiotemporal objects. Thus, to capture the continuous change of such phenomena, methods are needed to model their spatial behavior between observations. The objective is to obtain continuous spatiotemporal representations that are close to the objects' actual shapes and movements at all times. This has been concretized on different data models and query languages, firstly using constraint databases [15], and then using Abstract Data Types (ADT) [16-21]. These solutions provide a comprehensive collection of general-purpose data types and query operations to deal with different application domains. Observations are represented by 2D shapes and spatial transformations between observations are represented using linear interpolation. However, few works exist on how to create realistic representations (interpolations) of real-world phenomena for spatiotemporal databases and GIS [22].

This paper investigates two approaches for creating spatiotemporal data, and the effect of different design strategies on the fidelity of the obtained representation when compared with the modelled phenomena. In the first approach, the transformation of an object is defined by the movement of line segments using the so-called rotating plane algorithm [24-26]. Translation and rotation are represented implicitly, and so, these solutions are poor in handling large rotations between observations. The second approach relies on using morphing techniques to attempt to preserve the morphological properties of 2D planar shapes during transformations [27-30]. Although these techniques are widely used, particularly in animation or video editing software packages, their use to model spatiotemporal phenomena in scientific applications is unusual

or nonexistent. The solution investigated in this work is based on compatible triangulations [27,28,31]. The aim is to bring together solutions from different research areas in order to be able to create realistic transformations representing spatiotemporal phenomena with small errors of approximation and reasonable runtime cost to enable querying large spatiotemporal datasets efficiently.

The comparison of these approaches is based on an empirical study using real and synthetic data. Real data consist of sequences of snapshots, tracking the movement of icebergs in the Antarctic. This is an interesting case study since it involves the three basic spatial transformations, namely, translation resulting from icebergs' movement, rotation caused by ocean currents near shorelines, and deformation due to melting and fragments collapsing. The objective is to compare the spatiotemporal representations created using both approaches with respect to the real-world phenomena modeled, as well as investigating computational aspects such as, the ability to yield spatiotemporal data that are topologically valid at all times.

The remainder of this paper is organized as follows: Section II presents the two approaches investigated in this study and highlights design and implementation issues of each one; Section III presents the results of the experimental study with emphasis on functional and computational aspects; Section IV presents a discussion and a comparison of the two approaches; Section V concludes and presents insights for future research.

II. CREATING SPATIOTEMPORAL DATA FROM OBSERVATIONS

This work addresses the problem of creating a continuous spatiotemporal representation of a phenomenon captured as a sequence 2D shapes at discrete times. It is assumed that the shapes have been previously extracted from a sequence of images or a video using well-known image processing tools such as ImageJ or OpenCV [32].

Let us consider a set of snapshots $D = (t, P)$, such that $t = \{t_i | i = 1 \dots n\}$ denotes an ordered set of timestamps and $P_i | i = 1 \dots n\}$ is a set of planar shapes. Each $P_i = \{v_j | j = 1 \dots m\}$ consists of an ordered sequence of 2D points corresponding to the vertices of the shape i . The shape is a polygon, if $v_1 = v_m$, or a polyline, otherwise. The former may represent an object such as an iceberg or a human tissue, and the latter may represent a forest fire front. A spatiotemporal transformation between consecutive snapshots is $D_i \xrightarrow{f(t)} D_{i+1}$, where $f(t) = E$ denotes a morphing algorithm that returns an estimation of a spatiotemporal object's shape (E) for every time $t \in [t_i, t_{i+1}]$. In addition, the topology of E must be valid for every time in that interval. A good solution should reduce deformation and create realistic transformations. In the following, the source (P_i) and the target (P_{i+1}) shapes in a spatiotemporal transformation $f(t)$ are denoted A and B , respectively.

A. The rotating plane algorithm

The rotating plane is a morphing algorithm proposed in [24] to create a linear interpolation between convex or concave polygons in spatiotemporal databases. This algorithm takes as

input two convex shapes and so, the angles of the edges of each shape with respect to a coordinate axis (e.g., the x-axis) are ordered. The main purpose of this algorithm is to create safe interpolations, i.e., to ensure that there are no intersecting segments during a transformation, so that the topology of the resulting spatiotemporal data is valid at all times.

The algorithm takes the first segment (edge) in both shapes and executes a synchronized scan of the two arrays of segments in A and B with respect to their angles. If the angles of the selected segments $\bar{a} \in A$ and $\bar{b} \in B$ are equal, a linear interpolation between them is performed, and the algorithm goes to the next segment in both arrays. This yields a trapezoid in space-time (Fig. 1, left). If the angles of \bar{a} and b are different, for instance, the angle of \bar{a} is smaller than the angle of \bar{b} , then the algorithm takes the first vertex v in \bar{b} , creates a linear interpolation between \bar{a} and v , and goes to the next segment in A . Notice that, in this case a segment degenerates into a point thus forming a triangle in space-time (Fig. 1, right).

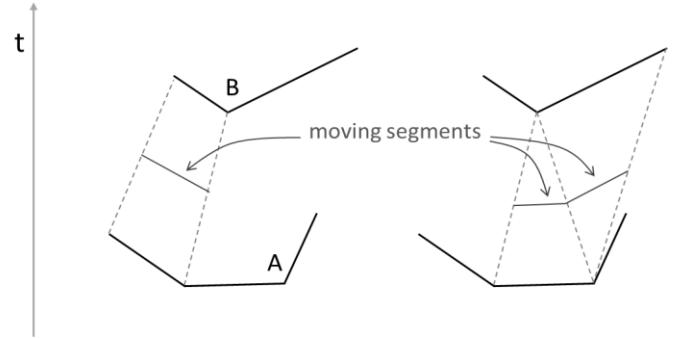


Fig. 1. Interpolation of two parallel segments (left) and a segment degeneration (right).

Concave shapes are split into features and organized as convex-hull trees. To build a convex-hull tree it is necessary to add extra segments to make the shape convex. Each extra segment gives origin to a new child node containing the convex-hull of the shape that should be removed from the parent node to obtain the original shape. The convex features may then be processed using the rotating plane algorithm.

Since each shape is decomposed into several convex features, it is necessary to find a matching between features in the convex-hull trees of A and B . Reference [24] recommends using criteria based on an overlap threshold between features. This strategy works well with synthetic data, but problems arise when dealing with real noisy data. For instance, consider Fig. 2, which depicts a projection of the convex-hull trees of the polygons representing the shape of an iceberg at two consecutive snapshots. The convex-hull tree at the top has four levels, and the one in the bottom has three levels. The shape of the iceberg is represented by the polygons in gray. The polygons in the circle show the details of the decomposition of a part of the shape into convex features. The feature in red (the larger one) denotes the convex-hull of the concavity pointed by the arrow in the right. The feature in blue (the smaller one) denotes a concavity in the feature in red. As this feature is convex, the decomposition stops at this level.

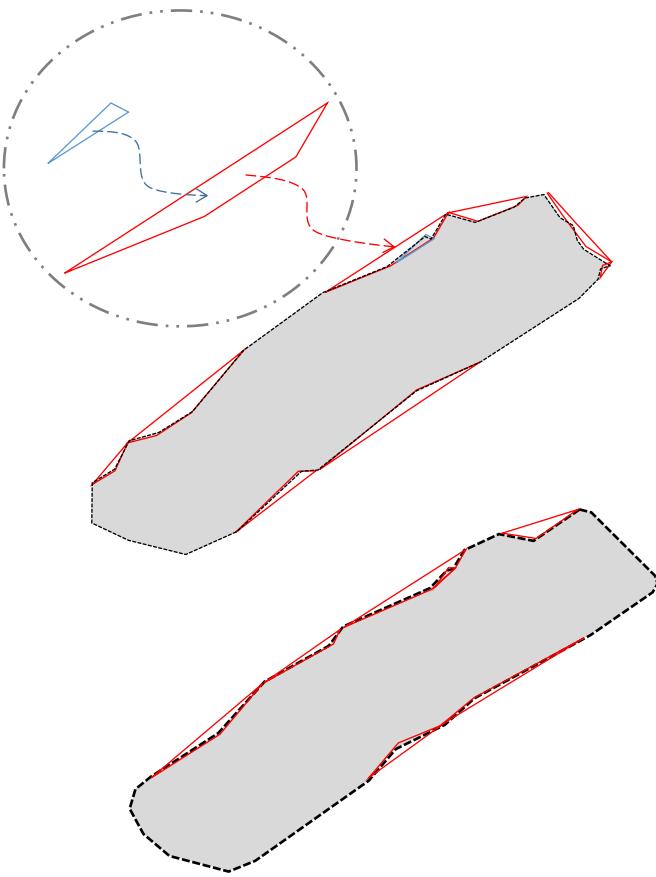


Fig. 2. Projection of the convex-hull trees for two consecutive snapshots of an iceberg.

This figure shows that the external features are often thin and it is not trivial to find a match in the other shape. In our implementation, no overlap threshold was found that would allow obtaining a good matching between features of the convex-hulls trees of A and B , when using real data. When the overlap threshold is high, the algorithm may fail to match small concavities; when the overlap threshold is low, there are components that may be erroneously matched; in some cases, there is no overlap at all. Fig. 3 gives illustrative examples. This figure depicts the overlapping between the features in levels two and three of the convex-hull trees of A (blue) and B (red).

As an alternative, a matching procedure using the minimum distance between the centroids of the features in the convex-hulls trees of A and B was implemented, but the results were worse than using an overlap threshold.

Paper [25] presents a solution to compute the morphing of two convex or concave shapes, which also uses the rotating plane algorithm, but the strategy to deal with concavities is different. In addition, the authors claim that it is not always possible to use a single interpolation to describe the transformation of a source polygon into a target polygon. This claim is supported with an example representing the transformation of two polygons with a snail shell configuration,

to illustrate that some configurations require splitting the interpolation into at most three interpolations.

The strategy to deal with concavities consists of mapping all segments forming a concavity in A into a single vertex in the convex-hull of B , or vice-versa. This means that a concavity in A degenerates into a point in B or a point in A expands to a concavity in B .

The last step consists of detecting edge intersections during the interpolation. These cases may occur for complex shapes with concavities that resemble spirals, i.e., when there are points that are surrounded by a concavity. The solution proposed in [25] consists of splitting an interpolation into smaller ones, so that surrounding concavities degenerate into a single point and then expand again. Note that this strategy, as well as the previous one used to deal with simple concavities, causes deformation during transformations. Indeed, the main purpose of the rotating plane algorithm and the other procedures described above is creating spatiotemporal data that are topologically valid at all times, i.e., to avoid segment intersections during spatial transformations.

The movement of the segments during a spatial transformation yields a triangle or a trapezoid, which are planar faces. This is an important feature because this kind of representation can be smoothly integrated into current spatiotemporal data models.

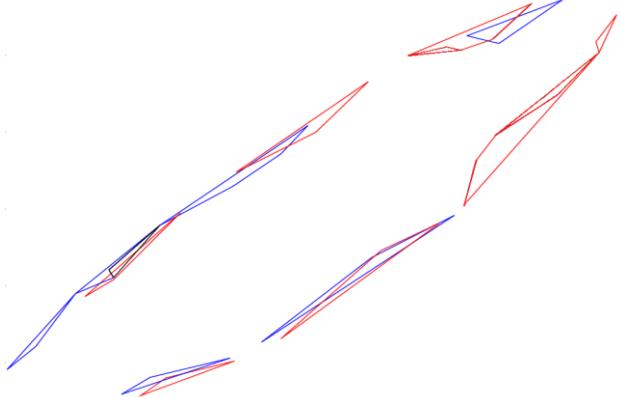


Fig. 3. Overlapping between features in the convex-hull trees of two consecutive snapshots of an Iceberg: only features in levels 2 and 3 are displayed.

B. Rigid shape interpolation

The aim of rigid shape interpolation is to preserve the shape's physical properties during the transformation of a source into a target, i.e., to perform a spatial transformation that keeps deformation as low as possible. This work uses compatible triangulations, and is based on the approach initially proposed by [28] and followed in several subsequent works [27,29,30]. This approach is considered to handle rotation naturally and has low runtime cost.

It is assumed that a correspondence between the vertices of A and B is created beforehand. The vertex correspondence is a mapping where each vertex in A has a corresponding vertex in B . If the number of vertices in A and B differs (a normal

situation with real data) it is necessary to add extra vertices in A or B . This is a well-known problem in morphing, which is commonly referred to as vertex correspondence problem and several solutions exist to perform this task [33,34]. The first step consists of creating a compatible triangulation to generate an isomorphic meshing of the interiors of A and B , denoted (\hat{A}) and (\hat{B}) , respectively. The algorithm presented in [31] was selected, because it allows to create compatible triangulations yielding a small number of Steiner vertices (extra vertices).

This algorithm tends to produce meshes that are not optimized. Therefore, a smoothing procedure is needed to obtain an approximately uniform spatial distribution of the vertices in a mesh. Two criteria for mesh smoothing were used: angle optimization and area optimization. Both use an iterative algorithm to generate alternative positions for each Steiner vertex in order to find the position that maximizes the minimum angle (or the minimum area) of the triangles sharing that vertex. Fig. 4 shows an example of a mesh before (top) and after (bottom) smoothing.

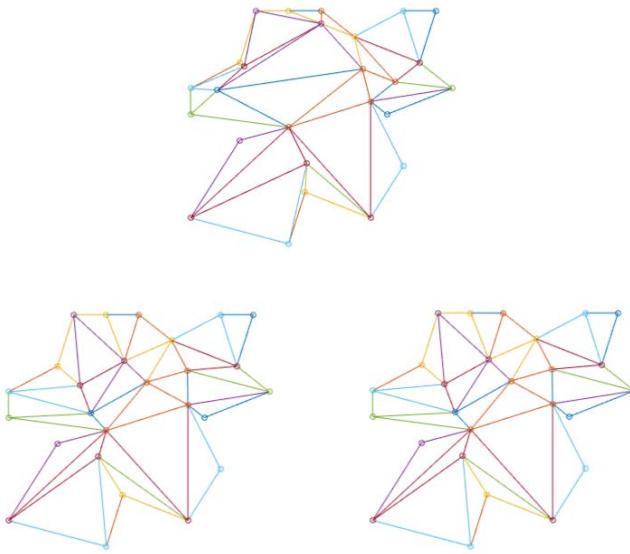


Fig. 4. Polygon triangulation before optimization (top), after minimum angle optimization (bottom-left) and after minimum area optimization (bottom-right).

The second step consists of the interpolation between \hat{A} and \hat{B} . The transformation of a triangle $X \in \hat{A}$ into a triangle $Y \in \hat{B}$ is a matrix M such that:

$$Y_i = M \cdot X_i + L, \quad i \in \{1,2,3\} \quad (1)$$

where M is an affine matrix and L is a vector denoting the translation of the centroid of A towards the centroid of B . Since the motivation of this work is modeling the extent of spatial phenomena during time, we assume that the translation component can be easily interpolated and so, the focus is on the evaluation of the affine matrix that minimizes deformation.

The simplest solution to evaluate the affine matrix is to use linear interpolation. However, this is not a good solution because when triangles rotate, they tend to shrink until the middle of the

interpolation and then they expand again, causing large deformation. An alternative solution proposed in [28] is a factorization of the affine matrix using Single Value Decomposition (SVD). Several types of decomposition derived from SVD have been proposed in literature [29]. In this work, the decomposition proposed in [27] was selected, since it ensures symmetry, i.e., the interpolation from source to target is equal to the interpolation from target to source. The affine matrix is a factorization $M = R_Y \cdot S$, such that, $R_Y = R_1 \cdot R_2$ and $S = R_2^T \cdot D \cdot R_2$. The matrices R_1 , D and R_2 are obtained by decomposition of the affine matrix M using SVD, and represent a rotation, a deformation (scaling and shear) and another rotation, respectively. So, the intermediate shape of a triangle at a time instant $t \in [t_i, t_{i+1}]$ is:

$$V_i(t) = M(t) \cdot X_i + L(t), \quad i \in \{1,2,3\} \quad (2)$$

where $V(t)$ is the estimation of the coordinates of the vertices of triangle X at time t . However, applying this formula to each triangle in a mesh can generate different positions for shared vertices since each triangle has its own transformation formula (Fig. 5).

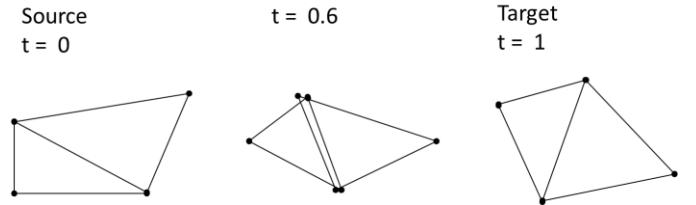


Fig. 5. Shared vertices with non-coincident positions during the interpolation of two adjacent triangles.

Two methods were implemented to deal with this problem. The simplest method was to consider that the position of a shared vertex is the centroid of the positions estimated in each transformation. The second method uses normal equations as proposed in [27]. The coordinates of the vertices of all triangles in a mesh are:

$$\hat{V}(t) = (\hat{P}^T \cdot \hat{P})^{-1} \cdot \hat{P}^T \cdot M_Y(t) \quad (3)$$

such that,

$$M_Y(t) = [M(t)_1^T \cdots M(t)_k^T]^T \quad (4)$$

$$\hat{P} = [\hat{P}_1^T \cdots \hat{P}_m^T]^T \quad (5)$$

where k stands for the number of triangles in the mesh and \hat{P}_j is a $2 \times m$ sparse matrix (m is the number of vertices of the source polygon) with all elements equal to zero, except the values in the columns corresponding to the indices of the vertices of triangle j . Each P_j denotes the coordinates of the vertices of triangle j . The results using the centroid of the coordinates were equal or worse than the results obtained using normal equations, and so, hereafter we only consider the latter.

Using the equations above, the rotation angles are in the interval $[-180, 180]$ degrees. However, this may not be the best overall angle to ensure consistency, since the rotation of adjacent triangles must have the same orientation. So, a post-processing step is needed to deal with rotational consistency [27]. First, the rotation matrix R_y is analyzed to check for adjacent triangles with different orientation. A discontinuity exists when the difference between the rotation angles of adjacent triangles is greater than 180 degrees. In such cases, multiples of 360 degrees are added or subtracted to make the absolute value of the difference between the rotation angles of adjacent triangles less or equal to 180 degrees. This creates consistent rotations but does not guarantee that the rotation of all triangles is minimal. This problem was mitigated by computing the average of all rotation angles, and adding or subtracting multiples of 360 degrees to make this average close to zero.

It is important to emphasize that the faces produced by the movement of the edges of a shape in space-time are non-planar, as depicted in Fig. 6.

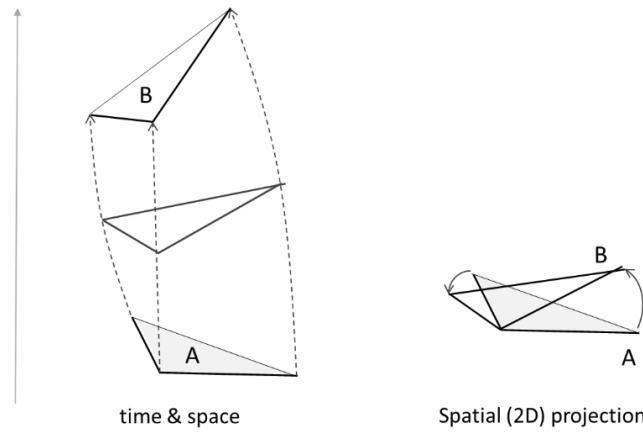


Fig. 6. The rotation of a shape yields non-planar faces.

III. EXPERIMENTAL STUDY

This section presents the results of an empirical study to evaluate and compare the methods presented in section II. The main assumptions considered during the design of the experimental study were: (1) the spatiotemporal data representations should minimize deformation and create a realistic transformation of the objects between observations; and (2) the spatial projection of a spatiotemporal value at any time instant between observations must be a valid shape (edge intersections are not allowed). The former is referred to as fidelity of spatiotemporal data representations. This study uses synthetic and real data. The former are used to put in evidence qualitative features that could be difficult to find in real data.

A. Experimental data and algorithms

Synthetic data are composed of twelve user-defined 2D planar shapes, including convex polygons such as triangles or pentagons and concave polygons like stars and other shapes resembling letters ('U' and 'T'), a table, a chair and a dog, with at most 20 vertices.

Real data were obtained from two sequences of satellite images tracking the location and the shape of icebergs over time:

- *Ross*: a sequence of thirty images tracking Iceberg *B-15* during 2000-2001. The time intervals between consecutive observations are irregular and vary from 1 to 29 days. The movement of the iceberg is predominantly translational but there are some parts where rotation is also significant.
- *B-15*: a sequence of ten images of two large blocks of iceberg *B-15* captured between November 19, 2004 and December 20, 2004. The predominant movement of the largest block (*B-15a*) is translation, while for the smaller one (*B-15j*) is rotation.

The original images, from which were extracted the data used in this study, are available in [35,36]. The procedures used to extract the icebergs' shape and to create the vertex correspondences between shapes is described in [31]. Some images have been discarded due to the presence of clouds.

Two morphing algorithms are compared:

- The first implements the rotating plane (*RPlane*) algorithm, as proposed in [25]. The algorithm proposed in [24] was excluded because of the matching problem between concavities described in section II.A.
- The second implements the algorithm based on compatible triangulations (*TMorph*) presented in Section Fig. 3, which uses normal equations to compute the position of shared vertices.

B. Evaluation of qualitative features using synthetic data

The design choice underlying the approaches presented in section II is quite different. While the approach presented in II.A represents translation and rotation implicitly, these spatial transformations are represented explicitly in Fig. 3. Both design choices have no impact on the representation of the objects' movement (translation). However, the implicit representation of rotation through linear movements of the shapes' edges used in *RPlane* algorithm may cause important deformations. Fig. 7 displays the variation of the area of a rigid object during a rotation between consecutive observations.

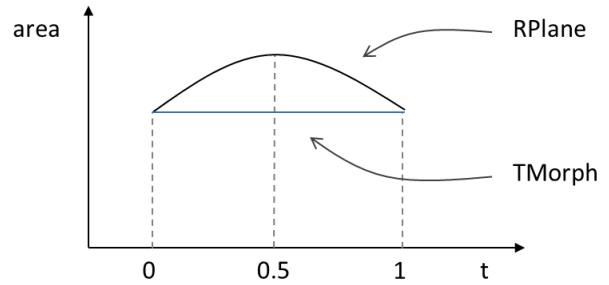


Fig. 7. Variation of the area during the rotation of a rigid object between consecutive observations.

In the case of *RPlane*, the area of the shape increases until the middle and decreases during the second half of the interpolation. The horizontal line represents the object's area

during an interpolation created using the *TMorph* algorithm. The area does not change, as expected for a rigid object. This is the typical behavior of *RPlane* algorithm with rotating objects, since they tend to get a squared shape in the middle of interpolation as illustrated in Fig. 8.

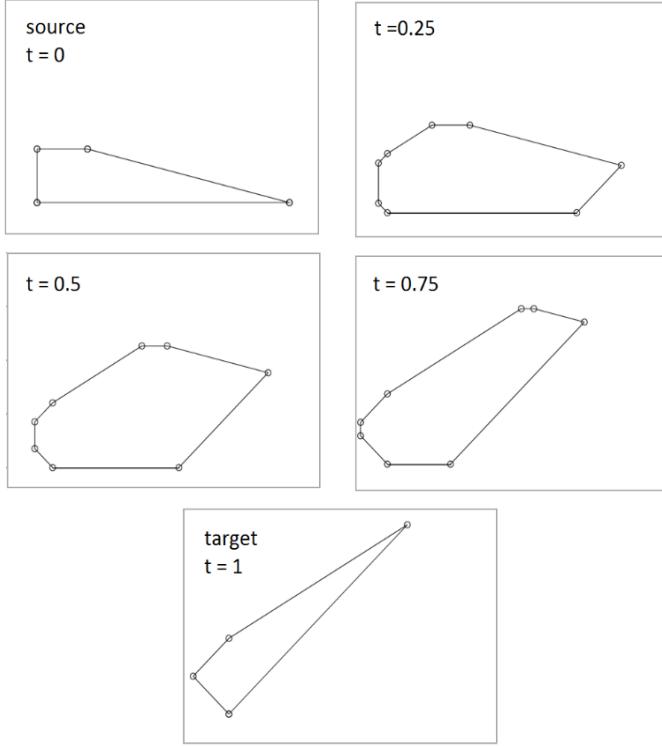


Fig. 8. Deformation of a rigid object during a rotation using the rotating plane algorithm.

C. Experiments with real data

To evaluate whether the spatiotemporal data representations are realistic, we use the following similarity measure:

$$\text{Similarity} = 1 - \frac{A(P_i \cup E) - A(P_i \cap E)}{(P_i \cup E)} \quad (6)$$

where $A(x)$ is a function that returns the area of a polygon x ; P_i is a polygon in a sequence of observations that represents the shape D_i of the object (iceberg) captured at time t_i ; and E is the object's shape at t_i , estimated using one of the algorithms investigated in this work from observations $D_{i-1} = (t_{i-1}, P_{i-1})$ and $D_{i+1} = (t_{i+1}, P_{i+1})$. Since we are interested in investigating the deformation of spatial objects with extent, the shapes P_i and E are aligned using a translation so that the centroids coincide, followed by a rotation that is calculated using a function that minimizes the sum of squared errors considering the distance between the corresponding vertices in both polygons. This procedure is illustrated in Fig. 9.

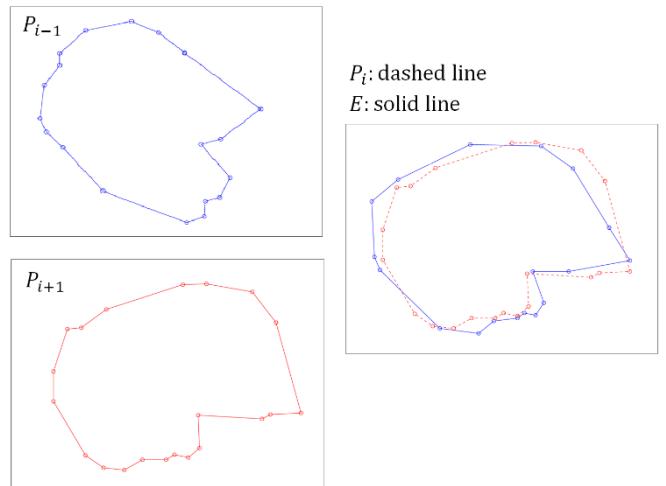


Fig. 9. Similarity between a captured P_i and an estimated E shape.

The results are summarized in TABLE I. The similarity values are the averages by algorithm and dataset.

TABLE I. SIMILARITY VALUES

Algorithm	Dataset		
	Ross	B-15a	B-15j
<i>RPlane</i>	0.86	0,92	0,83
<i>TMorph</i>	0,89	0,90	0,89

The best results were obtained for *B-15a*. This is a large iceberg, which has no important concavities and the rotation between snapshots is nearly nonexistent. The worst results occurred for *B-15j*. This is an iceberg with a large concavity and significant rotation between snapshots. In this case, the results obtained using *TMorph* are close to the results obtained for *B-15a*, but the similarity values using *RPlane* show that there is significant deformation during interpolation. This deformation is mainly caused by difficulties in handling rotation and by the method used to deal with concavities. The data used in this study did not allow to investigate the relative importance of each of these factors in deformation, but Fig. 10 shows an illustrative example.

The shape estimated in the middle of the transformation ($t = 0.5$) has a larger size than the source and target shapes, and the concavity tends to vanish. The former is due to the rotation of the shape and the latter comes from the mapping of all vertices in a concavity to a single vertex in the convex-hull of the other shape. A similar transformation created using *TMorph* is illustrated in Fig. 11.

As depicted in Fig. 10 and Fig. 11, *TMorph* can handle rotation and concavities better than *RPlane*, and is able to create transformations that are visually more natural. This is also in accordance with the numerical results just presented.

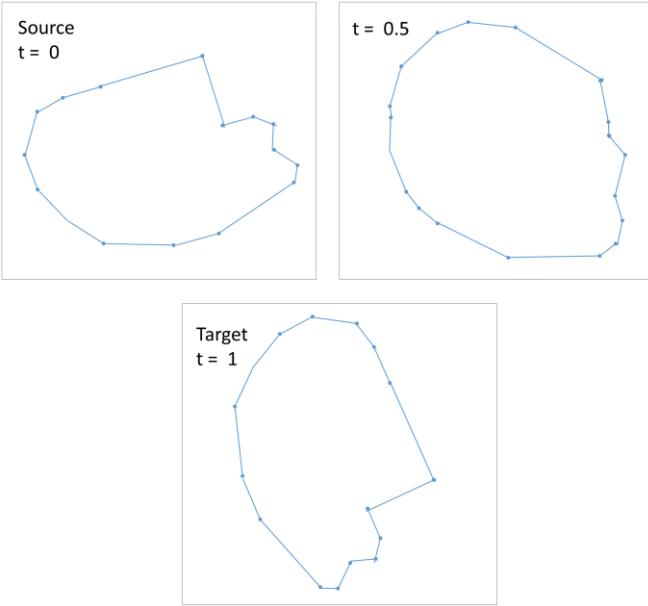


Fig. 10. Estimation of the shape of *B-15j* using the *rotating plane* algorithm.

The results obtained for *Ross* are in between the results for *B-15a* and *B-15j*, since there are periods where the rotation of the iceberg between observations is considerable and almost nonexistent in other periods.

In absolute terms, the margin of improvement for *TMorph* is between 10 and 11%, and for *RPlane* is between 8 and 17%. There are several factors contributing to these errors, such as, noise in source images, approximation errors due to the segmentation of source images or bad vertex correspondences, besides the interpolation algorithms studied here. For instance, in the case of icebergs, there are numerous ice fragments around the icebergs and in certain images it is difficult to distinguish the boundaries of the main block unambiguously. In addition, a large fragment of the *B-15a* collapsed during the sampling period, and so, the similarity value obtained for the time interval between the observations immediately before and after the collapsing was lower than in the other cases.

D. Topological and geometric features.

Several constraints must hold on the representation of spatiotemporal data. Specifically, the segments defining the geometry of a shape must not intersect during transformations, because this is a topological anomaly in view of spatial data systems currently in use. Only intersections between the segments' end points are allowed.

The spatiotemporal representations created using the algorithms studied in this work were topologically valid at all times, even for *B-15j*, which is the iceberg with the most complex geometry. For synthetic data, there also were no segment intersections in the spatiotemporal data representations created using *RPlane*. This means that mapping all edges in a concavity into a single point in the convex-hull of another shape was enough to create topologically safe spatial transformations at all times. However, this strategy may cause important deformations.

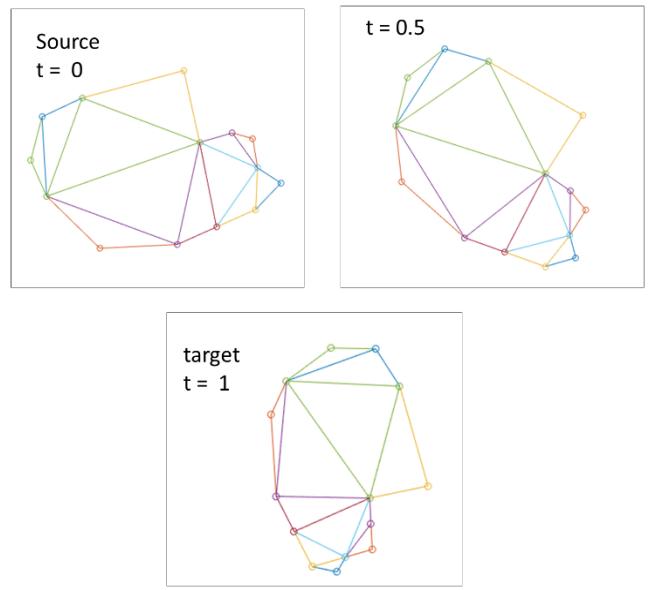


Fig. 11. Estimation of the shape of *B15j* using compatible triangulations.

The interpolations created using *TMorph* are not completely safe, and topological anomalies may arise in the interpolation of very dissimilar shapes. The mesh structure is important because topological anomalies occurred more often when there are thin triangles, highlighting the importance of the post-processing step to improve the mesh of source and target shapes. The incidence of topological anomalies also tends to decrease with the number of vertices, i.e., the results indicate that it is possible to increase the robustness of the algorithm adding Steiner vertices into the triangulated shapes. Note that, the occurrence of these topological anomalies depends on several factors. For instance, it is difficult to define a vertex correspondence between highly dissimilar shapes (e.g., a table and a car), and so, the results may be ambiguous. As it is difficult to create a coherent suite of test cases, we just present an example of a topological anomaly generated using a mesh before and after refinement (Fig. 12), rather than numerical results.

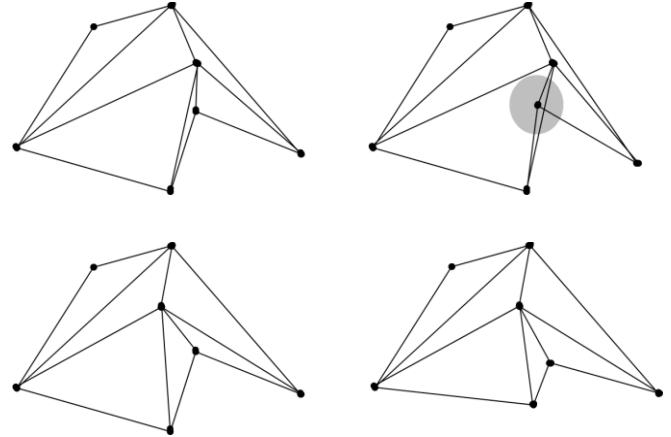


Fig. 12. Source polygon with a thin triangle (top-left) and topological anomaly (edges intersection) during a transformation (top-right); the same transformation after mesh refinement (bottom).

From a geometrical point of view, the projection of a spatiotemporal value at any time instant returns a planar shape using any of the algorithms studied in this work. However, while the projection of a spatiotemporal value created using the *RPlane* algorithm during a time interval between two consecutive observations returns always a planar face (Fig. 1), *TMorph* may produce non-planar faces (Fig. 6). Although the latter allows creating spatiotemporal representations that are more realistic, the algorithms to deal with this kind of geometries are more complex.

IV. DISCUSSION

The results presented in previous section provide interesting insights on the development of solutions to represent spatiotemporal phenomena in information systems. Two main approaches are presented. The first, implements the rotating plane algorithm [24,25]. This algorithm is able to create spatial transformations that are safe at all times but is poor in handling rotations. There is no previous matching between shapes in a sequence of observations and the correspondence between the first pair of edges is based on simple rules, such as, selecting the edges with lowest angles relatively to a given coordinate axis. Thus, to ensure that the results are realistic it is necessary that the time interval between snapshots is sufficiently small so that rotation between them is also small, which might not be possible in some use cases.

The second approach used in this work is based on [27,28,31], which requires defining a vertex correspondence between a source and a target shape beforehand. The interpolation is split into components, and so, rotation is represented explicitly in the model, allowing to create spatiotemporal representations that are more realistic than with the previous approach. In addition, the use of triangulation is an interesting approach, because there are many algorithms in computational geometry based on triangulation and the implementation of complex spatiotemporal operations using this kind of decomposition can be simplified. In this work, we assumed that translation and rotation are linear and uniform, and so, the focus was on the morphology of objects with spatial extent. However, splitting spatial transformations into components, enables the modeling of translation and rotation using special-purpose functions, which may take into account external factors such as ocean currents, or wind direction and speed.

The results of this study show that there is not a best solution to represent the evolution of spatiotemporal phenomena over time. While the solution using the *RPlane* tries to ensure that data is topologically valid at all times, *TMorph* generates more realistic representations of spatiotemporal phenomena. It is also important to note that there is no formal proof that the algorithms are safe, i.e., that the spatial transformations created using an algorithm are topologically valid at all times for any pair of source and target shapes. Thus, validation is usually made by example.

The combination of both strategies is possible: first, we can use *TMorph* to create an interpolation based on the observations; second, we split the spatial transformation into smaller time intervals to create pseudo-observations that are closer to each

other; and third, we can use *RPlane* and the pseudo-observations to create the interpolations. The first step would allow creating a more realistic representation of the real-world phenomena and the last step would replace a representation producing non-planar faces by another algorithm producing planar faces only. As pseudo-observations are closer to each other, large rotations between observations were broken into several smaller rotations, thus reducing deformation caused by the *RPlane* algorithm.

V. CONCLUSION

This paper focuses on the representation of the evolution of spatial phenomena over time, using continuous models of space and time. Although the case study is about monitoring icebergs in the Antarctic, the aim is to investigate generic solutions that may be used in several application domains, e.g., earth sciences, medicine or biology.

Tackling the problem of creating spatiotemporal data from a sequence of observations taken at discrete times, this paper investigates two interpolation algorithms. The first uses a rotating plane algorithm to represent spatial transformations between consecutive observations, as proposed in [25]. Although this is the main approach to represent spatiotemporal phenomena in databases, the experiments performed using real data put into evidence issues on dealing with rotation and concavities. The other algorithm splits the representation of spatial transformations between observations into components [27,28], namely, translation, rotation and deformation, and uses compatible triangulations of source and target polygons [31]. This algorithm is better when dealing with rotation and concavities, producing a more realistic morphing, but the interpolations are not safe since topological anomalies may occur during morphing. These anomalies occurred only with experiments using synthetic data, where source and target polygons were highly dissimilar. It is also important to note that only the first algorithm is compatible with current data models proposed in spatiotemporal databases.

Despite being an important tool in several application domains, there still are important open issues regarding spatiotemporal data processing. The support for developing applications is still limited, and it is often necessary to make great efforts on the development of special-purpose algorithms to implement complex spatiotemporal operations. In the future, it would be interesting to have generic tools, where approximation errors are bounded, and high-level query languages to make the development of automated processes for storage, management, interpretation and analysis of spatiotemporal data easier. The use of morphing techniques sounds a promising approach to achieve these goals. A good solution should create realistic transformations that are topologically valid at all times, using compact representations and simple procedures to enable retrieving large amounts of spatiotemporal data efficiently. Morphing techniques based on computationally intensive algorithms, e.g., iterative algorithms, are not a good choice.

The use of real and synthetic data was important, since both have enabled finding different issues on the representation of spatiotemporal data. The creation of benchmarks to enable a

comprehensive evaluation of key properties of spatiotemporal data management systems, is also an interesting research topic in this area.

ACKNOWLEDGMENT

This work is partially funded by National Funds through the FCT - Foundation for Science and Technology, in the context of the project UID/CEC/00127/2013.

REFERENCES

- [1] N. Mamoulis, Spatial data management, Morgan & Claypool Publishers, Synthesis Lectures on Data Management, 2012.
- [2] P. Rigaux, M. Scholl and A. Voisard, Spatial databases with applications. The Morgan Kaufmann Series in Data Management Systems, 2002.
- [3] M. Koubarakis et al., Spatiotemporal databases: The ChoroChronos Approach, Springer–Verlag, Lecture Notes in Computer Science, 2003.
- [4] M. Bügelmayer, D. M. Roche and H. Renssen, Representing icebergs in the iLOVECLIM model (version 1.0) – a sensitivity study, Geoscientific Model Development, vol. 8, pp 2139–2151, July 2015.
- [5] Z. Zhao, S.-L. Shaw and D. Wand A Space-Time raster GIS data model for spatiotemporal analysis of vegetation responses to a freeze event, Transactions in. GIS, vol. 19, pp 151–168, February 2015.
- [6] E. Meijering, O. Dzyubachyk and I. Smal, Methods for cell and particle tracking, Methods Enzymol vol. 504, 183–200, Elsevier, 2012.
- [7] X. Silvani, F. Morandini and J.L.Dupuy, Effects of slope on fire spread observed through video images and multiple-point thermal measurements, Experimental Thermal and Fluid Science, vol. 41, 99–111, September 2012.
- [8] D. Casbeer, D. Kingston, R. Beard and T. McLain, Cooperative forest fire surveillance using a team of small unmanned air vehicles, International Journal of Systems Science, vol. 37, pp. 351–360, June 2006.
- [9] L. Merino et al., An unmanned aircraft system for automatic forest fire monitoring and measurement. Journal of Intelligent, Robotics Systems, vol. 65, pp. :533–548, January 2012.
- [10] P. Naden, Siltation in Rivers: A Review of Monitoring Techniques. Conserving Natura 2000 Rivers Conservation Techniques, Series no. 6, March 2015.
- [11] T. Nelson, J. Long, K. Laberee and B. Stewart, A time geographic approach for delineating areas of sustained wildlife use, Annals of GIS, vol. 21, pp 81–90, May 2015.
- [12] T. Nelson and J. Long, Home range and habitat analysis using dynamic time geography, Journal of Wildlife Management, vol 79, pp. 481–490, April 2015.
- [13] G. Hamarneha and T. Gustavsson, Deformable spatio-temporal shape models: extending active shape models to 2D+time, Image and Vision Computing, Elsevier, vol. 22, pp 461–470, 2004.
- [14] M. Way, P. Gazis and J. Scargle, Structure in the 3D galaxy distribution. II. Voids and watersheds of local maxima and minima, The Astrophysical Journal, vol. 799 (95), 24pp, January 2015.
- [15] S. Grumbach, P. Rigaux and Luc Segoufin, Manipulating interpolated data is easier than you thought, in Proceedings of the 26th International Conference on Very Large Data Bases (VLDB '00), Cairo, Egypt, pp 156–165, 2000.
- [16] R. Güting, M. Böhlen, M. Erwig, C. S. Jensen, N. Lorentzos, M. Schneider, and M. Vazirgiannis, A foundation for representing and querying moving objects, ACM Transactions on Database Systems, vol. 25, pp 1–42, March 2000.
- [17] J. A. Lema, L. Forlizzi, R. Güting, E. Nardelli, M. Schneider. Algorithms for Moving Objects Databases, Computer Journal, vol. 46, pp 680–712, June 2003.
- [18] X. Meng and J. Chen, Moving objects management: models, techniques and applications, Springer–Verlag, 2010.
- [19] L. Matos, J. Moreira, A. Carvalho. A spatiotemporal extension for dealing with moving objects with extent in Oracle 11g, ACM SIGAPP Applied Computing Review, vol. 12, pp 7–17, June 2012.
- [20] L. Zhao, P. Jin, X. Zhang, L. Zhang and H. Wang. STOC: extending oracle to support spatiotemporal data management, In Proceedings of the 13th Asia-Pacific web conference on Web technologies and applications, pp 393–397, 2011.
- [21] N. Pelekis, Y. Theodoridis, S. Vosinakis and T. Panayiotopoulos, Hermes: A Framework for Location-Based Data Management, in Proceedings of the 10th International Conference on Advances in Database Technology, Saint-Petersburg, Russian Federation, pp 1130–1134, 2006.
- [22] S. Gebberta and E- Pebesma, A temporal GIS for field based environmental modelling, Environmental Modelling & Software, vol. 53, pp 1–12, March 2014.
- [23] W. Siabato, C. Claramunt, M. Manso-Callejo and M. Bernabé-Poveda, TimeBibliography: a dynamic and online bibliography on temporal GIS, Transactions in GIS, vol. 18, pp 799–816, March 2014.
- [24] E. Tøssebro, R. Güting, Creating Representations for Continuously Moving Regions from Observations, in Proceedings of 7th Symposium on Advances in Spatial and Temporal Databases, Redondo Beach, CA, USA, 321–344, 2001.
- [25] M. McKenney and R. Frye, Creating moving regions from snapshots of complex regions, ACM Transactions on Spatial Algorithms and Systems, vol 1, pp 4.1–4.30, July 2015.
- [26] H. Liu and M. Schneider, Tracking continuous topological changes of complex moving regions, in Proceedings of the 2011 ACM Symposium on Applied Computing (SAC '11), Taichung, Taiwan , pp 833–838, 2011.
- [27] W. Baxter, P. Barla and K-I. Anjyo, Rigid shape interpolation using normal equations, in Proceedings of the 6th International Symposium on Non-Photorealistic Animation and Rendering (NPAR '08), Annecy, France, pp 59–64., 2008.
- [28] M. Alexa, D. Cohen-Or and D. Levin, As-rigid-as possible shape interpolation, in Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH'00, New Orleans, Louisiana, USA, pp 157–164, 2000.
- [29] D. Xu, H. Zhang, Q. Wang and H. Bao. Poisson shape interpolation, in proceedings of the ACM Symposium on Solid and Physical Modelling, Boston, MA, USA, pp 267–274, 2005.
- [30] R. Sumner, M. Zwicker, C. Gotsman and J. Popović, Mesh-based inverse kinematics, in Proceedings of the 32nd International Conference on Computer Graphics and Interactive Techniques, SIGGRAPH'05, Los Angeles, CS, USA, pp 488–495, 2005.
- [31] V. Surazhsky and C. Gotsman, High quality compatible triangulations, Engineering. with Computers, vol. 20, pp 147–156, July 2004.
- [32] J. Moreira, P. Dias and L. Paulo, Creating moving objects representations for spatiotemporal databases, Encyclopedia of Information Science and Technology, Third Edition, pp 1703–1709, 2015.
- [33] O. Kaick, H. Zhang, G. Hamarneh and D. Cohen-Or, A Survey on Shape Correspondence, Computer Graphics Forum vol. 30, pp 1681–1707, September 2011.
- [34] L. Liu, G. Wang, B. Zhang, B. Guo and H.-Y. Shum, Perceptually based approach for planar shape morphing, in Proceedings. 12th Pacific Conference on Computer Graphics and Applications, Seoul, South Korea, pp. 111–120, 2004.
- [35] Rutherford Appleton Laboratory, The ATSR project [online], available URL: <http://www.atsr.rl.ac.uk/images/sample/ross/index.shtml>, accessed May 2016.
- [36] Eosdis earthdata, Ross sea subset, terra 1km true color, [online], available https://lance.modaps.eosdis.nasa.gov/imagery/subsets/?project=antarctic_a&subset=RossSea.2004324.terra.1km, accessed May 2016.

An n -gram cache for large-scale parallel extraction of multiword relevant expressions with LocalMaxs

Carlos Goncalves

ISEL – Instituto Superior de Engenharia de Lisboa
 IPL – Instituto Politécnico de Lisboa
 1959-007 Lisboa, Portugal
 cgoncalves@deetc.isel.pt

Joaquim F. Silva, Jose C. Cunha

NOVA Laboratory for Computer Science and informatics
 Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa
 2829-516 Caparica, Portugal
 jfs@fct.unl.pt, jcc@fct.unl.pt

Abstract—LocalMaxs extracts relevant multiword terms based on their cohesion but is computationally intensive, a critical issue for very large natural language *corpora*. The *corpus* properties concerning n -gram distribution determine the algorithm complexity and were empirically analyzed for *corpora* up to 982 million words. A parallel LocalMaxs implementation exhibits almost linear relative efficiency, speedup, and sizeup, when executed with up to 48 cloud virtual machines and a distributed key-value store. To reduce the remote data communication, we present a novel n -gram cache with cooperative-based warm-up, leading to reduced miss ratio and time penalty. A cache analytical model is used to estimate the performance of cohesion calculation of n -gram expressions, based on *corpus* empirical data. The model estimates agree with the real execution results.

Keywords—Large Corpora; Statistical Extraction; Multiword Terms; Parallel Processing; n -gram cache; Performance Evaluation; Cloud Computing

I. INTRODUCTION

Documents can be well summarized by relevant expressions capturing their core contents, and then used for indexing. LocalMaxs [1] is a language-independent, statistical method, which for a given *corpus* and an n -gram size ($n \geq 2$) extracts n -grams that can be classified as relevant expressions (RE), based on the internal cohesion (or glue) sticking together the words within an n -gram and a relevance criterion that compares the internal cohesion with the cohesion of its adjacent n -gram neighbors. The main advantage of LocalMaxs lies on its neutrality concerning language, syntax and semantics. Its precision and recall are in the order of 75 % for *corpus* sizes beyond 10 million words (Mword).

Sequential LocalMaxs implementations [1] are typically unable to process large *corpora*, typically beyond a few hundred million words, due to memory and time limitations. These limitations can be overcome by parallel and distributed computing. ParLocalMaxs is a parallel implementation of LocalMaxs focused on the study of the possibilities to extract RE from very large *corpora* with varied n -gram sizes. This approach is based on a decoupling of LocalMaxs in three main phases: Counting of all n -gram occurrences in the input *corpus* (phase 1); Calculating the cohesion of all distinct n -grams (phase 2); Selecting the relevant n -grams (phase 3). Each phase exploits parallelism to process multiple data partitions. The approach relies on a distributed in-memory key-value store

for the intermediate data, including all the n -gram frequency counts, and their cohesion values. By exhaustively computing all the n -gram occurrences in a *corpus* for the counting and glue calculation, this ensures the same precision and recall values as LocalMaxs. We developed an implementation [2] which for a cloud environment with up to 54 machines, was able to extract relevant 2-grams and 3-grams from English *corpora* up to 1 Gword, achieving almost linear behavior for the relative speedup and sizeup metrics. However, due to the exhaustive computation requirement, the total execution time is dominated by the cohesion calculation in phase two (over 75 %) due to the intensive access to the distributed key-value store. To reduce this communication-intensive behavior, we designed a cache system for n -grams, which tries to capture the temporal locality of the accesses to the n -gram occurrences in the *corpus* during the algorithm execution. To understand the patterns of repetition of n -grams in natural language *corpora*, and guide the n -gram cache design, we conducted an empirical study of the distribution of n -grams in a range of *corpora* from 2 to 982 Mword in the English and French languages. The experimental results presented in this paper show that, for a range of *corpora* from 25 Mword up to 682 Mword and 2-gram and 3-gram RE, the use of this cache system resulted in a significant reduction (in the order of 55 %) in the global execution time of the ParLocalMaxs algorithm, due to the obtained reduction in the order of 70 % in phase 2 execution time, which is the focus of this paper.

In the paper we present an analytical model to estimate the n -gram cache performance behavior (miss ratio and miss time penalties) as a function of the *corpus* size, the n -gram size, and the number of machines. The model estimates agree with the experimental results from the real execution. We also discuss the relationship between the distinct n -gram distribution for very large *corpora* sizes and the expected performance behavior of the parallel algorithm. Due to the finiteness of existing language vocabularies, we observe that, for each n -gram size, the evolution of the numbers of distinct n -grams as a function of the *corpus* size, tend to a *plateau*.

After discussing the background and related work (Section II), this paper presents an empirical study on n -grams distribution in natural language *corpora* (Section III), a model for glue calculation (Section IV) with an n -gram cache (Section

V), a comparison of the model estimates with the real execution experiments (Section VI), a global discussion (VII) and conclusions (Section VIII).

II. BACKGROUND AND RELATED WORK

Relevant expressions can be extracted by statistical, linguistic or hybrid approaches. The latter two [3–5], [6–8] need specific language information and impose a linguistic dependency, and relevancy is not completely determined by morphosyntactic patterns. Metrics for several statistical approaches were compared in [9], but only for 2-grams, i.e. sequences of two consecutive words. The main advantage of LocalMaxs lies on its neutrality concerning language, syntax and semantics. Furthermore, the method is the same irrespectively of the n -gram size. A detailed comparison of LocalMaxs with other statistical methods is presented in [10]. The LocalMaxs method is defined as follows.

Definition 1. $SCP_f(\cdot)$ [1] is a cohesion metric to calculate the glues of n -grams:

$$SCP_f(w_1 \dots w_n) = \frac{f(w_1 \dots w_n)^2}{\sum_{i=1}^{n-1} f(w_1 \dots w_i) \cdot f(w_{i+1} \dots w_n)} \quad (1)$$

where $f(w_1 \dots w_n)$ is the frequency of the n -gram $(w_1 \dots w_n)$ in the *corpus*. The denominator has the frequencies of all leftmost and rightmost sub n -grams of sizes from 1 to $n-1$ contained in $(w_1 \dots w_n)$.

For example there is a strong cohesion between the n -grams within the n -gram “European Court of Human Rights”, whose relative frequency $f(\text{“European Court of Human Rights”})$ is calculated as the number of occurrences of this 5-gram divided by the total number of 5-gram occurrences in the *corpus*. The leftmost and rightmost sub n -grams, whose frequencies are needed for the glue calculation of the above 5-gram, are: two 1-grams (“European”, “Rights”); two 2-grams (“European Court”, “Human Rights”); two 3-grams (“European Court of”, “of Human Rights”); two 4-grams (“European Court of Human”, “Court of Human Rights”).

Definition 2. Let $W = (w_1 \dots w_n)$ be an n -gram and $g(\cdot)$ a generic cohesion metric. Let $\Omega_{n-1}(W)$ be the set of $g(\cdot)$ values for all contiguous $(n-1)$ -grams within W ; Let $\Omega_{n+1}(W)$ be the set of $g(\cdot)$ values for all contiguous $(n+1)$ -grams containing W ; according to LocalMaxs, W is relevant if and only if:

$$\begin{aligned} & \forall_x \in \Omega_{n-1}(W), \forall_y \in \Omega_{n+1}(W) \\ & \text{length}(W) = 2 \wedge g(W) > y \quad \vee \\ & \text{length}(W) > 2 \wedge g(W) > \frac{x+y}{2} \end{aligned} \quad (2)$$

Definition (2) reflects how an n -gram glue stands out with respect to its neighborhood. Parallelism has been exploited for extraction, e.g. frequent itemsets or pattern mining [11–17]. MapReduce [18] has been used for data intensive text

processing [19] and n -gram statistics [20]. It can be used directly in LocalMaxs phase 1 [21], but lacks support for multiphase applications and per-phase intermediate communications [22]. We use a workflow [23] to express the three LocalMaxs phases and we explicitly specify the intermediate access to the leftmost and rightmost sub n -grams in phase 2, and the Ω sets in phase 3. Distributed in-memory key-value stores, as Memcached [24] or Cassandra [25], represent a widely used approach to handle scalability when large datasets are involved. ParLocalMaxs uses a distributed in-memory key-value-store (KVS) with similar functionalities as the above, for implementing the n -gram tables that keep all distinct n -grams counts and glue values, required by the RE extraction. Statistical studies on the history of word occurrences in natural language *corpora* have been made in different contexts: in particular cache-based n -gram models for linguistic applications were proposed by [26]. The relationships between the Zipf-like [27] distributions and caching for Web search engines have also been extensively discussed [28,29]. Balkir *et al.* [30] also uses an in-memory key-value store for n -gram frequency tables accessed through a Bloom filter and a cache. A distinctive aspect of our work is that we explored the properties of n -gram distribution in natural language *corpora* by an empirical study to improve the performance of a statistical extraction algorithm, and integrated the n -gram cache concept with a parallel implementation and a distributed in-memory key-value store for the n -grams. Also, the proposed n -gram cache, although designed in the context of LocalMaxs, can be used by other n -gram based algorithms that require processing of large *corpora* and distributed n -gram stores. All phases of ParLocalMaxs exploit data parallelism to reduce the execution time, and exploit data distribution to process large *corpora*. Both are supported by a distributed collection of virtual machines (VM) which ensure that the algorithm data structures are fully kept in memory, and also support an in-memory key-value-store, for the n -gram tables that keep all distinct n -grams counts (phase 1) and glue values (phase 2), required by the RE extraction (phase 3). Each VM contains multiple controller processes for executing the LocalMaxs functions, and also contains a KVS server.

Phase 1 corresponds to the n -gram counting problem [15]. Based on a data-parallel and a global aggregation step, it generates the distinct n -gram tables, one for each n -gram size, with the total n -gram counts. These tables are equally partitioned by hashing among the distributed KVS servers, defining the n -gram partitions used by the controllers in phase 2 and 3. During phase 2 the controllers in each VM access the counts of all the leftmost and rightmost sub n -grams contained in each input n -gram whose glue must be calculated (Definition 1). After filtering all the n -grams which occur only once (singletons), the tables resulting from phase 2 are then used by phase 3, which updates them with n -gram relevance Boolean flags. The relevance of each RE, e.g. “grammatical” or “information retrieval”, must be decided by the user for each application. For illustration purposes the set of RE for the *corpora* used in our experiments is available in [31].

III. Corpus CHARACTERIZATION

We considered *corpora* (C) from 2 to 982 Mword, from Wikipedia [32] with topics in the English language, from 1-grams up to 6-grams. The total number of n -gram occurrences of a given size (T_i), is close to the *corpus* size, i.e. the total number of 1-grams: $T_1 = |C|$ is the number of 1-grams; $T_2 = |C| - 1$ is the number of 2-grams; $T_3 = |C| - 2$ is the number of 3-grams, etc. Figure 1 shows the number of distinct n -grams ($|D_i|$, $1 \leq i \leq 6$) in million *versus* the *corpus* size, where D_i is the set of distinct n -grams of size i .

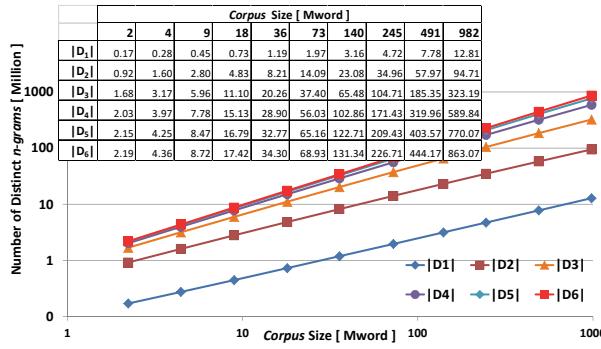


Fig. 1. Distinct n -grams (million) *versus* *corpus* size (Mword) log-log

For relatively small *corpora*, the observed behavior in Figure 1 is apparently linear. However, for each n -gram size, the total number of distinct n -grams $|D_i|$ grows in a logarithmic-like way with $|C|$ when considering the entire *corpus* size range up to infinity (Figure 10). This benefits the scalability of algorithms whose problem size is proportional to the number of distinct n -grams. As this number in a language is strongly constrained by the size of the actual available dictionary, for each n -gram size the ratio $|D_i|/|C|$ decreases with $|C|$, tending to zero as the *corpus* size grows towards infinite. As the *corpus* size grows the n -grams tend to occur more often, with an average repetition factor for each n -gram size, $R_i = T_i/|D_i|$, also increasing with $|C|$. For example, for 1-grams, when $|C| = 2$ Mword we have $R_1 = 13.02$ and when $|C| = 982$ Mword $R_1 = 76.64$. This behavior allows exploring the temporal locality in the n -gram references during glue calculation, captured by a n -gram cache. This benefit is greater for the smaller n -gram sizes because they have higher repetition factors. Furthermore, among all the n -grams of each size there are individual n -grams that occur much more frequently than other. Their distribution exhibits a Zipf-like behavior where a small percentage of n -grams is responsible for most of the n -gram occurrences of the n -grams of a given size. This is useful in applications with frequent n -grams (heavy hitters) as the most popular terms.

IV. MODELING GLUE CALCULATION IN PHASE 2

Phase 2 main components (Figure 2) are the controllers and the KVS servers (with the n -gram table partitions). The $D_{i(j)}$ denotes the table of distinct n -grams of size i ($1 \leq i \leq n$) in VM j ($1 \leq j \leq K$), where K is the number of machines. In the following, we omit the j index whenever $K=1$.

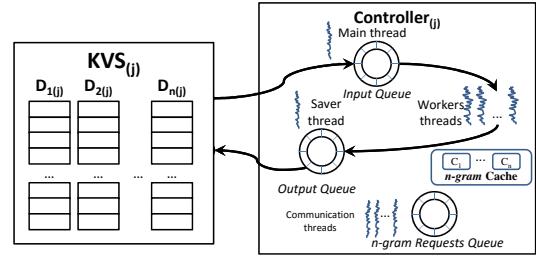


Fig. 2. Per machine Phase 2 Components

A main controller thread inputs the n -grams whose glue must be calculated by a pool of worker threads: i) Identify the sub n -grams within each n -gram; ii) Check an n -gram cache and fetch sub n -grams; iii) Calculate the glue; iv) A saver thread outputs the n -gram glues back to the n -gram table. The input and output are overlapped with the glue calculation. To fetch the needed sub n -grams, the worker thread uses a cache system, composed of an n -gram cache for each n -gram size (C_1, \dots, C_{n-1}).

Glue Calculation Memory References: Phase 2 problem size (N_n) is the total number of glue operations for finding RE for a given range of n -gram sizes. We consider two cases: i) Isolated glue calculation, consisting only of the glue g_i for n -grams of size i , with $i \geq 2$; ii) Combined glue calculation, consisting of all the glues $g_{n_1 \dots n_2}$ for the n -grams from n_1 up to and including n_2 , with $2 \leq n_1 < n_2$. For example, calculating glue $g_{2 \dots 4}$ involves the glues for all 2-grams, 3-grams and 4-grams, $N_n = |D_2| + |D_3| + |D_4|$. For the isolated glue calculation of 2-grams (g_2) $N_n = |D_2|$; for 3-grams (g_3) $N_n = |D_3|$; for 4-grams (g_4) $N_n = |D_4|$. To analyze the references generated we consider as example the glue calculation g_2 where the table $D_{2(j)}$ represents the local partition of the 2-grams in the j^{th} machine, with $1 \leq j \leq K$. The glue for each individual 2-gram depends on the frequencies of its two sub 1-grams. Thus the total number of references to sub 1-grams ($all_{glue_2 Ref(j)}$) in the j^{th} machine is $2|D_{2(j)}|$. Among these references there are repeated n -gram occurrences, which can be filtered out by a cache mechanism, leading to $D_{1 in D_{2(j)}}$, the set of all the distinct leftmost and rightmost 1-grams in the 2-grams found in table $D_{2(j)}$. Table I shows the number of sub n -gram references and the number of distinct sub n -grams for glue calculations g_2, g_3, g_4 .

TABLE I
GLUE REFERENCES

Sub n -grams $all_{glue_i Ref(j)}$	Distinct sub n -grams $D_{t_i(j)}$
g_2 $2 D_{2(j)} $	$ D_{1 in D_{2(j)}} $
g_3 $2 D_{3(j)} + 2 D_{3(j)} $	$ D_{1 in D_{3(j)}} + D_{2 in D_{3(j)}} $
g_4 $2 D_{4(j)} + 2 D_{4(j)} + 2 D_{4(j)} $	$ D_{1 in D_{4(j)}} + D_{2 in D_{4(j)}} + D_{3 in D_{4(j)}} $

For glue g_i , with $2 \leq i \leq 4$, the sub n -grams in left column include all the references to the leftmost and rightmost sub n -grams from sizes 1 up to $i-1$ (as in Definition 1). The right column gives the total number of distinct n -grams among those references. For the combined glue g_{234} the total number of

references to sub n -grams ($all_{glue_{234}Ref(j)}$) is the accumulated sum of the individual references for the included glues, that is $2.|D_{2(j)}| + 4.|D_{3(j)}| + 6.|D_{4(j)}|$.

We consider the work to memory reference ratio ($W_{rm(j)}$) per machine (j), as the number of glue calculations ($N_{n(j)}$) divided by the number of references: $W_{rm(j)} = N_{n(j)} / (2.N_{n(j)} + all_{glue_iRef(j)})$. We obtain $1/W_{rm(j)} = (2.N_{n(j)} + all_{glue_{234}Ref(j)}) / N_{n(j)} = 2 + all_{glue_{234}Ref(j)} / N_{n(j)}$, the number of generated references per glue operation. The first parcel represents two local accesses to the n -gram table partition: one to input the n -gram whose glue must be calculated, and other to output the glue value. The parcel $all_{glue_{234}Ref(j)} / N_{n(j)}$ represents the number of references to the required sub n -grams per glue operation, and it varies only slightly with the *corpus* size. As the number of distinct sub n -grams grows logarithmically with the *corpus* size (section III) then $all_{glue_{234}Ref(j)}$ also grows logarithmically with the *corpus* size, at a similar rate as $N_{n(j)}$. From the empirical analysis of the considered ranges of *corpus*, we observed that for glue g_{234} $all_{glue_{234}Ref(j)} / N_{n(j)}$ has an average value of 5, i.e. an average of 5 references to sub n -grams per glue operation, and for glue $g_{2..6}$, an average value of 7. This very low $W_{rm(j)}$ does not improve significantly by increasing the *corpus* size, because both $N_{n(j)}$ and $all_{glue_iRef(j)}$ increase at similar rates.

A Simplified Model of Glue Calculation Time: Assuming a strict sequential execution in each machine, the phase 2 execution time is $T_{2(j)} = T_{input(j)} + T_{glue(j)} + T_{comm(j)} + T_{output(j)}$, where: $T_{input(j)}$ is the input time; $T_{glue(j)}$ is the glue calculation time, assuming that the required sub n -grams are already local; $T_{comm(j)}$ is the remote sub n -gram fetch time; and $T_{output(j)}$ is the glue value output time. The glue time is: $T_{glue(j)} = N_{n(j)}.t_{gn}$, where t_{gn} is the average time for the glue calculation of any n -gram with size from 2 to n , assuming that all the glue arguments are already local. The communication time is: $T_{comm(j)} = all_{glue_iRef(j)}.f_{(j)}.t_{fetch_n}$, where $all_{glue_iRef(j)}$ is the total number of references to the needed sub n -grams, $f_{(j)}$ is the fraction of those references which are remote (i.e. the miss ratio), and t_{fetch_n} is the average time to remotely fetch any n -gram of size from 1 to $n - 1$. In our case both t_{gn} and t_{fetch_n} were experimentally measured. For a single machine T_2 gives the total phase 2 execution time. For K machines working in parallel, each independently calculates the glue of its local n -gram partition, thus $T_{2(j)}$ gives the phase 2 execution time for each machine, and the total phase 2 execution time is the maximum $T_{2(j)}$ among the K machines. As N_n is equally divided by K machines, in each machine $T_{input(j)}$, $T_{output(j)}$ and $T_{glue(j)}$ are proportional to the local partition size, that is N_n/K . Thus they reduce linearly with K . The glue time is $T_{glue(j)} = N_{n(j)}.t_{gn} = (N_n/K).t_{gn}$. The communication time $T_{comm(j)} = all_{glue_iRef(j)}.f_{(j)}.t_{fetch_n}$. The parcel $all_{glue_iRef(j)}$ reduces linearly with K , as $all_{glue_iRef(j)} = all_{glue_iRef}/K$, while the fraction $f_{(j)}$ grows with K as a power function, as discussed in section V. As a result $T_{comm(j)}$ decreases non linearly with K , as long as there is no network

congestion. By using multiple machines (K) in parallel, we can reduce the time for local glue calculation (T_{glue}) by a factor of K , while T_{comm} also decreases but not so quickly. This leads to a low computation-to-communication ratio:

$$G_{(j)} = \frac{T_{glue(j)}}{T_{comm(j)}} = \frac{N_n}{all_{glue_iRef}} \cdot \frac{1}{f_{(j)} \cdot t_{fetch_n}} \quad (3)$$

Even if we use K machines to speedup the local glue calculation by a factor of K , and also achieving some reduction in the communication time, the $N_n/all_{g_{2..6}Ref}$ ratio for each individual machine would not change as it does not depends on K , and is less than 1. Furthermore, the ratio t_{gn}/t_{fetch_n} is also less than 1, and remained constant in the considered range of machines and *corpus* sizes. On the other hand, the fraction of remote sub n -gram references $f \leq 1$, thus it contributes to increase G . However, a very low value of f would be required in order to compensate the combined effect of the above two factors and lead to G greater than 1. In the following we discuss an approach for significantly reducing the value of f , corresponding to lowering the penalty of the sub n -gram references by transforming most of them into local accesses to an n -gram cache.

V. AN n -GRAM CACHE SYSTEM

The repeated sub n -gram occurrences justify an n -gram cache system in each VM, with one cache for each n -gram size C_1, \dots, C_{n-1} . We consider an infinite cache and analyze the cold or first occurrence misses. We ensure this assumption holds in the considered *corpus* ranges by providing enough local memory to each VM so that all the distinct sub n -grams of each size occurring in each partition can be contained in each cache. Figure 3 shows ($all_{glue_gRef(j)}$) the total number of sub n -gram references generated during the glue calculation (section IV) and how they are distributed among the individual caches ($all_{glue_gRef(j)_{i-gram}}$).

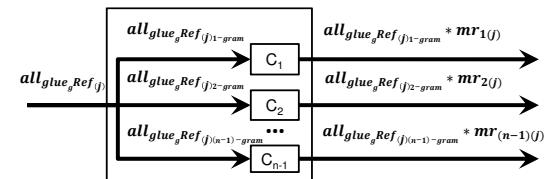


Fig. 3. n -gram cache system

The number of n -gram misses for the cache C_i is obtained as $all_{glue_gRef(j)_{i-gram}}.mr_{i(j)}$, where $mr_{i(j)}$ is the corresponding individual cache miss ratio. This system can be considered globally as a composite cache with a global miss ratio $mr_{(j)}$, whose input is given by $all_{glue_gRef(j)}$, and the output is the total number of missed references given by the sum of the missed references from the individual caches. Thus $all_{glue_gRef(j)}.mr_{(j)} = all_{glue_gRef(j)1-gram}.mr_{1(j)} + \dots + all_{glue_gRef(j)2-gram}.mr_{2(j)} + \dots + all_{glue_gRef(j)(n-1)-gram}.mr_{(n-1)(j)}$. Different number of glue calculation tasks use each cache according to each case,

Table of Contents
◀◀
◀
▶
▶▶

e.g. g_2 only uses C_1 and g_{23456} has 5 tasks using C_1 , 4 tasks using C_2 , etc. A combined glue calculation is preferable as it contributes to an increased cache utilization, reducing the number of remote sub n -gram fetches.

Impact of Cold Misses: We estimate the number of missed references from each cache, which start empty (cold-start), by considering a model where the cold misses are determined by the number of distinct sub n -grams in the n -gram table partitions (D_i for $i > 2$, in Figure 2). For each isolated glue in Table I, the column labeled “Distinct sub n -grams” gives the number of cold-start misses, defining the global miss time penalty when multiplied by t_{fetch_n} , the average time for remotely fetching an n -gram. The corresponding global miss ratio is the ratio of that column by the column “Sub n -grams”.

The Single Machine case ($K = 1$): For a single machine, the total number of distinct 1-grams in the *corpus* is $|D_1| = |D_1 \cap D_i|$, $2 \leq i \leq 6$, and the total number of distinct 2-grams is $|D_2| = |D_2 \cap D_i|$, $3 \leq i \leq 6$, etc. This implies that the number of missed sub 1-gram references to cache C_1 is the same ($|D_1|$) for any of the glue calculations g_i of the n -grams, $i \geq 2$. Thus the cache C_1 individual cold miss ratio for the glue calculation g_i , $2 \leq i \leq 6$, is $mr_{C_1}(g_i) = |D_1| / (2 \cdot |D_i|)$. Likewise the number of cold missed references to cache C_2 is $|D_2|$ for any of the glue calculations, etc.

For each of the n -gram sizes $1 \leq i \leq 5$, and for each glue g_n , $2 \leq n \leq 6$, the miss ratio mr_{C_i} decreases when the *corpus* size increases because the ratio $|D_i| / |D_n|$ also decreases. This is due to the fact that $|D_i|$ grows slower than $|D_n|$ (Figure 1). Also for each of the *corpus* sizes the miss ratio mr_{C_i} decreases as the glue calculation n -gram size increases. For example $mr_{C_1}(g_2) > mr_{C_1}(g_3) > mr_{C_1}(g_4)$ and so on. This is due to the increased reutilization of sub n -grams when increasing the n -gram sizes of the n -gram tables. The estimated global cold miss ratio of the cache system (Figure 3), for a given *corpus* size, for the glue $g_{2\dots 6}$ is: $mr_{cold}(g_{2\dots 6}) = \frac{|D_{t_2}| + |D_{t_3}| + |D_{t_4}| + |D_{t_5}| + |D_{t_6}|}{2 \cdot |D_2| + 4 \cdot |D_3| + 6 \cdot |D_4| + 8 \cdot |D_5| + 10 \cdot |D_6|} = \frac{5 \cdot |D_1| + 4 \cdot |D_2| + 3 \cdot |D_3| + 2 \cdot |D_4| + |D_5|}{2 \cdot |D_2| + 4 \cdot |D_3| + 6 \cdot |D_4| + 8 \cdot |D_5| + 10 \cdot |D_6|}$. The numerator counts the sum of the distinct sub n -gram references, including repetitions, of each glue calculation g_n , $2 \leq n \leq 6$, when considered individually, because in the cold-start scenario each cache starts empty, and the glues are calculated one at a time.

Figure 4 shows the misses for a single machine case as a function of the *corpus* size for the combined glue g_{234} : The global cold miss ratio decreases with the *corpus* size, but the decreasing rate is lower for larger *corpus* sizes; The global miss penalty, i.e. the total number of missed sub n -gram references of the caches (C_1, C_2, C_3), increases with the *corpus* size; The higher n -gram caches have greater influence upon the global miss penalty, concerning the absolute number of misses and the growth rate.

The Multiple Machines case ($K > 1$): Each machine handles the distinct sub n -gram references in its local n -gram tables ($D_{2(j)}, D_{3(j)}$, etc.). Unlike the local n -gram tables, which are equal size partitions, the same behavior does not apply to the sets of distinct sub n -grams ($D_{i \cap D_{n(j)}}$) in

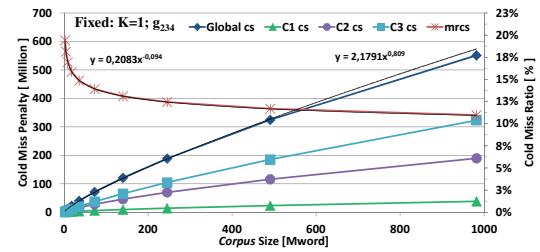


Fig. 4. Cold miss penalty (global, C_i) and global cold miss ratio when $K = 1$

each local partition table $D_{n(j)}$, with $n: 2, 3, 4, \dots$, and $1 \leq i < n$. In fact, for each pair of values of i and n , $|D_{i \cap D_{n(j)}}| > |D_i| / K$, where $|D_i|$ is the total number of distinct sub n -grams of size i in the *corpus*. For example, when $K = 2$ each machine gets a local table of distinct 2-grams ($D_{2(j)}$), whose size is half of the total distinct number of 2-grams ($|D_2|$), while each set $D_{i \cap D_{2(j)}}$ in each machine $j: 1, 2, \dots$, is larger than half of the total number of distinct 1-grams in the *corpus* ($|D_1|$). This is due to the behavior of the n -gram distribution (section III) and reflects the repetition of the most frequent n -grams on each VM. The distribution of the number of distinct sub n -grams per machine ($|D_{i \cap D_{n(j)}}|$) as a function of K , for different values of i and n is shown in Figure 5 for the case $i=1$ when n goes from 2 up to 4: this corresponds to the number of distinct 1-grams ($|D_{1 \cap D_{2(j)}}|$) observed per machine in its local 2-gram table, and to the distinct 1-grams ($|D_{1 \cap D_{3(j)}}|$) in its local 3-gram table, and to the distinct 1-grams ($|D_{1 \cap D_{4(j)}}|$) in its local 4-gram table, when K varies from 1 to 48, for a fixed *corpus* size of 466 Mword.

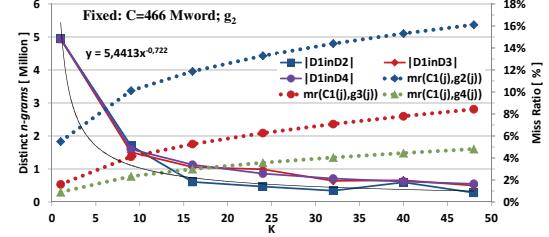


Fig. 5. Distinct 1-grams in D_2 , D_3 , and D_4 and cold miss ratio for cache C_1 per machine for a *corpus* size of 466 Mword

Let us consider the distribution $|D_{1 \cap D_{2(j)}}| = 5.4 \times K^{-0.7}$, as an approximation of the trend¹ in Figure 5. As an approximation to the cold miss ratio of the cache C_1 per machine for the glue g_2 , we have $mr(C_{1(j)}, g_{2(j)}) = |D_{1 \cap D_{2(j)}}| / (2 \cdot |D_{2(j)}|)$. As $|D_{2(j)}| = |D_2| / K$ where $|D_2|$ is the total number of distinct 2-grams, we get $mr(C_{1(j)}, g_{2(j)}) = 5.4 \times K^{-0.7} \times K / (2 \cdot |D_2|) = 5.4 \times K^{0.3} / (2 \cdot |D_2|)$. Thus cache C_1 cold miss ratio increases with the number of machines (K), as shown in the upper dotted curve in Figure 5. Likewise for $mr(C_{1(j)}, g_{3(j)})$ (median curve) and $mr(C_{1(j)}, g_{4(j)})$ (lower curve). For each K the

¹Power trend available in Microsoft® Excel™

cache C_1 cold miss ratio decreases from glue g_2 , to glue g_3 , and to glue g_4 , due to the increasing of the repetition of 1-gram occurrences when the total cache references go from $2|D_2|$, to $2|D_3|$, to $2|D_4|$. Although the cold miss ratio $mr(C_{1(j)}, g_{2(j)})$ increases with K , the time penalty due to the missed 1-grams decreases. On the other hand, the fetching of the remote n -grams by all the machines is made in parallel, that is in overlap mode, leading to a reduction in the overall communication time when compared to when done sequentially. Similar behaviors were observed for the distribution of distinct sub 2-grams, 3-grams, etc, that is $D_i in D_{n(j)}$ for values of i going from 2 up to 5, and the corresponding C_i caches: The per machine global cold miss penalty decreases with K , and the global miss ratio increases with K and tends to stabilize for larger values of K .

Cache Warming up by Combined Glue Calculation: Cache reusability is increased through the n -gram repetitions and also by combining the calculation of the glues of n -grams, from 2 up to n , e.g. g_2 , g_3 and g_4 . The sharing of common sub n -grams is achieved by having multiple controller instances (Figure 2) that in each machine calculate the glues g_2 , g_3 , and g_4 , and access all local caches (C_1 , C_2 , ..., C_{n-1}). The glue calculations of disjoint n -grams sets contained in each of the input tables $D_{2(j)}$, $D_{3(j)}$ and $D_{4(j)}$, require to generate references to the sub n -grams in the input tables ($D_1 in D_{2(j)}$, $D_1 in D_{3(j)}$, $D_1 in D_{4(j)}$, $D_2 in D_{3(j)}$, and $D_2 in D_{4(j)}$), to obtaining the frequency counters stored in the distributed KVS servers. As there are common sub n -grams, there are references shared by the glue calculations. The number of shared sub n -gram references depends on K . For example, when considering cache C_1 for the combined glue g_{234} , the $D_1 in D_n$ sets are not disjoint, i.e., $D_1 in D_2 \cap D_1 in D_3 \cap D_1 in D_4 \neq \emptyset$. When $K=1$ these sets are identical to D_1 which is the distinct set of 1-grams in the entire *corpus*, so all sub 1-gram references (D_1) are completely shared by the g_2 , g_3 and g_4 calculations. So, we can first execute the glue g_2 calculation as a driver to warm-up cache C_1 , and this ensures that there are no 1-gram misses at all, during the glue calculations g_3 and g_4 . However, when $K>1$, the sets of distinct sub 1-grams in each machine are different, $D_1 in D_{2(j)} \neq D_1 in D_{3(j)} \neq D_1 in D_{4(j)}$, due to the way the n -grams tables are distributed by hashing in the end of phase 1. Nevertheless, there still are common 1-grams in the sets $D_1 in D_{2(j)}$, $D_1 in D_{3(j)}$ and $D_1 in D_{4(j)}$. It is desirable to identify such common sub 1-grams because they contribute to increasing the reutilization of cache items once they are fetched, in a similar effect to the cooperative caching strategies. To evaluate the impact of those shared sub n -grams, we conducted an empirical analysis for several *corpus* sizes and number of machines. As a result of this empirical study we confirmed that for $K=1$, $D_1 in D_2 = D_1 in D_3 = D_1 in D_4 = D_1$, the distinct 1-grams in the *corpus*; $D_2 in D_3 = D_2 in D_4 = D_2$, the distinct 2-grams in the *corpus*; and so on. So by first running the glue g_2 (thus filling C_1 with all $D_1 in D_2$) we entirely achieve the effect of warming-up the cache C_1 with the required 1-grams needed by the glue g_3 (that is $D_1 in D_3$) and also g_4 (that is $D_1 in D_4$). Similarly due to the equality

of $D_2 in D_3 = D_2 in D_4$ the cache C_2 is warmed-up by running the calculation of the glue g_3 before starting the calculation of glue g_4 . Note that when starting the calculation of glue g_4 the cache C_1 is already warm due to the previous running of the g_2 and g_3 glue calculations. The overall effect of the above strategy corresponds to greatly reducing the penalty due to the cold misses of all the caches for the n -grams 1 to $n-2$, when calculating the glue for n -grams of size n . Thus the only observed misses are the distinct n -grams of size $n-1$. As observed for the combined glue calculation g_{234} , the above warm-up effect contributes to a significant reduction of the global miss ratio of the cache system when compared to a No-cache configuration.

For $K > 1$, the sets of common sub 1-grams referenced by the different glue calculations g_2 , g_3 , and g_4 depend on the hashing method used in phase 1 for distributing the n -grams among the local partitions (omitted due to lack of space).

The cache warming-up strategy in each machine follows the sequential execution of the glue calculation g_2 , g_3 , etc. Thus the cache C_1 warming-up efficiency can be improved by incremental warm-up, first by the glue calculation g_2 , then by g_3 , then g_4 etc. The implementation supports such incremental warming-up of all the individual caches C_i , $1 \leq i \leq (n-1)$. However, in the model estimates of the warm-up cache behavior, for cache C_i , we only considered the warm-up due to the glue g_{i+1} . For example, the cache C_1 warmed-up by the glue calculation g_2 , and its impact upon the glue calculation g_3 , and also its impact upon the glue calculation g_4 , etc. As a result the model estimates lead to higher values of the misses compared to real execution.

Estimation of Cache Benefits for a Single Machine: For calculating the glues g_2 , g_{23} , g_{234} , g_{2345} and g_{23456} we considered three cases: i) No-cache: calculating the glues when all the sub n -grams references (all_{nRef}) are remote; ii) Cold-start: calculating each glue g_2 , g_3 , g_4 , g_5 , g_6 separately, each starting with its corresponding caches empty; iii) Warm-start: calculating the glues g_2 , g_3 , g_4 , g_5 , g_6 in a strict sequence with warm-start caches, that is: first calculating the glue g_2 ; then calculating the glue g_3 ; yet still keeping the cache C_1 with the previous 1-grams contents; calculating glue g_4 but also keeping caches C_1 and C_2 respectively with the previous 1-grams and 2-grams contents, and so on. Table II shows, for the glue calculations g_2 , g_{23} , g_{234} , g_{2345} and g_{23456} , the global cold miss ratio of the cache system, $mr_{csnc} = cs/nc$, where cs is the number of cold misses, and nc is the total number of generated references, corresponding to the remote references in the No-cache system. Table II also shows the ratio of the warm miss ratio to the cold miss ratio, that is ws/cs , where cs is the number of cold misses, and ws is the total number of misses out of the cache system.

The product of the values in each cell of Table II, for the case $Cs/NoCache$, with each corresponding cell for the case Ws/Cs gives the warm miss ratio $mr_{wsnc} = ws/nc$, i.e. the reduction in the number of missed references from No-cache to a Warm-start cache. Figure 6 shows that the warm miss penalty is less than the cold miss penalty, due to the warm-up of caches

TABLE II
MISSES FOR $ColdStart/NoCache = mr_{cold}$ AND
 $WarmStart/ColdStart$ WHEN $K=1$

C [Mw]	$ColdStartMisses = \frac{Cs}{NoCacheMisses} = \frac{Cs}{NoCache}$				$WarmStartMisses = \frac{Ws}{ColdStartMisses} = \frac{Ws}{Cs}$				
	g2	g23	g234	g2 5	g2 6	g23	g234	g2 5	g2 6
2	0.09	0.15	0.19	0.23	0.26	0.86	0.69	0.54	0.44
4	0.09	0.14	0.18	0.22	0.25	0.87	0.70	0.56	0.45
9	0.08	0.13	0.17	0.21	0.24	0.88	0.71	0.57	0.46
18	0.08	0.12	0.16	0.20	0.23	0.88	0.73	0.58	0.47
36	0.07	0.11	0.15	0.19	0.22	0.89	0.74	0.59	0.48
73	0.07	0.10	0.14	0.17	0.21	0.89	0.75	0.61	0.49
140	0.07	0.10	0.13	0.17	0.20	0.89	0.76	0.62	0.50
245	0.07	0.09	0.12	0.16	0.19	0.89	0.76	0.63	0.51
491	0.07	0.09	0.12	0.15	0.18	0.89	0.77	0.64	0.52
982	0.07	0.08	0.11	0.14	0.17	0.89	0.78	0.65	0.53

C_1 and C_2 . As a result of the glue calculation g_2 there are no 1-gram misses observed during the glue calculation of g_3 and g_4 . As a result of the glue calculation g_3 there are no 2-gram misses observed during the glue calculation of g_4 . Overall, for g_{234} the difference between the cold and warm miss penalties is equal to $2.|D_1| + |D_2|$. This difference increases with the maximum size of the n -gram for which the combined glue calculation is performed. For the combined glue calculation $g_{2\dots n}$ the above difference is given by $\sum_{i=2}^{n-1} (n-i).|D_{i-1}|$ where the $|D_{i-1}|$ represents the reduction in the number of misses observed by cache C_{i-1} , each time is used for each glue calculation from 2 up to n .

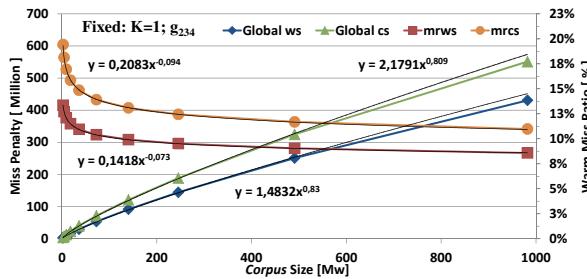


Fig. 6. Single machine global miss penalty and miss ratio: warm vs cold

The total number of warm misses depends on the combined glue calculation of n -grams when n varies from 2 up to 6. This is given by $\sum_{i=1}^{n-1} |D_i|$ which corresponds to the sum of all distinct n -grams from 1-grams up to $(n-1)$ -grams and tends to stabilize for the larger values of n . On the other hand the global warm miss ratio is given by $mr_{ws} = \text{NumberWarmMisses}/\text{all}_{g_{2\dots n}}\text{Ref}$, where $\text{NumberWarmMisses} = \sum_{i=1}^{n-1} |D_i|$ and $\text{all}_{g_{2\dots n}}\text{Ref} = \sum_{i=2}^n 2.(i-1).|D_i|$.

Estimation of Cache Benefits for Multiple Machines: For simplicity we focus on a fixed size *corpus* of 466 Mword, and considered a distinct number of machines (K): 1, 9, 16, 24, 32, 40, 48, and a single combined glue calculation, g_{234} . Figure 7 shows: the number of references to sub n -grams generated per machine (million) in the column labeled *nc* for the case of No-cache; the total number of cold misses per machine in column *cs* for the Cold-start; the total number of warm misses

per machine in column *ws* for the Warm-start. It also shows the reduction in the number of missed sub n -grams from Cold-start to a Warm-start cache (column *ws/cs*); from No-cache to Warm-start cache (column *ws/nc*); and from No-cache to Cold-start (column *cs/nc*). Column *cs/nc* reflects the impact of the distribution of the distinct sub n -grams among the machines when K increases, corresponding to an increase in the cold miss ratio of the cache system per machine. However, that increase is moderate, for example when going from $K=1$ to 48, the cold miss ratio increased only by a factor less than 4. For the glue calculation g_{234} the cache C_1 warming-up by g_2 includes all the distinct 1-grams (D_1 in $D_{2(j)}$), and the cache C_2 warming-up by g_3 includes all the distinct 2-grams (D_2 in $D_{3(j)}$). For $K > 1$, columns *ws* and *cs* reflect the warm and cold miss time penalties per machine when going from $K=1$ to 48, showing a reduction in the time penalty by a factor close to 13, assuming that the average n -gram remote fetch time (t_{fetch_n}) stays constant with K .

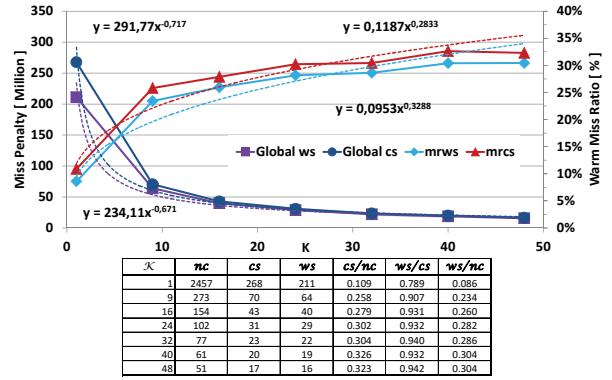


Fig. 7. Per machine global miss penalty and miss ratio: warm vs cold

VI. EXPERIMENTAL PERFORMANCE RESULTS

We experimented with multiple runs of the ParLocalMaxs implementation with distinct configurations, for a Wikipedia English *corpus* [32], with sizes 25, 227, 466, and 682 Mword, to extract all the 2-gram and 3-gram relevant expressions. ParLocalMaxs is implemented in Java, and uses the AWARD [23] workflow framework. All the experiments were conducted in a public cloud (Lunacloud [33]). Each VM was configured with 4 CPU @ 1.5 GHz and a memory configuration ranging from 16 to 90 Gbyte depending on the number of machines used and the *corpus* size, so that the memory required by the algorithm (including the n -gram tables and infinite cache) does not exceed the maximum available VM memory [2]. We experimented with distinct number of VM (K): 1, 9, 16, 24, 32, 40, 48. The execution times presented are averages taken among 3 repeated runs for each experiment.

n-gram Cache Evaluation - Real Execution Results vs Model Estimates: We compared the Cold-start and the Warm-start approaches, and each against a No-Cache configuration. Table III presents the average values of the observed miss ratio for the individual caches C_1 , C_2 , and C_3 (C_1 and C_2 with warm-start) and the global miss ratio when calculating the

glue g_{234} for a selected *corpus* size, 466 Mword. During the execution of the algorithm, in each machine the total number of generated references and the missed references to the local cache system are registered. These values are the average among the K machines. At the end of execution, these values are collected into a log and used for calculation of the per machine miss ratios. We compared the observed data with the cache model estimates.

Figure 8 shows the model estimated miss ratio and miss penalty and the observed values for the glue g_{234} , executed as the sequence $g_2; g_3; g_4$, for each of the involved caches (C_1, C_2 and C_3). The missed 1-gram references in cache C_1 correspond to the cold-start misses when executing g_2 plus the misses when executing g_3 and g_4 . The missed 2-gram references in cache C_2 correspond to the cold-start misses when executing g_3 plus the misses when executing g_4 .

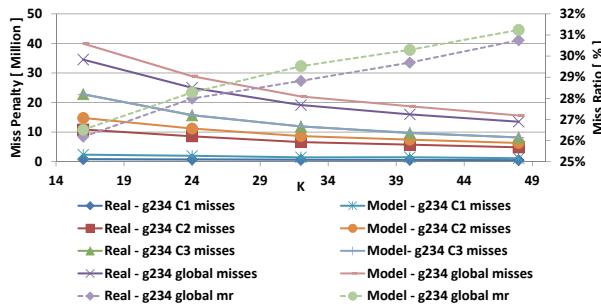


Fig. 8. g_{234} miss penalties (global and C_i): model estimates vs real results for the 466 Mword *corpus*

Phase 2 Real Execution Time and Cache Behavior:

Unlike the assumption made in the simplified model in section IV, in the real execution, phase 2 components of $T_{2(j)}$ (see section IV) overlap in time, being executed by multiple concurrent threads per machine. The experiments confirmed that the remote fetching of the sub n -grams takes most of phase 2 execution time ($T_{comm(j)}$ in $T_{2(j)}$). This agrees with the phase 2 model, as both $T_{input(j)}$ and $T_{output(j)}$ only involve local accesses, and the ratio T_{glue}/T_{comm} is very low (expression 3). Thus $T_2 = Cte + T_{comm(j)}$ more faithfully captures the behavior of the overlapped execution: Cte includes the glue calculation time and is constant for each *corpus* size and number of machines configuration; $T_{comm(j)} = misses_{(j)}.t_{fetch_n}$, where $misses_{(j)}$ is the number of per machine misses for No-cache (nc), Cold-start (cs) or Warm-start (ws), and t_{fetch_n} remained constant in all configurations. Figure 9 presents, for a fixed *corpus* size of 466 Mword, the measured phase 2 execution times for the Warm-start and the Cold-start, compared to the No-cache configuration. The time ratios shown in the three lower lines are approximately equal to the corresponding ratios between the numbers of the observed misses. The ratio $T_{ws}/T_{nc} \simeq mr_{warm}$ comparing the Warm-start cache with the No-cache case has an average value of 30 %, when K goes from 16 to 48 VM.

Total Execution Time vs Phase 2 Execution Time: We present the global results of the execution, with infinite caches



Fig. 9. Phase 2 total times and time reduction: warm vs colds vs no-cache

C_1, C_2 , and C_3 . Table IV presents the absolute execution time ($T_{parallel}$ in minutes) for phase 2 and the entire algorithm (sum of the 3 phases), for each K and *corpus* size C (Mword). Due to lack of space we omit the corresponding tables for phases 1 and 3.

From the No-cache to the Warm-up configuration, the percentage of phase 2 execution time with respect to the total execution time was reduced from an average value (over the configurations of different number of machines and *corpus* sizes) of 78 % in the configuration without cache to an average value of 51 % for the warmed-up cache system. A 70 % reduction in the phase 2 execution time due to the cache warm-up, $T_{ws}/T_{nc} = 0.30$, caused a 55 % reduction in the total execution time ($T_{ws}/T_{nc} = 0.30 \times 0.78/0.51$). The tables show that the observed total execution time is approximately proportional to $1/K$, i.e. with a reduction rate close to linear with K . Thus for each fixed *corpus* size and the above range of machines, the relative speedup, $Sp_{K_1 \rightarrow K_2} = T_{par}(K_1)/T_{par}(K_2)$ (where $T_{par}(K)$ is the total execution time with K machines) is almost linear and the relative efficiency, $E_{K_1 \rightarrow K_2} = (K_1.T_{par}(K_1))/(K_2.T_{par}(K_2))$, is close to 1. For each fixed *corpus* size the relative speedup and efficiency are interpreted relative to the reference minimum number of machines to execute the algorithm due to the ParLocalMaxs memory requirements. The relative sizeup, $Sz_{p_{K_1 \rightarrow K_2}} = N_{par}(T, K_2)/N_{par}(T, K_1)$, is also close to linear: increasing the number of machines by a factor K_2/K_1 allows to increase the problem size ($N_{par}(T, K)$) by the same factor, keeping the total execution time unchanged.

VII. DISCUSSION

Currently in ParLocalMaxs the optimization of the memory space was not a concern, so it is only able to process each given *corpus* size if at least a minimum number of machines ($K_{minimum}$) is provided. Nevertheless, for the considered corpora and machine ranges, the relative parallel efficiency for $K > K_{minimum}$ is almost constant and the relative speedup and sizeup are almost linear. Still, the best known sequential implementation by Luis Gomes², is able to extract RE up to 4-grams from a *corpus* with 330 million words in about 2 hours (when running on a 3.30 GHz Intel® processor, 32 GB RAM, under Ubuntu 12.04.5 LTS), although it is unable to

²<http://research.variancia.com/>

TABLE III
OBSERVED INDIVIDUAL C_1 & C_2 & C_3 MISS RATIO FOR g_{234}

K	C_1 warm-start				C_2 warm-start				C_3 cold-start				Global miss ratio			
	Corpus Size [Mword]				Corpus Size [Mword]				Corpus Size [Mword]				Corpus Size [Mword]			
	25	227	466	682	25	227	466	682	25	227	466	682	25	227	466	682
1	0.00				0.00				0.35				0.11			
9	0.02	0.02			0.26	0.18			0.73	0.65			0.31	0.26		
16	0.03	0.02	0.01		0.33	0.23	0.20		0.76	0.68	0.64		0.34	0.29	0.26	
24	0.04	0.03	0.02	0.01	0.38	0.27	0.23	0.18	0.78	0.70	0.67	0.64	0.36	0.31	0.28	0.26
32	0.05	0.03	0.02	0.02	0.41	0.29	0.24	0.21	0.79	0.71	0.68	0.65	0.38	0.32	0.29	0.27
40	0.06	0.03	0.02	0.02	0.43	0.31	0.26	0.23	0.80	0.72	0.68	0.67	0.39	0.33	0.30	0.28
48	0.06	0.04	0.03	0.02	0.45	0.33	0.26	0.25	0.80	0.73	0.70	0.67	0.40	0.33	0.31	0.29

TABLE IV
TOTAL EXECUTION TIME AND PHASE 2 EXECUTION TIME

$K \setminus C$	Total Execution Time				Phase 2 Execution Time			
	25	227	466	682	25	227	466	682
1	40.1				19.1			
9	6.8	61.6			3.7	32.2		
16	5.0	35.4	69.5		2.6	18.8	36.6	
24	3.7	25.5	42.7	64.0	1.9	13.2	22.1	33.3
32	3.2	19.1	33.7	47.3	1.7	10.1	17.3	24.1
40	3.0	16.2	31.2	40.4	1.5	8.3	15.6	19.4
48	3.0	15.9	26.8	35.8	1.5	8.3	14.1	16.5

process *corpora* much larger than that size. In the current ParLocalMaxs implementation, by using multiple machines, we achieve lower absolute execution times (Table IV) than the sequential implementation, but the obtained absolute speedup regarding the sequential implementation ($\frac{T_{seq}}{T_{par}}$) is sublinear, corresponding to a low absolute parallel efficiency ($\frac{T_{seq}}{K \cdot T_{par}}$). This is due to the optimizations performed within the monolithic single-machine implementation, and to the inefficiencies of the current parallel implementation mainly regarding the communication overheads. However, note that the sequential implementation is unable to process *corpora* beyond a certain size, in the order of a few hundred of million words, suggesting that the parallel implementation is an alternative approach to handle the scalability of LocalMaxs for larger *corpus*. Ideally it will be possible to combine the single-machine optimizations of a sequential version of LocalMaxs with the scalability obtained through parallelism and distribution.

From the conducted empirical analysis (section III), we observed that the n -gram repetition factors increase with the *corpus* size, at different growth rates for each n -gram size. As the *corpora* sizes further increase beyond 1 Gword, the number of distinct n -grams of each size increases monotonically, and gradually tends to a *plateau* that is determined by the finite number of words in the language vocabulary. In [34] we present a theoretical model for estimating the distinct n -gram distribution, based on the Zipf-Mandelbrot Law and the Poisson distribution, which was validated, for any *corpus* size, with the English and French languages. Figure 10 shows the *corpus* size thresholds ($|C|$) beyond which each of the total number of distinct n -grams, from 1-grams up to 6-grams, reaches a *plateau* in the case of the English language. The same figure shows the expected evolution of the total number

of Warm and Cold misses during phase 2 of ParLocalMaxs as a function of the number of machines. In this analysis we assume that the distribution of n -grams per machine is proportional to $1/K$, which corresponds to a simplification of the real trend as discussed in section V. Although the real values would be slightly greater than the presented ones, this simplification still allows us to highlight the overall trend.

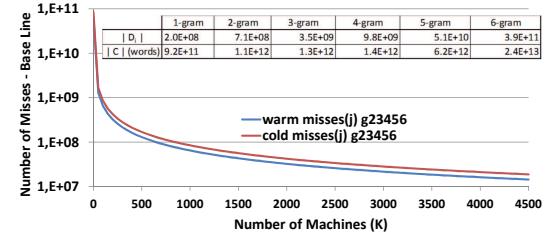


Fig. 10. n -gram plateaux and cache misses base line for glue g_{23456}

The total number of misses decreases as the number of machines (K) increases. The miss decrease rate gets progressively lower with K , leading to number of misses reaching values in the order $1 * 10^8 — 2 * 10^7$ for $1000 < K < 4500$. In order to sustain the infinite cache assumption each machine local memory must contain the n -gram table partitions and the n -gram caches as required by the n -gram plateau values. This depends on the available maximum VM memory configuration and maximum number of available machines. Nevertheless, in the cases where sustaining an infinite cache is not possible, the current implementation is able to handle the cold-start misses and also the cache capacity misses. Under the infinite and Warm-start cache assumption for the plateau corpora ranges, there is an overall reduction of the remote communication of about 98.53 % (as the warm miss ratio is about 1.47 %). This corresponds to the asymptotic constant time complexity reached by phase 2 and 3 for *corpora* beyond the above thresholds. However, the corresponding absolute number of n -gram misses is still very large, and is significant with respect to the problem size (N_n), due to the low value of the computation-to-communication ratio G would be given by $(N_n/WarmMisses) \cdot (t_{gn}/t_{fetch_n}) = 7 \cdot (t_{gn}/t_{fetch_n})$.

VIII. CONCLUSIONS & FUTURE WORK

Extraction algorithms typically exhibits time complexities linear or worse in the number of elements of the input dataset.

ParLocalMaxs is a parallel implementation of LocalMaxs able to process large *corpora*, achieving close to linear relative speedup and size-up in a public cloud using up to 54 virtual machines, for extracting 2-gram and 3-gram RE from *corpora* up to 1 Gword [2], also agreeing Table IV. For the observed *corpus* range, as phase 2 dominates the total execution time, the computation time decreases linearly with K and the n -gram misses communication time also decreases with K as a power function K^{-b} with $b < 1$, as long as there is no network congestion. An n -gram cache reduced the remote data communication, leading to a 70% reduction of the glue calculation, and a 55% reduction in the total execution time, in a cloud with up to 48 VM, for extracting 2-gram and 3-gram RE for *corpora* from 25 up to 682 Mword. The n -gram cache efficiency increases due to the n -gram repetitions and by a novel warming-up strategy based on the combined glue calculations of different n -gram sizes. This will be further evaluated for larger *corpora* and n -gram sizes. Even using the n -gram cache, ParLocalMaxs still has a significant communication overhead. An alternative method is under way based on a collection of machines which independently and in parallel extract RE candidates from separate *corpus* partitions, followed by their global combination. This completely eliminates the phase 2 communication overhead, leading to improved scalability. However, this method is not able to ensure the same precision and recall as LocalMaxs that depend on the *corpus* and partition sizes.

The proposals in this paper have wider applicability beyond LocalMaxs to a general class of n -gram extraction methods and applications, involving other cohesion metrics such as *Dice*, *MI*, φ^2 , etc. [35].

ACKNOWLEDGMENTS

FCT MCTES, NOVA LINCS UID/CEC/04516/2013; Lunacloud www.lunacloud.com; ISEL “AzoresCloud”.

REFERENCES

- [1] Silva et al., “Using LocalMaxs Algorithm for the Extraction of Contiguous and Non-contiguous Multiword Lexical Units,” in *Proceedings of the 9th Portuguese Conf. on Artificial Intelligence: Progress in Artificial Intelligence*, ser. EPIA ’99. Springer-Verlag, 1999.
- [2] Goncalves et al., “A Parallel Algorithm for Statistical Multiword Term Extraction from Very Large Corpora,” in *Proceedings of 17th IEEE Int. Conf. on High Performance Computing and Communications (HPCC)*, Aug 2015, pp. 219–224.
- [3] Kulkarni et al., “jMWE: a Java toolkit for detecting multi-word expressions,” in *Proceedings of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World*, ser. MWE ’11. ACL, 2011.
- [4] R. M. K. Sinha, “Stepwise mining of multi-word expressions in Hindi,” in *Proceedings of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World*, ser. MWE ’11. ACL, 2011.
- [5] Wehrli et al., “Sentence Analysis and Collocation Identification,” in *Proceedings of the Workshop on Multiword Expressions: from Theory to Applications*, ser. MWE ’10. COLING, 2010.
- [6] Park et al., “Automatic glossary extraction: beyond terminology identification,” in *Proceedings of the 19th Int. Conf. on Computational linguistics*, ser. COLING ’02. ACM, 2002.
- [7] Velardi et al., “Mining the Web to Create Specialized Glossaries,” *Intelligent Systems, IEEE*, vol. 23, no. 5, sep 2008.
- [8] Wernter et al., “Finding new terminology in very large corpora,” in *Proceedings of the 3rd int. Conf. on Knowledge Capture*, ser. K-CAP ’05. ACM, 2005.
- [9] D. Pearce, “A Comparative Evaluation of Collocation Extraction Techniques,” in *Proceeding 3rd Int. Conf. on Language Resources and Evaluation*, ser. LREC ’02. European Language Resources Association, 2002.
- [10] Ventura et al., “Mining concepts from texts,” in *Procedia Computer Science*. Elsevier, 2012.
- [11] Agrawal et al., “Mining Association Rules Between Sets of Items in Large Databases,” in *Proceedings of the 1993 ACM SIGMOD Int. Conf. on Management of Data*, vol. 22, no. 2, jun 1993.
- [12] Itkär et al., “Distributed Sequential Pattern Mining: A Survey and Future Scope,” *Int. Journal of Computer Applications*, vol. 94, no. 18, may 2014.
- [13] Wang et al., “An Overview of Microsoft Web N-gram Corpus and Applications,” in *Proceedings of the NAACL HLT 2010 Demonstration Session*, ser. HLT-DEMO ’10. ACL, 2010.
- [14] Banerjee et al., “The Design, Implementation, and Use of the Ngram Statistics Package,” in *Proceedings of the 4th Int. Conf. on Computational Linguistics and Intelligent Text Processing*, ser. CICLing’03. Springer-Verlag, 2003.
- [15] Lin et al., “New Tools for Web-Scale N-grams,” in *Proceedings of the Int. Conf. on Language Resources and Evaluation*, 2010.
- [16] Li et al., “Pfp: Parallel Fp-growth for Query Recommendation,” in *Proceedings of the 2008 ACM Conference on Recommender Systems*, ser. RecSys ’08. ACM, 2008.
- [17] Pereira et al., “A parallel multikey quicksort algorithm for mining multiword units,” in *Proceedings of LREC-04 Workshop on Methodologies & Evaluation of Multiword Units in Real-world Applications*, 2004.
- [18] Dean et al., “MapReduce: Simplified Data Processing on Large Clusters,” in *Proceedings of the 6th Conf. on Symposium on Operating Systems Design & Implementation*. USENIX Association, 2004.
- [19] J. Lin and C. Dyer, *Data-Intensive Text Processing with MapReduce*. Morgan and Claypool Publishers, 2010.
- [20] Berberich et al., “Computing N-gram Statistics in MapReduce,” in *Proceedings of the 16th Int. Conf. on Extending Database Technology*, ser. EDBT ’13. ACM, 2013.
- [21] Goncalves et al., “Data Analytics in the Cloud with Flexible MapReduce Workflows,” in *Proceedings of 4th IEEE Int. Conf. on Cloud Computing Technology and Science*, ser. CloudCom ’12. IEEE Computer Society, December 2012, pp. 427–434.
- [22] Lee et al., “Parallel Data Processing with MapReduce: A Survey,” *SIGMOD Rec.*, vol. 40, no. 4, jan 2012.
- [23] Assuncao et al., “Autonomic Activities in the Execution of Scientific Workflows: Evaluation of the AWARD Framework,” in *Proceedings of 9th Int. Conf. on Autonomic Trusted Computing*, 2012.
- [24] Memcached. (‘16, May). [Online]. Available: <https://memcached.org/>
- [25] Cassandra. (‘16, May). [Online]. Available: <http://cassandra.apache.org/>
- [26] R. Kuhn, “Speech Recognition and the Frequency of Recently Used Words: A Modified Markov Model for Natural Language,” in *Proceedings of the 12th Conf. on Computational Linguistics*, ser. COLING ’88, vol. 1. Stroudsburg, PA, USA: ACM, 1988, pp. 348–350.
- [27] G. K. Zipf, “The Psychobiology of Language: An Introduction to Dynamic Philology,” in *Cambridge M.I.T. Press*, Mass, 1935.
- [28] Baeza-Yates et al., “The Impact of Caching on Search Engines,” in *Proceedings of the 30th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, ser. SIGIR ’07. New York, USA: ACM, 2007, pp. 183–190.
- [29] Breslau et al., “Web caching and Zipf-like distributions: evidence and implications,” in *Proceedings of the 8th Annual Joint Conf. of the IEEE Computer and Communications Societies*, ser. INFOCOM ’99, vol. 1, Mar 1999, pp. 126–134 vol.1.
- [30] Balkir et al., “A distributed look-up architecture for text mining applications using mapreduce,” in *Proceedings of Int. Conf. for High Performance Computing, Networking, Storage and Analysis (SC)*, Nov 2011, pp. 1–11.
- [31] ParLocalMaxs. (‘16, May). [Online]. Available: <http://cjsg.dynip.sapo.pt/>
- [32] Wikimedia. (‘16, May). [Online]. Available: <http://dumps.wikimedia.org>
- [33] LunaCloud. (‘15, May). [Online]. Available: www.lunacloud.com/
- [34] Silva et al., “A Theoretical Model for n-gram Distribution in Large Natural Language Corpora,” in *Submitted*, 2016.
- [35] Silva et. al, “A local Maxima Method and a Fair Dispersion Normalization for Extracting Multiword Units,” in *In Proceedings of the 6th Meeting on the Mathematics of Language*, Quinta Da Torre - Monte Da Caparica, 1999, pp. 369–381.

DOT-K: A Distributed Online Top-K Elements Algorithm using Extreme Value Statistics

Nicholas Carey

Department of Computer Science
Johns Hopkins University
Baltimore, Maryland 21218
ncarey4@jhu.edu

Tamás Budavári

Department of Applied Math and Statistics
Johns Hopkins University
Baltimore, Maryland 21218
budavari@jhu.edu

Yanif Ahmad

Department of Computer Science
Johns Hopkins University
Baltimore, Maryland 21218
yanif@jhu.edu

Alexander Szalay

Department of Physics and Astronomy
Johns Hopkins University
Baltimore, Maryland 21218
szalay@jhu.edu

Abstract—Extremely large (peta-scale) data collections are generally partitioned into millions of containers (disks/volumes/files) which are essentially unmovable due to their aggregate size. They are stored over a large distributed cloud of machines, with computing co-located with the data. Given this data layout, even simple tasks are difficult to perform and naive algorithms can easily become quite expensive. We present a one pass, communications-efficient technique useful for both estimating upper order quantiles and selecting the largest k elements across a highly distributed dataset or stream. Our novel approach draws its foundations from Extreme Value Statistics (EVS) to reason about the statistical relationships between the tail distributions of dataset partitions. The tail of each partition is fitted by the Generalized Pareto Distribution, which captures threshold exceedances. The obtained parameters are communicated to a central coordinator and used to estimate quantiles, or solve for a threshold above which there are approximately k elements. We discuss the computational and bandwidth costs of the algorithm, and demonstrate the accuracy of the method on both a variety of synthetic datasets and a PageRank dataset.

I. INTRODUCTION

We present DOT-K, a new communication-efficient algorithm for solving a two common tasks in distributed data management: (a) estimating the k th largest element of a data set split over many partitions and (b) subsequently retrieving the largest k elements. More formally, given any real-valued $x_r = f(A_r)$ function defined on (one or more) attributes A_r of row r in the dataset D_i of partition i , our objective is to accurately estimate the k th largest value $f_{(k)}$ over all $r \in D$ the entire collection of partitions such that the size of the result set is equal to a given k ,

$$|f(A_r) : f(A_r) > f_{(k)}| = k. \quad (1)$$

Top-k and quantile queries are relevant to a variety of applications, including outlier detection and general data exploration. Top-k queries are commonly executed in all sorts of database environments, many of which are sensitive to different bottlenecks. The scope of top-k query processing

is reflected in the amount of solutions that cater to specific problem environment requirements [1]. The Threshold Algorithm (TA), developed by [2][3][4], is a well understood and efficient method of finding the top- k values of a monotonic aggregation function over row attributes; in the TA setting, the top- k query result will be the largest valued outputs from a scoring function that takes several row object attributes as input. The TA-style top- k query has received much attention and has successfully been adapted to a distributed environment by algorithms such as KLEE and TPUT [5][6]. Distributed TA-style algorithms are appropriate for a column-partitioned database where a single object's attributes may be spread across multiple partitions, such as a heterogeneous multimedia database where a row object's video data is stored at a different node than its associated image or text data. In contrast, we examine the top- k query in a highly distributed, row-partitioned database system; DOT-K assumes that the data objects over which a top- k query is made are represented in full at their respective partitions. Applications where a single, homogeneous dataset must be split between many parallel nodes are an appropriate use case for DOT-K. An enterprise network tap, in which logs of company net activity are collected and stored at many distributed locations, would be an ideal setting. Another example is a top- k query over a scientific dataset row-partitioned across a large cluster of machines, at a scale where minimizing communications costs between machines becomes a priority.

We focus on consuming minimal network resources when solving top-k queries in a highly distributed environment. Let P be the amount of a dataset containers, or partitions. A naive distributed top-k query would ask each machine to forward their local partition's largest k values and subsequently sort the P lists to find the global top-k result. The naive solution is completely acceptable when P or k is small; however, at a large, datacenter scale the network cost of communicating the P local top-k lists and the computation cost of sorting those

lists becomes prohibitive. In DOT-K, we address these costs by summarizing the tail of each dataset partition using an Extreme Value Statistics distribution. Instead of communicating P sets of local top-k lists, each partition forwards the EVS distribution parameters that describe the local top-k values. The central query coordinator then relates the tail distributions of each partition and calculates an estimate for the global dataset's k th order statistic. Finally, the estimate for the k th largest value is communicated to all dataset partitions and all values greater than the k th order statistic estimate are sent back to the query coordinator.

DOT-K is also communications-efficient at estimating many high-order quantiles in one query. While not applicable to answering medians or any quantile query across the board as in Greenwald-style quantile estimation algorithms, such as those given by Zhang and Wang[7][8], depending on the DOT-K implementation choice of local EVS distribution parameter estimators, the DOT-K method can achieve similar memory and compute costs as sketch-based quantile estimation algorithms as detailed in Section 3.2.

In Section 2 we examine the necessary Extreme Value Statistics pertinent to DOT-K. In Section 3, we present the DOT-K algorithm and our Threshold Equation. Section 4 outlines our experimental communications and accuracy evaluations of DOT-K, and we conclude with possible improvements in Section 5.

II. EXTREME VALUES

Extreme Value Statistics (EVS) is concerned with characterizing the tail distributions, or extreme values, of random variables. EVS has traditionally been used to model extreme environmental phenomena, such as sea levels or wind speeds, as well as weakest-links in reliability modeling[9]. An area of interest in EVS is modeling a distribution's random variables which exceed a threshold. As a top-k query is concerned with the k data elements which exceed the k th order statistic threshold, points-over-threshold EVS has a natural application to top-k query processing. An EVS theorem developed by Pickands, Balkema and de Haan states that the distribution of threshold exceedences of a sequence of independent and identically-distributed random variables with a common continuous underlying distribution function is approximated by the Generalized Pareto Distribution, and that the approximation converges as the tail threshold rises[10][11][9]. Therefore, for a large class of common data distributions, the k largest values may be well approximated by a Generalized Pareto Distribution if the k th order statistic is an appropriately high enough threshold. Picking a threshold that defines the tail can be a delicate choice; due to a bias-variance trade-off, a lower threshold results in a worse GPD approximation while a higher threshold limits the amount of available threshold exceedences leading to greater parameter estimation uncertainty[12]. However this property bodes well for larger datasets, and benefits DOT-K at extreme scales. The true dataset tail grows with the dataset size, which affords a more accurate tail summarization.

In a distributed query setting, we summarize the extreme values, or the tail, of each partition in order to attain minimal communication costs while still relate data across distributed nodes. Rather than sending samples of tail data, the partitions describe their local largest k values by fitting a GPD and communicating the GPD parameters to the central query coordinator.

The probability distribution function of the Generalized Pareto Distribution

$$p(x|\xi, \sigma, \mu) = \frac{1}{\sigma} \left[1 + \xi \cdot \left(\frac{x-\mu}{\sigma} \right) \right]^{-\frac{1}{\xi}} \quad (2)$$

is defined by three parameters: the threshold μ , the shape ξ , and the scale σ . There are several methods for estimating the GPD parameters that best model a given set of threshold exceedences. A survey of these methods along with their respective strengths and weaknesses is not within the scope of this paper. Many authors have approached the subject of GPD parameter estimation including Pickands [10], Hosking and Wallis [13], Castillo and Hadi [14], Zhang [15], and Husler [16]. In our experimental implementation of DOT-K, we use a Maximum Likelihood Estimator based approach to fit a GPD to each dataset partition's extreme values. The accuracy of a DOT-K query result is entirely based upon the accuracy of the GPD summarizations of each dataset partition tail. We assume an extremely large distributed dataset with at least thousands of extreme values within each dataset partition. When fitting 500 or more data elements an MLE-based method is a proven GPD parameter estimation technique[13].

For a given GPD, one can calculate the threshold level x_m that is exceeded on average once every m observations in the underlying dataset [9]. The threshold level is useful when estimating dataset quantiles. By relating m to the dataset size, we can solve for order statistics. Coles[9] gives the formula to solve for the m-observation return level

$$\zeta_\mu \left[1 + \xi \cdot \left(\frac{x_m - \mu}{\sigma} \right) \right]^{-\frac{1}{\xi}} = \frac{1}{m} \quad (3)$$

where ζ_μ is the probability of an observation exceeding the GPD threshold parameter μ , which is estimated by

$$\hat{\zeta}_\mu = \frac{N_\mu}{N} \quad (4)$$

where N is the dataset size and N_μ is the number of elements greater than the GPD threshold μ . Since DOT-K fits each partition's local largest k values to a GPD, then the value of ζ_μ is calculated with $N_u = k$ and $N = n_i$ where n_i is the local partition set size. We expand on these ideas and generalize them to partitioned datasets.

III. DOT-K ALGORITHM

Our goal is to first estimate the k th largest element of the entire data collection D and subsequently retrieve all elements greater than the estimate. This is achieved in the following steps:

- 1) Model the tail distribution of the quantity of interest in each partition and communicate the results to a central coordinator
- 2) Using the GPD fits from all partitions, solve for a global threshold that is expected to yield the largest k elements
- 3) Query the partitions with a global threshold and return all elements that exceed the limit

The rest of this section is concerned with obtaining the global threshold, the local strategies and analysing the communication requirements of the algorithm.

A. Global Threshold

For each partition i , the coordinator knows not only the GPD parameters but also the number of elements $\{n_i\}$ with which in hand, it can estimate the tail distribution of the entire collection, $D = \cup D_i$. For that we obtain the m -observation exceedance equation

$$\sum_{i=1}^p n_i \zeta_{\mu_i} \left[1 + \xi_i \cdot \left(\frac{x_m - \mu_i}{\sigma_i} \right) \right]^{-\frac{1}{\xi_i}} = k \quad (5)$$

which simply states that the total estimated number of exceedances above the global threshold x_m is equal to the requested number of elements, k , cf. Eq. (3). This one-dimensional equation can be solved numerically using standard off-the-shelf procedures. We note that the interval on which x_m is defined is determined from individual fits that each limit the possible values. Depending on the shape parameters the constraints can bound from both sides. In practice, this is not a limitation but a way to better initialize the numerical solvers.

It is worth pointing out that if any one of the nodes fail to deliver their estimates due to a temporary outage, the coordinator can obtain a global threshold by simply ignoring the missing partitions. This estimate will be more generous than the true solution would have been in the sense that using the obtained threshold on the entire collection would simply return more than the expected number of elements. If the failed partition comes back online for the second pass when the actual selection is performed, the result of the query is still going to be correct at the expense of a few more communicated data elements.

B. Quantile Estimation

DOT-K is easily adapted to high order quantile estimation. Once the local GPD parameters are collected at the central coordinator, Eq.(5) may be used to estimate order statistics greater than the original k th order statistic by varying k in the equation, at no additional communications cost for queries concerned with the state of the dataset at the time of local GPD parameter collection. Further queries on a stream with an updated underlying dataset require a refresh of the local GPD parameters and a new round of communications.

Quantile DOT-K, if implemented with the more inaccurate but computationally light Method of Moments[17] GPD parameter estimation algorithm, need only track local dataset mean, variance, and size at each distributed partition or stream. While local partition compute and memory cost associated

with maintaining these basic statistics would be significantly less than a summary data structure based quantile algorithm, DOT-K query error would undoubtedly increase from the worse Method of Moments technique[17]. Future work includes further experiments with different GPD parameter estimation techniques in order to optimize DOT-K's local compute and memory costs while maintaining query accuracy.

C. Local Strategies

Like several other distributed top-k algorithms, at each node DOT-K assumes either sorted access to the local partition data[1]. If sorted access is unavailable, a linear scan is performed to find each partition's local top-k values.

In an online distributed stream setting, DOT-K requires summary statistics maintained over the local streams in order to keep up-to-date GPD parameters describing the local tail. For the more accurate but computationally expensive parameter estimators, such as Maximum Likelihood, a length k priority queue data structure maintaining each local stream's top data elements is needed in order to obtain accurate GPD parameters at the time of a top-k query.

However, there may be ways to estimate the GPD fit in each partition using approximate sub-linear algorithms. Wu and Jermaine have discussed an approximate Bayesian method to predict the extreme values[18]. The method is based on the observation that the expected value of the k -th ranked element out of N samples is the (k) -th value out of n sub-samples, where $k/N = k/n$. One can generalize this technique to compute the GPD fit from a small subset of the approximate top ranked (e.g. $k = 10, 50, 100$) elements. One can also include a few extra elements for redundancy and validation of the fit. This can provide an additional speedup of the otherwise liner scan of the partition data. We need to explore further that given a sub-sampling rate what subset of the ranks is optimal for the estimating the GPD parameters.

IV. EXPERIMENTS AND DISCUSSION

We run and discuss two separate sets of experiments in order to evaluate the performance of DOT-K. Our first experiment investigates the communication costs and overhead incurred by DOT-K in a distributed computing environment; the second experiment gauges the quality and accuracy of DOT-K queries at scale over a variety of datasets in a pseudo-distributed environment.

A. Communication Overhead

DOT-K exhibits minimal communication costs. There are four series of messages between the query coordinator and the dataset partitions. DOT-K starts with a message from the coordinator to the partitions containing the query details, namely, the value for k . After the P partitions calculate the local top-k values and the GPD fits, each partition communicates the three GPD parameters and local partition size to the coordinator. Next, the coordinator forwards the k th order statistic estimate to the partitions, and finally the partitions send the estimate exceedances to the coordinator. Therefore,

there are a total of $4P$ messages with a total of $6P + \hat{k}$ real values transmitted, where \hat{k} is the count of estimated k th order statistic exceedances.

When used to estimate high-order quantiles, DOT-K requires even less communications between nodes. The query starts with a request from the coordinator to the distributed partitions, and communications are finished when the partitions forward the locally fitted GPD parameters back to the coordinator. With a total of $2P$ messages containing a few real values each, the coordinator may estimate any order statistic greater than the k th order statistic.

1) Message Count Experiment Setup: The goal of this experiment is to empirically evaluate our predicted communication costs of DOT-K, for both computing a quantile and a top-k query over a partitioned dataset. We implement DOT-K on Apache Storm[19], an open source distributed stream processing engine currently in use by many industry members including Twitter[20]. Storm treats a distributed program as a directed graph, with computation happening at graph nodes and data transfer along graph edges. DOT-K is ideally mapped to the Storm distributed computing model: there is a node for each distributed dataset partition, or incoming stream, and there is a central coordinator node that receives the partition summaries, then estimates the k th order statistic and broadcasts that threshold back to the partition nodes. Each Storm node logs whenever it emits or receives messages to and from other nodes; from these logs we recreate all node-to-node messages, and quantify the communications costs incurred by DOT-K.

We deploy eighty Storm workers, or computation nodes, spread over twenty Amazon AWS EC2 'm1.medium' instances. We simulate a streaming environment by randomly scattering the dataset among the nodes and constructing a Storm 'spout' to emit the data values one by one. Each node runs a Storm computation 'bolt' that maintains a priority queue of the largest-valued data elements emitted from the stream and estimates local GPD parameters with data held in the priority queue. Note that even if two nodes reside on the same instance, they still communicate using the Storm message service and the communication will be logged and counted in our evaluation. The experiment records the total number of individual messages sent between computation nodes during the course of both DOT-K quantile and top-k queries executed at scale varying between ten and eighty nodes. Figure 1 shows the global message count associated with executing DOT-K at a given parallel scale.

2) Message Count Discussion: Shown in Figure 1, it is clear that total message count per query grows linearly with parallelization, or partition count. In the previous section, we calculated that there would be one round of messages, or $2P$ necessary messages, in order to estimate high order quantiles and two rounds of communication, or $4P$ necessary messages, in order to obtain a top-k query result. However, the experiment shows a slightly higher message count than what was estimated; upon inspection of message contents, we found the extra few communications to be overhead

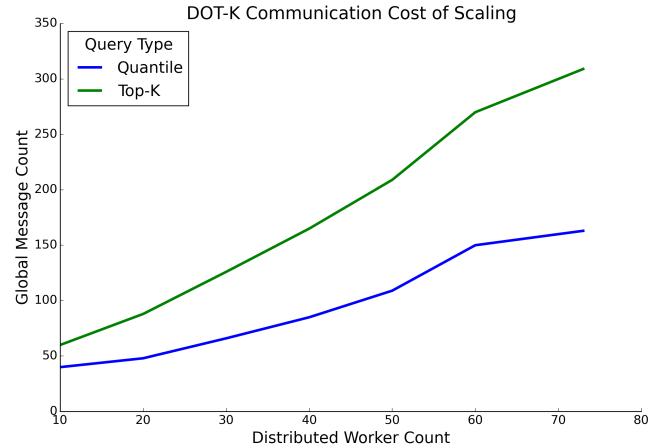


Fig. 1. Machine-to-machine communications sent during DOT-K query executions

incurred by Storm's fault tolerance and reliability features. Had we disabled the guaranteed message passing features or not counted them in the total message count, the results would match our predictions more exactly. Regardless, we show DOT-K communication cost trends to be minimal for distributed quantile and top-k queries.

Note that the size of the contents of each message is less than a dozen real-valued numbers, with the exception of the final batch of messages during a top-k query in which each dataset partition sends the data elements greater than the k th order statistic back to the coordinator to form the top-k result; these messages contain, on average, k/P data elements.

B. Query Accuracy Evaluation

Our experimental goal is to discover any error trends and determine the practicality of scaling DOT-K on a variety of datasets. While the Pickands-Balkema-de Haan theorem states that the GPD is a good approximation of threshold exceedances, it is important to show the accuracy of this summarization method in practice. We implemented a pseudo-distributed version of DOT-K to evaluate the quality of top-k query results.

Our experiments show the relative error of DOT-K as we increase the number of dataset partitions. The number of elements of each partition is constant; as we increase partition count, the global dataset size grows as well. If we had kept global dataset size constant while increasing partition count, GPD approximation of partition tails would become worse as partition size decreased and we would be unable to show how partition count alone affects DOT-K accuracy.

We demonstrate DOT-K on datasets with several different partitioning strategies. While some databases are partitioned randomly, many applications partition datasets on some criterion and it is restricting to assume every dataset will have identically and independently distributed data across partitions.

The relative error metric we use is described in equation 6. \hat{k} is the top-k query result from DOT-K, and k is the set of dataset values that exceed the estimated k th order statistic.

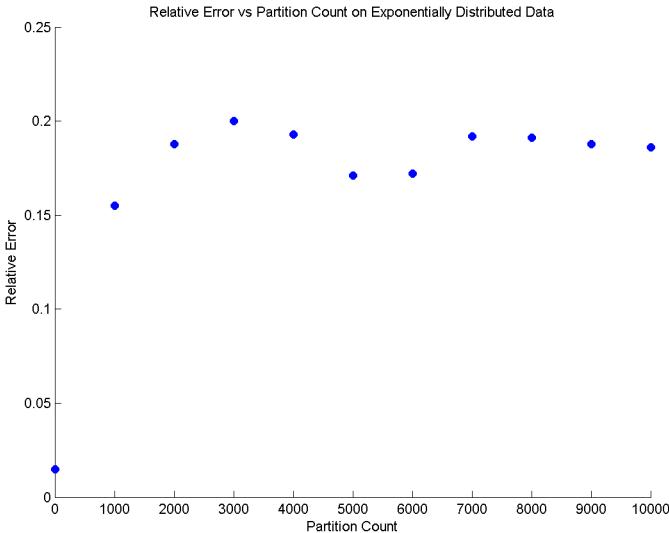


Fig. 2. Relative error at large scale on a randomly partitioned exponentially distributed synthetic dataset.

$$\delta k = \left| \frac{k - \hat{k}}{k} \right| \quad (6)$$

1) *Synthetic Datasets*: We evaluate DOT-K on synthetic datasets consisting of real-valued random variables drawn from a randomly partitioned exponentially distributed synthetic dataset. DOT-K relative error results were obtained from the average of twenty executions on datasets generated with the same distribution parameters but different random seeds. For these experiments, k is 1,000, partition size is fixed to 300,000 data elements, and the partition count ranges from 1,000 to 10,000.

Figure 2 shows DOT-K relative accuracy at high scale with partition counts from 1,000 to 10,000.

2) *Berkeley PageRank Dataset*: We evaluated DOT-K on the Berkeley Amplab Big Data Benchmark PageRank dataset[21] in order to test our method on a somewhat common application: finding the k highest ranked websites. The Berkeley PageRank dataset is based on the distributed system analysis benchmark work done by Pavlo et al[22] and consists of approximately 90 million URLs associated with their respective rank. For these experiments, k is set to 1,000, the amount of partitions ranges from 100 to 1,000, and the partition size is set to 90,000 URL-PageRank pairs. We run accuracy experiments on two versions of this dataset: randomly partitioned and biased partitioned. For the randomly partitioned PageRank dataset we shuffled the data elements randomly over each partition. The biased partitioning scheme consists of simply ordering the data elements as we obtained them from the Berkeley Benchmark download; upon inspection, it is clear that chunking the original dataset, in order, produces partitions with very different tails. This undefined, but clearly biased, partitioning scheme is useful in simulating a

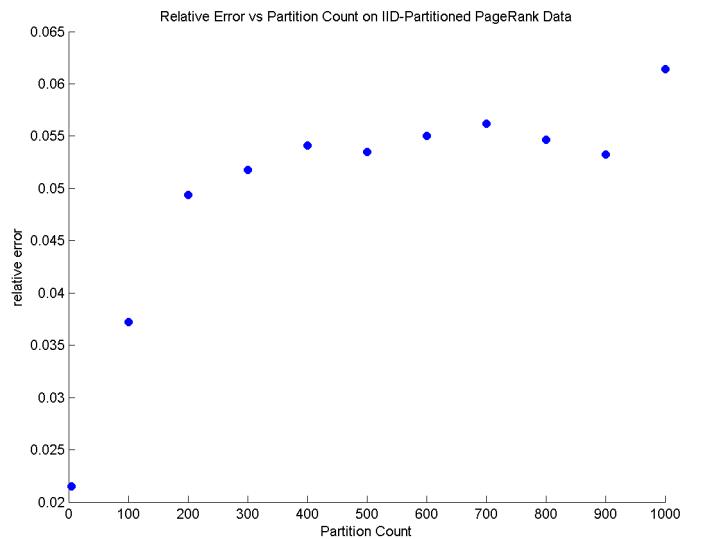


Fig. 3. Relative DOT-K Error on a PageRank dataset with a random partitioning scheme.

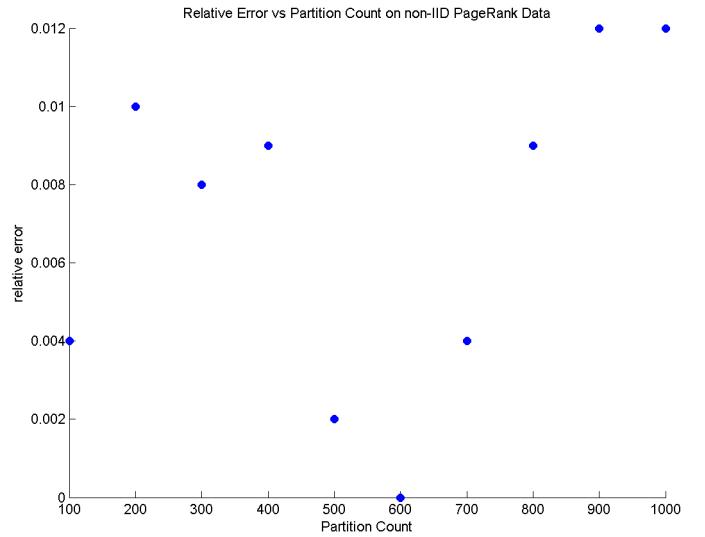


Fig. 4. Relative DOT-K Error on a PageRank dataset in which extreme values are concentrated in few partitions.

real world practical query, as data is not always independently and identically distributed.

We ran experiments on two partitioning schemes of the PageRank dataset. Figure 3 shows average relative error of running DOT-K over twenty different random partitionings of the PageRank dataset. In this experiment, the data across partitions is independent and identically distributed.

Figure 4 shows the relative error of DOT-K executed on the PageRank dataset in the same state in which we obtained it; that is, partition data is differently distributed between partitions and there is a clear bias of extreme values concentrated in a few partitions. This experiment shows the result of one execution of DOT-K over a single PageRank dataset with a biased partitioning scheme.

3) *Accuracy Discussion:* DOT-K exhibits two trends between error and partition count. The error on the PageRank and exponentially distributed datasets either tails off to a limit or is too low to show any certain trends. However, error generally rises with partition count, likely due to the rising cumulative GPD fit uncertainty. Each GPD fit to local partition data has at least some small amount of error in its description of the real data; as partition count rises and more GPD parameter sets are involved in the DOT-K estimate of the global k th order statistic, there is both a greater cumulative amount of GPD fit error and a greater k th order statistic estimate error.

Surprisingly, DOT-K showed less overall error on the biased partitioned PageRank dataset than the randomly partitioned PageRank datasets. DOT-K executed on the biased partitioned PageRank data resulted in error too small to show any real trend. The nature of the biased partitioning may have aided DOT-K; in the unaltered PageRank dataset, URLs with outlier ranks were concentrated to few partitions. The DOT-K global threshold equation estimates that the many partitions with lesser tail distributions would not contribute any elements to the top-k query result. Partitions with maximum data values less than the largest threshold GPD parameter will not be represented in the global top-k query result, and may be pruned away. Therefore, DOT-K only needed to relate few partitions, meaning less cumulative GPD fit error represented in the k th order statistic estimate. This property benefits applications in which the data is not randomly partitioned; DOT-K error scales with the amount of partitions that are estimated to contain elements belonging to the query result.

V. CONCLUSION

DOT-K shows promise as a communications efficient distributed top-k elements algorithm. By summarizing the tail distributions of dataset partitions, DOT-K pushes computation out to distributed nodes in order to conserve bandwidth usage; only the query, GPD parameters, k th order statistic estimate, and global top-k elements are communicated. Due to Pickands-Balkema-de Haan Theorem, the Generalized Pareto Extreme Value Distribution becomes more effective at describing dataset tails as the dataset size increases. DOT-K will become more useful as datasets grow larger and the desired k largest values are numerous enough to otherwise incur substantial bandwidth and sorting cost.

There are several improvements to be made to the DOT-K algorithm. Firstly, the accuracy of the entire method rests on the quality of the GPD parameter estimation at each of the dataset partitions. Numerous publications examine the quality of GPD parameter estimators, and several new methods given by Zhang[15] and Husler, Li, and Raschke[16] combine the favorable qualities of different estimators. Performance would benefit if these methods provide a better or faster GPD fit. Secondly, we would like to experiment with different methods of reasoning about the relative tail distributions of the dataset partitions. Pruning away partitions estimated to not contribute to the top-k result is a start, but further inferences on the relationships between partition tail data may be possible.

Finally, we would like to experiment with sublinear sampling techniques when fitting GPDs to partition tails. Currently DOT-K requires each partition to perform a linear scan to find the local top-k values; if a GPD could describe a sample of a partition tail without a significant loss of overall accuracy, the overall performance of DOT-K would benefit.

REFERENCES

- [1] I. F. Ilyas, G. Beskales, and M. A. Soliman, "A survey of top-k query processing techniques in relational database systems," *ACM Comput. Surv.*, vol. 40, no. 4, pp. 11:1–11:58, Oct. 2008. [Online]. Available: <http://doi.acm.org/10.1145/1391729.1391730>
- [2] S. Nepal and M. V. Ramakrishna, "Query processing issues in image(multimedia) databases," in *Proceedings of the 15th International Conference on Data Engineering*, ser. ICDE '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 22–29. [Online]. Available: <http://dl.acm.org/citation.cfm?id=846218.847271>
- [3] U. Guntzer, W.-T. Balke, and W. Kiessling, "Optimizing multi-feature queries for image databases," in *Proceedings of the 26th International Conference on Very Large Data Bases*, ser. VLDB '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 419–428. [Online]. Available: <http://dl.acm.org/citation.cfm?id=645926.671875>
- [4] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," in *Proceedings of the Twentieth ACM SIGMOD-SIGART-SIGART Symposium on Principles of Database Systems*, ser. PODS '01. New York, NY, USA: ACM, 2001, pp. 102–113. [Online]. Available: <http://doi.acm.org/10.1145/375551.375567>
- [5] S. Michel, P. Trianfyllou, and G. Weikum, "Klee: A framework for distributed top-k query algorithms," in *Proceedings of the 31st International Conference on Very Large Data Bases*, ser. VLDB '05. VLDB Endowment, 2005, pp. 637–648. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1083592.1083667>
- [6] P. Cao and Z. Wang, "Efficient top-k query calculation in distributed networks," in *Proceedings of the Twenty-third Annual ACM Symposium on Principles of Distributed Computing*, ser. PODC '04. New York, NY, USA: ACM, 2004, pp. 206–215. [Online]. Available: <http://doi.acm.org/10.1145/1011767.1011798>
- [7] M. Greenwald and S. Khanna, "Space-efficient online computation of quantile summaries," *SIGMOD Rec.*, vol. 30, no. 2, pp. 58–66, May 2001. [Online]. Available: <http://doi.acm.org/10.1145/376284.375670>
- [8] Q. Zhang and W. Wang, "A fast algorithm for approximate quantiles in high speed data streams," in *Scientific and Statistical Database Management, 2007. SSBDM '07. 19th International Conference on*, July 2007, pp. 29–29.
- [9] S. Coles, *An Introduction to Statistical Modeling of Extreme Values*, 1st ed., ser. Springer Series in Statistics. Springer-Verlag London, 2001.
- [10] J. P. III, "Statistical inference using extreme order statistics," *Ann. Statist.*, vol. 3, no. 1, pp. 119–131, 01 1975. [Online]. Available: <http://dx.doi.org/10.1214/aos/1176343003>
- [11] A. A. Balkema and L. de Haan, "Residual life time at great age," *Ann. Probab.*, vol. 2, no. 5, pp. 792–804, 10 1974. [Online]. Available: <http://dx.doi.org/10.1214/aop/1176996548>
- [12] C. Scarrott and A. MacDonald, "A review of extreme value threshold estimation and uncertainty quantification," *REVSTAT - Statistical Journal*, vol. 10, no. 1, pp. 33–60, 2012. [Online]. Available: <https://www.ine.pt/revstat/pdf/rs120102.pdf>
- [13] J. R. M. Hosking and J. F. Wallis, "Parameter and quantile estimation for the generalized pareto distribution," *Technometrics*, vol. 29, no. 3, pp. 339–349, Sep. 1987. [Online]. Available: <http://dx.doi.org/10.2307/1269343>
- [14] E. Castillo and A. S. Hadi, "Fitting the generalized pareto distribution to data," *Journal of the American Statistical Association*, vol. 92, no. 440, pp. 1609–1620, 1997.
- [15] J. Zhang, "Likelihood moment estimation for the generalized pareto distribution," *Australian and New Zealand Journal of Statistics*, vol. 49, no. 1, pp. 69–77, 2007. [Online]. Available: <http://dx.doi.org/10.1111/j.1467-842X.2006.00464.x>
- [16] J. Husler, D. Li, and M. Raschke, "Estimation for the generalized pareto distribution using maximum likelihood and goodness of fit," *Communications in Statistics - Theory and Methods*, vol. 40, no. 14, pp. 2500–2510, 2011.

- [17] P. de Zea Bermudez and S. Kotz, "Parameter estimation of the generalized pareto distribution," *Journal of Statistical Planning and Inference*, vol. 140, no. 6, pp. 1353 – 1373, 2010. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0378375809002766>
- [18] M. Wu and C. Jermaine, "Guessing the extreme values in a data set: a bayesian method and its applications," *The VLDB Journal*, vol. 18, no. 2, pp. 571–597, 2009. [Online]. Available: <http://dx.doi.org/10.1007/s00778-009-0133-6>
- [19] N. Marz, *Storm: Distributed and Fault-Tolerant Realtime Computation*. storm.apache.org, 2014.
- [20] A. Toshniwal, S. Taneja, A. Shukla, K. Ramasamy, J. M. Patel, S. Kulkarni, J. Jackson, K. Gade, M. Fu, J. Donham, N. Bhagat, S. Mittal, and D. Ryaboy, "Storm@twitter," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '14. New York, NY, USA: ACM, 2014, pp. 147–156. [Online]. Available: <http://doi.acm.org/10.1145/2588555.2595641>
- [21] P. Wendell, *Berkeley AmpLab Big Data Benchmark*. <https://AMPLAB.cs.berkeley.edu/benchmark/>, 2014.
- [22] A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden, and M. Stonebraker, "A comparison of approaches to large-scale data analysis," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '09. New York, NY, USA: ACM, 2009, pp. 165–178. [Online]. Available: <http://doi.acm.org/10.1145/1559845.1559865>

Budget Distribution Strategies for Scientific Workflow Scheduling in Commercial Clouds

Vahid Arabnejad, Kris Bubendorfer and Bryan Ng

School of Engineering and Computer Science, Victoria University of Wellington, New Zealand

Email: {vahid, kris and bryan.Ng}@ecs.vuw.ac.nz

Abstract—Scientific research is increasingly reliant on big compute and big data, the fusion of which is known as data intensive science. Large scale scientific analyses are typically represented as workflows which are the typical model for characterizing e-science experiments in distributed systems. Workflows with a large number of tasks are distributed in parallel across computing resources to speed up analyses. The provision of compute capabilities is undergoing a rapid migration from dedicated infrastructure to the cloud. This migration is fuelled by dynamic infrastructure scalability with changes in demand. Cloud instances incur different costs and execution time with different configurations. A key concern for workflow scheduling is to make an appropriate trade-off between these two factors. In this paper, we introduce the Budget Distribution with Trickling (BDT) algorithm that presents new notions for distributing budget based on the dependency structure inherent in workflows. In addition we propose several new strategies for sharing or distributing the budget, and propose trickling to redistribute unspent budget down to other levels. Our results show that biasing the budget distribution to the earlier computation within a workflow will generally produce a lower makespan within budget.

I. INTRODUCTION

Basic research, and consequently scientific discovery, are in the midst of a disruptive transformation. Research is increasingly reliant on big compute and big data, the fusion of which is known as data intensive science. The execution of analytics on big data over large compute, is most commonly modelled and managed in workflows [1]. Scientific workflows vary in size from a couple of tasks to thousands or million of tasks. Workflows with a large number of tasks are distributed in parallel across computing resources. The provision of compute capabilities is undergoing a rapid migration from dedicated infrastructure to the cloud [2]. Cloud computing enables significant computational leverage to be applied to many real world problems, be they industrial, medical or scientific.

While cloud platforms provide enormous elastic computing capacity, they also pose unique multi-objective scheduling challenges with respect to cost, time and data movement. Efficiently managing pay-per-use heterogeneous cloud infrastructure for large data and compute within research budgets is a challenge that is, or soon will be, faced in almost every research domain.

The financial cost and total execution time of a workflow depends on the number and types of instances requested during resource provisioning. The cost plays a significant role in a cloud environment as users wish to minimise costs

and providers maximise profits. Most cloud providers, like Amazon, charge users for a minimum period of time - even if the instance is only used for a shorter period.

In this paper we will focus on the issue of scheduling budget constrained workflows on commercial pay-per-use clouds while trading off cost and time. One problem in scheduling budget constrained workflow is how to spend that budget for the best performance. This is essentially a budget assignment problem (BAP), and in existing workflow scheduling approaches [3], [4] is shared proportionally based on a subset of the execution characteristic(s) of the task (or cluster of data related tasks) being scheduled, such as, execution time, CPU requirements or memory requirements.

The novelty of our work is that we look at distributing budget based on the dependency structure embedded in the workflow. Essentially we transform the workflow into internally dependency free “bags of tasks” (called levels [5]) and we then distribute the workflow budget over these levels using six strategies. Three of these strategies are designed explicitly for our means of budget distribution and therefore also represent novel work. We ensure that any budget share that is unused by the level to which it is allocated is trickled down to the next level. For the remainder of this paper we will refer to our approach as Budget Distribution with Trickling (BDT).

Based on our results we suggest two hypotheses worthy of further consideration:

Hypothesis 1 *The earliest tasks in the workflow are the most critical when constructing a schedule.*

Hypothesis 2 *Assigning a higher budget to the earliest tasks in a workflow generally leads to a lower makespan.*

The rest of the paper is organised as follows: In Section II, we formalize the workflow and system models. In Section III, we present the BDT building blocks, strategies and algorithm. In Section IV, we describe the evaluation method using CloudSim, followed by results and performance evaluation. In section V we present the related work, and finally, we conclude this paper in Section VI.

II. WORKFLOW AND SYSTEM MODELS

A. Workflow Model

A Directed Acyclic Graph (DAG) is the most common representation of a workflow. A workflow is defined as a graph $G = (T, E)$ where $T = \{t_0, t_1, \dots, t_n\}$ is a set of tasks

represented by vertices and $E = \{e_{i,j} \mid t_i, t_j \in T\}$ is a set directed edges denoting data or control dependencies between tasks.

An edge $e_{i,j} \in E$ represents the precedence constraint as a directed arc between two tasks t_i and t_j where $t_i, t_j \in T$. The edge indicates that task t_j can start only after completing the execution of task t_i with all data received from t_i , and this implies that task t_i is the parent of task t_j , and task t_j is the successor or child of task t_i . Each task can have one or more parents or children. Task t_i cannot start until all parents have completed.

B. System Model

Public cloud provides instance types containing various amounts of CPU, memory, storage and network bandwidth at different prices. In this paper we use a resource model based on the Amazon Elastic Compute cloud, where instances are provisioned on demand. The pricing model is a pay as you go with minimum hourly billing. Under this pricing model, if an instance is used for one minute, a user pays for the whole hour. The costs and instance types used in this paper are given in Table I, and were accurate in March 2016.

TABLE I: Instance Types

Type	ECU	Memory(GB)	Cost(\$)
m3.medium	3	3.75	0.067
c4.large	8	3.75	0.105
c3.xlarge	14	7.5	0.21
m4.2xlarge	26	32	0.479
c4.4xlarge	62	30	0.838
c3.8xlarge	108	60	1.68

We assume that cloud vendors provide access to unlimited number of instances and the instances are heterogeneous (denoted by $P = \{p_0, p_1 \dots p_h\}$, where h is the index of the instance type). We also assume that all instances and storage services are located in the same region and also assume that the average bandwidth between the instances is essentially identical.

III. THE BUDGET-AWARE SCHEDULING ALGORITHM

In this section, we describe our budget-aware scheduling algorithm, Budget Distribution with Trickling (BDT). The algorithm is divided into four main phases (each of which relates to a following subsection: III-A- III-D):

- (A) Workflow partitioning: The workflow is partitioned into dependency free bags of tasks, called levels.
- (B) Budget Distribution: The user-defined budget is then allocated to each defined level using one of six different strategies.
- (C) Task Selection: A task is selected based on its priority in the ready list for execution.
- (D) Instance Selection: The instances are chosen to meet the available budget.

The focus of this paper is on budget distribution, the other phases are included for completeness.

A. Workflow Partitioning

We aim to maximize task parallelism by arranging tasks in levels, where within each level no tasks have dependencies on another in the same level, such that there are no dependencies between tasks. Each level can therefore be thought of as a bag of tasks (BoT) containing a set of independent tasks.

There are two main algorithms for allocating tasks to different levels, Deadline Bottom Level (DBL) [5] and Deadline Top Level (DTL) [6]. DBL and DBT categorize tasks in bottom-top direction and top-bottom direction, respectively. In this paper, we use the DBL algorithm to partition tasks into different levels.

We describe the level of task t_i as an integer representing the maximum number of edges in the paths from task t_i to the exit task (see Fig. 1). The level number (denoted by N_L) associates a task to a BoT. For the exit task, the level number is always 1, and for the other tasks, it is determined by:

$$N_L(t_i) = \max_{t_j \in \text{succ}(t_i)} \{N_L(t_j) + 1\} \quad (1)$$

where $\text{succ}(t_i)$ denotes the set of immediate successors of task t_i . All tasks are then grouped into Task Level Sets (TLS) based on their levels.

$$\text{TLS}(\ell) = \{t_i \mid N_L(t_i) = \ell\} \quad (2)$$

where ℓ is an integer denoting the level in $[1 \dots N_L(t_{entry})]$.

B. Budget Distribution

As the principle of distributing budget based on the dependency structure of a workflow (levels) is new – we need to evaluate the performance of a variety of strategies to gauge the value of the approach. We start with the most basic strategies, random and uniform, to provide a baseline comparison. We then explore more complex strategies - width, which is an analogue of prior work on proportional schemes, and then the strategies designed specifically for our BDT approach - height, area and “All in”.

The most significant differences in each strategy lie in the calculation of the sub-budget. In some strategies, we have a Budget Factor (BF) that determines a share of the budget for each level. Each sub-budget assigned to a level is termed the level budget.

- 1) Random: The budget is allocated randomly over the levels in the workflow
- 2) Uniform: Each level gets a $1/L$ share of the budget, where L is the total number of levels.
- 3) Height Proportional: Each level gets a share of the user budget proportional to its distance from the entry node. The smaller the distance the greater the share.
- 4) Width Proportional: Each level gets a share of the user budget proportional to the number of tasks within that level.

- 5) Area Proportional: Combines width and height strategies to set the budget for each level.
- 6) All in: Places the entire budget on the entry level and any remainders are trickled down to later levels. This is a refined version of Height proportional, that was formulated as an extreme test of hypotheses 1 and 2, rather than a realistic suggestion. We did not expect this strategy to work in the general case, as we anticipated a reduced success rate. However counterintuitively it returned the best overall performance.

We now present an example to show how the budget is distributed among different levels. Random needs no further explanation, so the example will only detail uniform through “All in” strategies. Figure 1 shows the structure of a sample workflow with ten tasks and their dependencies. In this figure, the left column shows level numbers calculated by equation 1. The right column is obtained by counting tasks in each level starts from the exit task. In this example $N_{max}=5$ which is the maximum level in the workflow. Also, a budget of 165 is assumed.

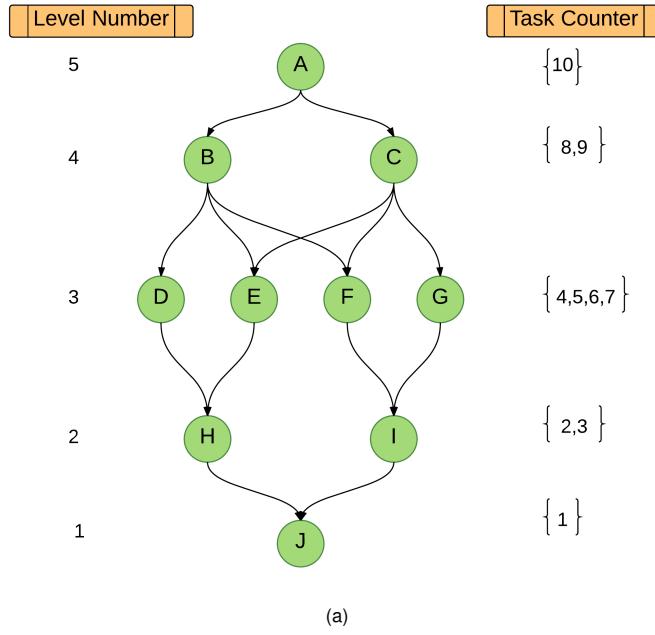


Fig. 1: A Sample Workflow with 10 tasks.

Each strategy distributes the user budget based on the following basis (see Table II for the complete set of budget shares):

- Uniform Proportional: Each level gets $165/5$ share of budget as our workflow has five levels.
- Height Proportional: Each level is assigned a weighted share of budget relative to its height in the workflow. This is calculated by:

$$L_{weight} = \sum_{k=1}^{N_{max}=5} k = 15.$$

The Budget Factor (BF) is calculated by:

$$BF = \frac{budget}{L_{weight}} = \frac{165}{15} = 11.$$

For instance, level 4 consisting task B and C are assigned a share of the budget equal to $4 \times BF = 4 \times 11 = 44$.

- Width Proportional: Each level gets a share of budget depending the number of tasks in corresponding level:

$$BF = \frac{budget}{tasknumbers} = \frac{165}{10} = 16.5.$$

For instance, the budget share assigned to level 4 with two tasks is $2 \times BF = 2 \times 16.5 = 33$.

- Area Proportional: In this strategy, the budget share allocated to each level is a combination of height and width strategies. Calculated by:

$$L_{weight} = \sum_{k=1}^{10} k = 55.$$

The Budget Factor (BF) is calculated by:

$$BF = \frac{budget}{L_{weight}} = \frac{165}{55} = 3$$

The budget then is distributed based on the sum of numbers in the right column in Fig. 1. For example, level 3 is allocated the share $(4+5+6+7) \times BF = 22 \times 3 = 66$.

- All in: The total budget is assigned to level 5. After scheduling all tasks in this level, any spare budget is trickled to the next level.

TABLE II: Budget distribution for each strategy over each level for a total budget of 165 in Figure 1.

	Budget Distribution Strategy				
	Uniform	Height	Width	Area	“All in”
Level 5	$\frac{165}{5} = 33$	$5 \times BF = 55$	$1 \times BF = 16.5$	$10 \times BF = 30$	165
Level 4	$\frac{165}{5} = 33$	$4 \times BF = 44$	$2 \times BF = 33$	$17 \times BF = 51$	0
Level 3	$\frac{165}{5} = 33$	$3 \times BF = 33$	$4 \times BF = 66$	$22 \times BF = 66$	0
Level 2	$\frac{165}{5} = 33$	$2 \times BF = 22$	$2 \times BF = 33$	$5 \times BF = 15$	0
Level 1	$\frac{165}{5} = 33$	$1 \times BF = 11$	$1 \times BF = 16.5$	$1 \times BF = 3$	0

C. Task Selection

In BDT, tasks are executed level by level which means a task can start execution once all tasks in previous levels have been scheduled. There are no dependencies between tasks that are at the same level. Therefore, all of them are ready to execute and are put to the task ready list. To select a task, at first, all tasks in the ready list should be prioritized. In this paper, tasks are prioritized based on their Earliest Start Time (EST).

It could be argued that it is pointless to prioritize task as all ready tasks are independent and their parent tasks have been

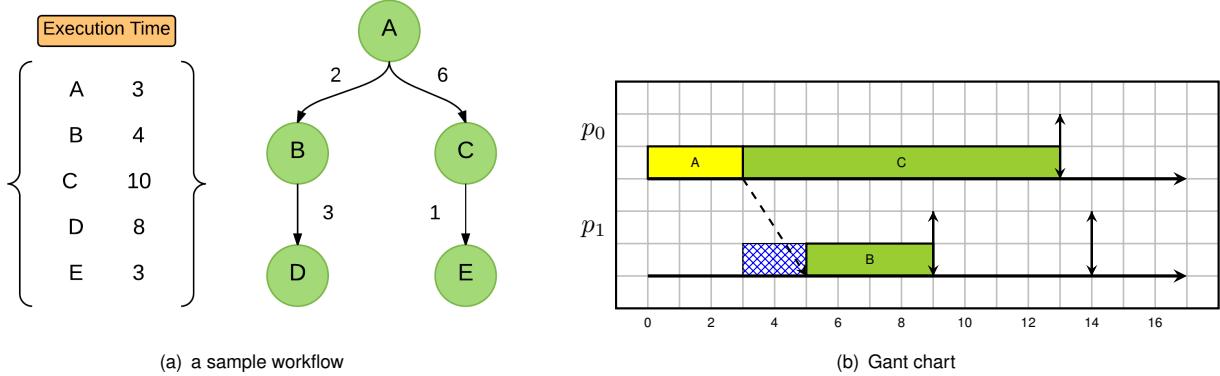


Fig. 2: Task Selection Example

already scheduled, but this is not the case. Let us trace the execution of a sample graph shown in Fig. 2. The number above each edge shows the data transfer time between tasks. Upon executing tasks A, B and C, all its children are placed in the ready list for execution. Note that due to execution of task A and C on the same instance, data is local, and therefore the transfer time is essentially zero. However, task B must wait to receive data from its parent (shown in blue intervals). Task D can start its execution on instance p_0 and p_1 at time 13 and 9, respectively (shown by \uparrow). The earliest time that task E can start on instance p_0 is 13, and on instance p_1 is 14. Therefore, we give a higher priority to task D.

The Earliest Start Time (EST) of a task t_i is calculated on the instance with the shortest execution time and defined as:

$$EST(t_i) = \begin{cases} 0 & , t_i = t_{entry} \\ \max_{t_j \in pred(t_i)} \{EST(t_j) + w_{t_j} + C_{i,j}\} & , \text{otherwise} \end{cases} \quad (3)$$

where w_{t_j} is the execution time of task t_j on the fastest instance type. The amount of data transferred from task t_i to task t_j is called communication time (denoted by $C_{i,j}$).

Task selection starts from the level that consists t_{entry} . After executing the first level, all tasks in the next level are put in the ready list to be scheduled.

D. Instance Selection

The BDT algorithm attempts to minimize the execution time while meeting the budget. Each level receives a computed budget share, that is the maximum that is able to be spent for the tasks within this level. We start by calculating both the time and the cost of executing each task on each instance type, given by equations 4 and 5, forming two sets of Cost and Time.

In the equation 4, $subBudget$ is a share of the budget that assigned to a level. The cost of scheduling for the current task, t_i , on the instance p_j is shown by C_i . The minimum cost of executing current task among all instances is C_{best} .

$$Cost_{t_i}^{p_j} = \frac{subBudget - C_i}{subBudget - C_{best}}. \quad (4)$$

In the Time set, the required time for the current task on instance p_j is a function of $ECT(t_i, p_j)$ and is expressed in equation 5. The maximum and minimum completion time of executing the task t_i among all instances are $ECT(max)$ and $ECT(min)$, respectively.

$$\text{Time}_{\frac{p_j}{t_i}} = \frac{ECT(max) - ECT(t_i, p_j)}{ECT(max) - ECT(min)}. \quad (5)$$

To find the best instance, we use the Time Cost Trade-off Factor (TCTF) in equation 6.

$$\text{TCTF}_{\frac{p_j}{t_i}} = \frac{\text{Time}_{\frac{p_j}{t_i}}}{\text{Cost}_{\frac{p_j}{t_i}}}. \quad (6)$$

There is a possibility that the total assigned budget for the level ℓ has already been spent ($subBudget=0$) while there are still some unscheduled tasks. If this condition is true, it makes equation 4 zero. Therefore, we can not launch a new instance as there is no budget left. Note that the value of equation 6 becomes zero as well.

When an instance is provisioned, the user is charged for the entire billing interval even if the task completes before the end of the interval. One way to reduce the cost of executing tasks is by using leftover capacity (residuals) in provisioned instances that have been already paid for. Therefore, if other tasks can execute on an existing instance with a residual, their execution costs can be considered zero. Moreover, the utilization of cloud resources depends on how tasks are placed together. Instance fragmentation and resource wastage occurs if tasks are not packed efficiently. The BDT algorithm utilises these residuals for executing ready tasks, which reduces makespan at no additional cost.

An important concept in our algorithm is trickling down unused budget and this is expressed by equation 7. We define Spare Budget (SB) as the amount of money remains after allocating all tasks in the level ℓ . We then add the leftover to the next level ($\ell + 1$).

$$SB = subBudget_\ell - \sum_{t_i \in TLS(\ell)} C_i. \quad (7)$$

IV. EVALUATION

We use five common scientific workflows: Cybershake, Montage, LIGO, Epigenomics and SIPHT to evaluate the performance of our algorithms under realistic load. The characteristics of which have been analyzed in [7]. We use CloudSim [8], configured with one data-center and six different instance types. The characteristics of these instance types are based on the EC2 instance configurations presented in Table I. The average bandwidth between instances is fixed to 20 MBps, based on the average bandwidth provided by AWS [9]. The processing capacity of an EC2 unit is estimated at one Million Floating Point Operations Per Second (MFLOPS) [10].

The estimated execution times are scaled by instance type CPU performance. In an ideal cloud environment, there is no provisioning delay in resource allocation. However, some factors such as the time of day, operating system, instance type, location of the data center, and number of requested resources at the same time, can cause delays in startup time [11]. Therefore, in our simulation, we adopted a 97-second boot time based real world measurements of EC2 [11].

To evaluate the budget sensitivity of the BDT algorithm and associated strategies, we considered different budget ranges for the scientific datasets from lowest possible through to sufficient. The lowest possible budget to schedule a workflow is given by equation 8.

$$Lowset_{cost} = \sum_{\forall t_i \in G} Cost_{t_i} p_j, \quad (8)$$

where p_j is the cheapest instance. To achieve this, all tasks are executed on the instance with the lowest cost (the cheapest instance). This assignment gives us the lowest possible cost required for executing a workflow, irrespective of finishing time. Using this lowest cost, we then calculate the minimal budget as follows:

$$budget = \alpha * Lowset_{cost} \quad 1 < \alpha < 10. \quad (9)$$

The budget range starts from 1.5 to consider minimum budget with increasing step length of 0.5. The EC2 instances charge hourly basis from the time of provisioning, even if the instance is only used for a fraction of that period. We ran the simulations with lease times of 15, 30, 45 and 60 minutes to evaluate the sensitivity of the algorithm to the length of the lease.

To compare performance with respect to workflow size we evaluated workflows with 200, 500 and 1000 tasks. However, as these results did not vary significantly, we present here only workflows with 1000 tasks. We used the Pegasus workflow generator [7] to create representative synthetic workflows with the same structure as real world scientific workflows (Cybershake, Montage, LIGO and SIPHT). For each workflow structure, and each budget range, 100 distinct Pegasus generated workflows were scheduled in CloudSIM and these results are detailed in the following section.

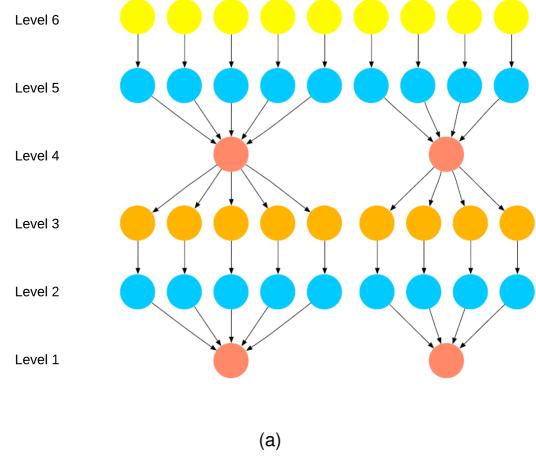


Fig. 3: A simple structure of LIGO with six levels

A. Analysis for LIGO

The Laser Interferometer Gravitational Wave Observatory (LIGO) attempts to detect gravitational waves produced by various events in the universe as per Einsteins theory of general relativity [7]. A simple LIGO-like workflow with six levels is shown in Figure 3.

For a LIGO workflow with 1000 tasks, Table III gives the levels with the corresponding task's numbers. We select the budget range of 5 with the corresponding budget of 7.035 to evaluate the budget distribution strategies. The share of budget that each level gets and the spare budget is also given in Table III. For instance, after scheduling of all tasks in the first level using the Height Proportional strategy, the spare budget of 0.15 (indicated in red) is added to assigned budget of next level (shown in blue) resulting in the total budget of 1.69 for that level (the budget trickling concept). The last row in Table III shows the makespan that was achieved by each strategy.

It is also interesting to look at which instance types and how many of each were provisioned by each strategy, this is given in Table IV. The most significant observation is that the “All in” strategy allocates a small number of powerful instances, reducing the data transfer costs and achieving the lowest makespan as given in Table III. From users’ perspective, finding a schedule with a lower makespan for a given budget is the main concern, while from the providers perspective, maximizing utilization is the main concern. Very few research papers report the total number of instances that a workflow needs when provisioning. Although all BDT strategies meet the budget (only possible due to trickle down) and find a schedule, the type and the amount of requested VMs are very different as shown in Table IV. Launching too many instances does not lower makespan. Instead, it causes high scheduling overhead and low instance utilization. Therefore, the budget distribution strategy has a direct impact on resource utilization - which is significantly important.

The makespan and success rate of LIGO for four specified

TABLE III: Example computed budget distribution for each strategy over each level of a LIGO for budget range=5 and budget=7.035

		Budget Distribution Strategy									
		Uniform		Height		Width		Area		"All in"	
	Task Numbers	sub Budget	Spare Budget	sub Budget	Spare Budget	sub Budget	Spare Budget	sub Budget	Spare Budget	sub Budget	Spare Budget
Level 6	229	1.173	0.01	2.01	0.15	1.61	0.01	2.85	0.01	7.035	0.001
Level 5	229	1.173	0.03	1.675	0.01	1.61	0.03	2.11	0.03	0.001	0.001
Level 4	21	1.173	0.05	1.34	0.03	0.14	0.006	0.15	0.01	0.001	0.001
Level 3	250	1.173	0.01	1	0.02	1.75	0.01	1.39	0.02	0.001	0.001
Level 2	250	1.173	0.03	0.67	0.009	1.75	0.02	0.51	0.06	0.001	0.001
Level 1	21	1.173	0.05	0.335	0.02	0.14	0.02	0.003	0.05	0.001	0.001
Achieved Makespan		938.97		798.71		798.61		682.6		603.93	

TABLE IV: VM requested types by different strategies based on Table III

Budget Distribution Strategy					
Vm Type	Uniform	Height	Width	Area	"All in"
m3.medium	1	1	6	2	0
c4.large	6	3	2	2	1
c3.xlarge	6	3	2	3	1
m4.2xlarge	0	2	2	2	0
c4.4xlarge	6	2	2	2	0
c3.8xlarge	0	2	2	2	4
#VMs	19	13	16	13	6
Total Cost	6.985	7.006	7.026	6.968	7.035

lease times are shown in Fig 4. The "All in" strategy again performs significantly better than other strategies for each of the defined budget factors from low to high values. The random strategy has the worst performance for both makespan and success rate. Another observation is that the general trend of all strategies for different instance lease times is largely similar.

B. Other workflows

The makespan and success rates of other workflows are presented in Fig. 5. The general trend is that by increasing the budget, a scheduler can launch more costly services, which in turn leads to a lower makespan.

In terms of makespan, for almost all workflows, the "All in" strategy has the best performance including lowest makespan and highest success rates. An interesting observation is that the structure and type of workflow appear to have a significant effect on success rate. For instance in CYBERSHAKE, the success rates of most of the strategies are generally poor, this is likely due to it being a data intensive workflow - the "All in" strategy of few, high power instances minimised the cost of data movement and produced the best makespan and success rate.

Overall, a budget bias towards the early tasks (and levels) of a workflow appears to produce better overall performance.

V. RELATED WORK

Scheduling in cloud environment encounters some challenges not present in traditional heterogeneous environments such as the Grid or HPC clusters. The cost model and

resource provisioning in cloud are among the main challenging differences. For instance, pricing schemes in cloud systems are based on lease intervals from the time of provisioning, even if the instance is only used for a fraction of that period. A significant number of cost aware workflow scheduling algorithms in Grid have been proposed such as [4], [12]. However, cost in Grid is calculated based on the accumulated cost of requested services [13]. Moreover, workflow scheduling in grid focuses on minimizing the makespan without considering the cost [14].

The most similar (although not very) related work to BDT appears in [3] and [15] whereby Zheng et al. proposed Budget constrained Heterogeneous Earliest Finish Time (BHEFT) which is an extension of HEFT algorithm [14]. In BHEFT, a current task budget (CTB) factor is introduced to distribute spare budget among unscheduled tasks. Their budget distribution is different from ours as the task budget and spare budget are calculated task by task. Moreover, their work is set within the context of a Grid environment which is not directly applicable to cloud environments due to differences in the cost model.

In cloud environments, most of the research focuses on QoS constrained workflow scheduling for deadline and budget. The main idea of most deadline constrained algorithms is how to distribute the user-defined deadline among workflow tasks or levels [5], [6], [16], [17]. In this paper, we are focusing on budget constrained algorithms for cloud environments.

In [18], Zeng et al. presented a budget-aware backtracking algorithm for executing large scale many task workflows, referred to as ScaleStar. The authors in [19] and [20] presented an algorithm with budget constraints called minimum end-to-end delay under cost constraint (MED-CC). Firstly, each task in a workflow is assigned to an instance. In the next step, all critical tasks are considered for rescheduling with the proposed Critical Greedy algorithm.

The cost model considered in [18]–[20] is based on the use of fractional resources. For instance, in [20] a base processing unit which is ten instructions per time unit is set with price at 0.01 per time unit. However, in most of the cloud providers, like Amazon EC2, users are billed based on a longer interval like one hour.

In [21], Li et al. presented an extension of the HEFT [14] algorithm called the Cost Conscious Scheduling Heuristic

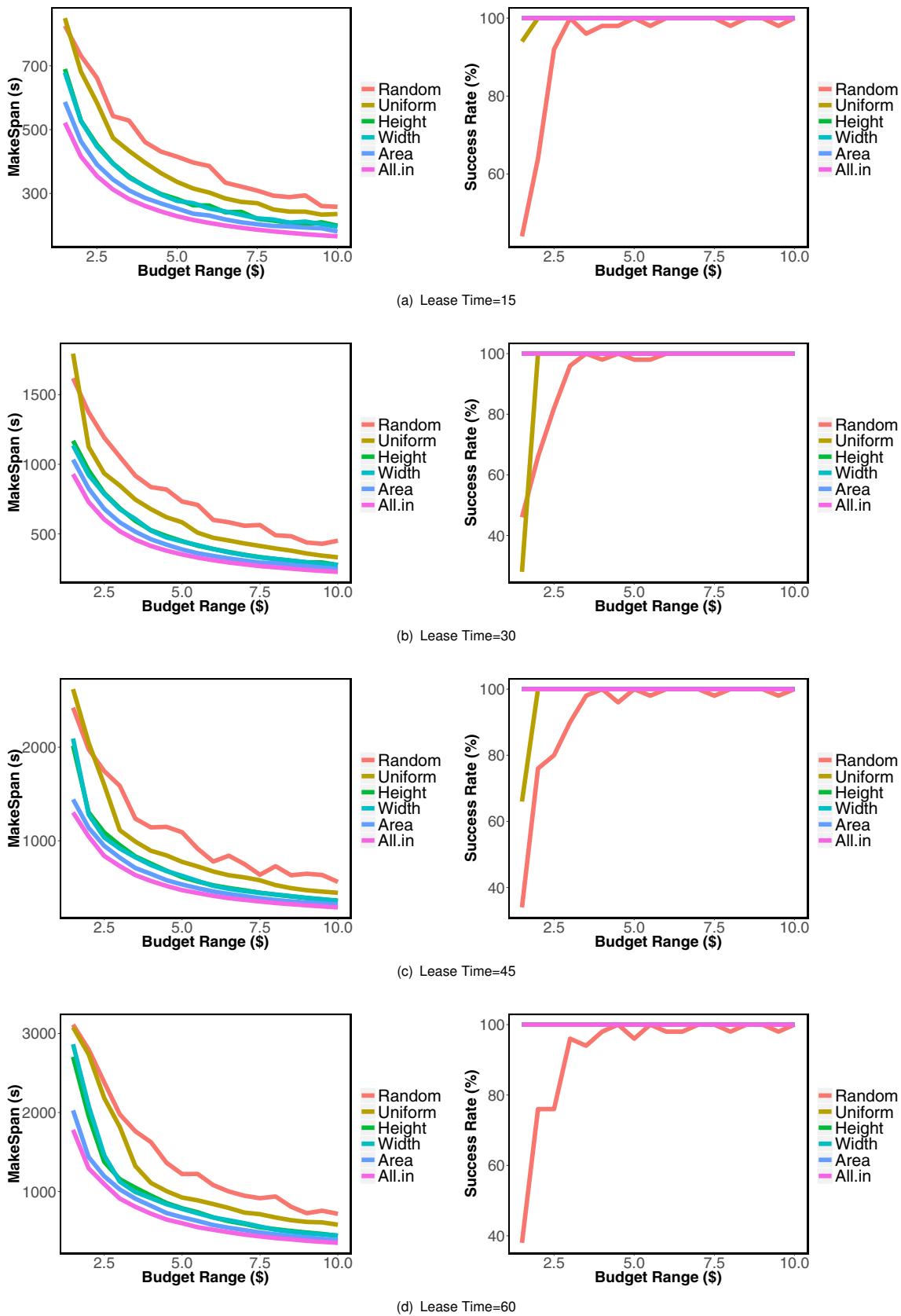


Fig. 4: Makespan and Success rate performance executing LIGO for all strategies for lease time of 15, 30, 45 and 60

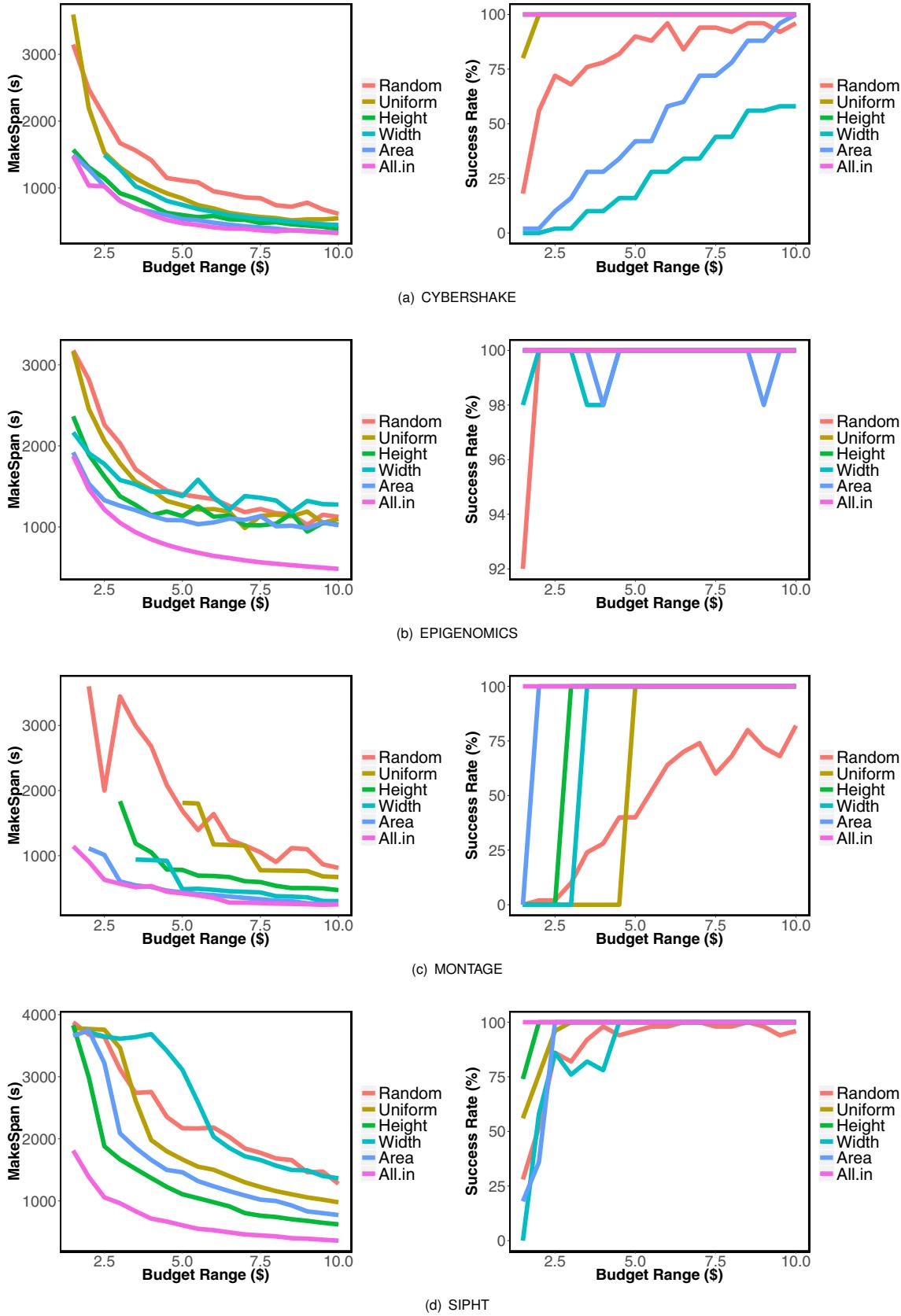


Fig. 5: Makespan and Success rate performance executing of workflows for all strategies for lease time 60

(CCSH). The CCSH algorithm, first constructs a priority list of tasks and then assigns the task with the highest priority value to the most cost-efficient virtual machine (VM). However, only one VM type and one pricing model is considered.

In [22] two auto scaling techniques to solve the budget constrained scheduling for a workload consisting multiple workflows were proposed. In this work, budget is distributed to different workflows proportionally based on assigned priorities.

Scheduling Bags of tasks under budget constraints in cloud is presented in [23]. One of the assumptions considered by authors is tasks are preemptive which mean they can be interrupted, delayed and then retriggered sometime later. Our model is different from [23] in such a way, we have non-preemptive dependent tasks in our workflow model, a less forgiving constraint.

VI. CONCLUSION

In this paper, we introduced the Budget Distribution with Trickling (BDT) algorithm that presents new notions for distributing budget based on the dependency structure inherent in workflows – levels. In addition we proposed several new strategies for sharing or distributing budget over these levels. We also proposed trickling to redistribute unspent budget down to other levels.

We evaluated the makespan and success rate of six strategies using five real-world workflows with different lease times on instances. The width and uniform strategies are largely analogous to existing budget distribution approaches and these are outperformed in makespan and success rate by Area and “All in” BDT strategies. Although, width and uniform also benefit from trickling and the other innovations built into BDT. Overall, the strategy that performed the best for all workflows in terms of both success rate and makespan was “All in”. This strategy was proposed largely to test the hypothesis that biasing the budget distribution to the earliest levels was beneficial in terms of reducing makespan. “All in” took this to an extreme by assigning the entire budget to the first level and relying wholly on the trickle down mechanism to distribute budget to later levels. Counterintuitively, “All in” gave the best success rates - and the best performance for data intensive workflows due to the resulting data locality, from fewer more powerful instances.

The main finding of our research is in the importance of biasing budget distribution to early levels in a workflow. This leads to finding a schedule with a lower makespan. From this we have extrapolated two hypotheses that need further investigation. In addition, these results suggest that gaining a better understanding of workflow types (compute and data intensive) and workflow structure can lead to better budget distribution and likely scheduling in general.

REFERENCES

- [1] I. J. Taylor, E. Deelman, D. Gannon, and M. Shields, *Workflows for e-Science*. Springer-Verlag London Limited, 2007.
- [2] E. Deelman, “Grids and clouds: making workflow applications work in heterogeneous distributed environments,” *International Journal of High Performance Computing Applications*, vol. 24, no. 3, pp. 284–298, 2010.
- [3] W. Zheng and R. Sakellariou, *Economics of Grids, Clouds, Systems, and Services: 8th International Workshop, GECON 2011, Paphos, Cyprus, December 5, 2011, Revised Selected Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, ch. Budget-Deadline Constrained Workflow Planning for Admission Control in Market-Oriented Environments, pp. 105–119. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-28675-9_8
- [4] R. Sakellariou, H. Zhao, E. Tsakkouri, and M. D. Dikaiakos, *Integrated Research in GRID Computing: CoreGRID Integration Workshop 2005 (Selected Papers) November 28–30, Pisa, Italy*. Boston, MA: Springer US, 2007, ch. Scheduling Workflows with Budget Constraints, pp. 189–202. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-47658-2_14
- [5] Y. Yuan, X. Li, Q. Wang, and Y. Zhang, “Bottom level based heuristic for workflow scheduling in grids,” *Chinese Journal of Computers-Chinese Edition-*, vol. 31, no. 2, p. 282, 2008.
- [6] J. Yu, R. Buyya, and C. K. Tham, “Cost-based scheduling of scientific workflow applications on utility grids,” in *e-Science and Grid Computing*, 2005. First International Conference on, July 2005, pp. 8 pp.–147.
- [7] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, “Characterizing and profiling scientific workflows,” *Future Generation Computer Systems*, vol. 29, no. 3, pp. 682 – 692, 2013, special Section: Recent Developments in High Performance Computing and Security.
- [8] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, “CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,” *Software: Practice and Experience*, vol. 41, no. 1, p. 2350, 2011.
- [9] M. R. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel, “Amazon S3 for science grids: a viable solution?” in *Proceedings of the 2008 international workshop on Data-aware distributed computing*. ACM, 2008, pp. 55–64.
- [10] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, “A performance analysis of ec2 cloud computing services for scientific computing,” in *Cloud Computing*, ser. Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, D. Avresky, M. Diaz, A. Bode, B. Cicani, and E. Dekel, Eds. Springer Berlin Heidelberg, 2010, vol. 34, pp. 115–131.
- [11] M. Mao and M. Humphrey, “A performance study on the vm startup time in the cloud,” in *Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing*, ser. CLOUD ’12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 423–430.
- [12] J. Yu and R. Buyya, “A budget constrained scheduling of workflow applications on utility grids using genetic algorithms,” in *2006 Workshop on Workflows in Support of Large-Scale Science*, June 2006, pp. 1–10.
- [13] I. Foster, Y. Zhao, I. Raicu, and S. Lu, “Cloud computing and grid computing 360-degree compared,” in *2008 Grid Computing Environments Workshop*, Nov 2008, pp. 1–10.
- [14] H. Topcuoglu, S. Hariri, and M.-Y. Wu, “Performance-effective and low-complexity task scheduling for heterogeneous computing,” *Parallel and Distributed Systems, IEEE Transactions on*, vol. 13, no. 3, pp. 260–274, Mar 2002.
- [15] W. Zheng and R. Sakellariou, “Budget-deadline constrained workflow planning for admission control,” *Journal of Grid Computing*, vol. 11, no. 4, pp. 633–651, 2013. [Online]. Available: <http://dx.doi.org/10.1007/s10723-013-9257-4>
- [16] V. Arabnejad and K. Bubendorfer, “Cost effective and deadline constrained scientific workflow scheduling for commercial clouds,” in *Network Computing and Applications (NCA), 2015 IEEE 14th International Symposium on*, Sept 2015, pp. 106–113.
- [17] V. Arabnejad, K. Bubendorfer, B. Ng, and K. Chard, “A deadline constrained critical path heuristic for cost-effectively scheduling workflows,” in *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, Dec 2015, pp. 242–250.
- [18] L. Zeng, B. Veeravalli, and X. Li, “Scalestar: Budget conscious scheduling precedence-constrained many-task workflow applications in cloud,” in *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on*, March 2012, pp. 534–541.
- [19] X. Lin and C. Q. Wu, “On scientific workflow scheduling in clouds under

- budget constraint,” in 2013 42nd International Conference on Parallel Processing, Oct 2013, pp. 90–99.
- [20] C. Q. Wu, X. Lin, D. Yu, W. Xu, and L. Li, “End-to-end delay minimization for scientific workflows in clouds under budget constraint,” IEEE Transactions on Cloud Computing, vol. 3, no. 2, pp. 169–181, April 2015.
- [21] J. Li, S. Su, X. Cheng, Q. Huang, and Z. Zhang, “Cost-conscious scheduling for large graph processing in the cloud,” in High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on, Sept 2011, pp. 808–813.
- [22] M. Mao and M. Humphrey, “Scaling and scheduling to maximize application performance within budget constraints in cloud workflows,” in Parallel Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on, May 2013, pp. 67–78.
- [23] A. M. Oprescu and T. Kielmann, “Bag-of-tasks scheduling under budget constraints,” in Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, Nov 2010, pp. 351–359.

Datatrack: An R package for managing data in a multi-stage experimental workflow

Data Versioning and Provenance Considerations in Interactive Scripting

Philip Eichinski, Paul Roe
 Science and Engineering Faculty
 Queensland University of Technology
 Australia
 philip.eichinski@qut.edu.au

Abstract—In experimental research using computation, a workflow is a sequence of steps involving some data processing or analysis where the output of one step may be used as the input of another. The processing steps may involve user-supplied parameters, that when modified, result in a new version of input to the downstream steps, in turn generating new versions of their own output. As more experimentation is done, the results of these various steps can become numerous. It is important to keep track of which data output is dependent on which other generated data, and which parameters were used. In many situations, scientific workflow management systems solve this problem, but these systems are best suited to collaborative, distributed experiments using a variety of services, possibly batch processing parameter sweeps. This paper presents an R package for managing and navigating a network of interdependent data. It is intended as a lightweight tool that provides some visual data provenance information to the experimenter to allow them to manage their generated data as they run experiments within their familiar scripting environment, where it may not be desirable to commit to a fully-blown comprehensive workflow manager. The package consists of wrapper functions for writing and reading output data that can be called from within the R analysis scripts, as well as a visualization of the data-output dependency graph rendered within the R-studio console. Thus, it offers benefit to the experimenter while requiring minimal commitment for integration in their existing working environment.

Keywords—computational science; data provenance; R language; R package, workflow;

I. INTRODUCTION

Exploratory data analysis research is often performed in interactive scripting environments such as R, Matlab, Octave or Python. These are popular because they afford the opportunity for researchers who do not necessarily have a strong software engineering or coding background to interactively manipulate data through a read–eval–print loop (REPL). They offer the flexibility of a programming language while allowing swift manipulation, inspection and visualization of data, enabling researchers to quickly roll-out experiments.

Exploratory data analysis in such scripting environments is often performed in a conceptual “workflow” where several distinct steps of data transformation are involved, each taking

one or more data objects, and later outputting some result data to be used as input for a subsequent downstream step.

In computational workflows, recording the data provenance - the dependencies, processes, parameters and people responsible for the creation of the data - is important, as it helps interpretation, verification, or tracing the origin of results [2].

When the research undertaken involves designing these analysis processes, inspecting the results of each step, tweaking code and reprocessing, the workflow is often not run in a single end to end execution, but rather, an individual step may be run in isolation, relying on saved intermediate data.

This necessitates that the input data objects be selected by the user, rather than piped in by the preceding step in the workflow. To make this selection, the user must be informed by the provenance metadata associated with all the potential candidates for selection.

Thus, data provenance is not only important from the point of view of reproducibility and debugging, but also to assist online decision-making during the execution of each step of the workflow. Having this provenance information easily available within the interactive scripting environment is of great benefit.

Existing solutions to workflow data provenance tracking normally involve scientific workflow management systems (SWfMS) such as Vistrails [3] or Kepler [4]. These systems are well suited to distributed systems, collaborative research and batch processing across a wide range of parameters, and are hugely advantageous for automating the cycle of moving data to a supercomputer for analysis or simulation, launching the computational processes and managing the storage of the output [5]. When working within a SWfMS, the workflow is formalized using a workflow language, which may be generated through a graphical user interface, and the management system invokes the processes of the workflow. This means that they are best suited where these processes are mature and stable.

However, exploratory research performed in interactive scripting environments, where the code that performs the analysis is being constantly modified, does not always lend itself to integration with a SWfMS. Even in circumstances

where a SWfMS could be used, it may be resisted by the researcher because integration will often draw the researcher out of the environment that they are comfortable working in and require additional effort to learn a new system. When the research calls for frequent modification to the code that runs the data processing steps, working in an SWfMS means effectively working in two environments possibly with two scripting languages.

Under these circumstances, a solution for automated data provenance recording is required for use in interactive scripting environments that do not use a SWfMS. Some tools have been proposed (discussed in section II), however while they address the aspects of data provenance related to verification and reproducibility, they are not aimed at online user decision-making, as discussed above.

This paper introduces **Datatrack**, a prototype R package that [6] manages the dependencies and versioning of data generated in R. It was created to address the problems with data provenance tracking in the interactive exploratory research performed in our lab. The purpose of Datatrack is not to formalize the workflow but rather to provide a way to automate some record-keeping of data generation and assist the experimenter in selecting the correct input data when executing a process. While it doesn't offer many of the features of most scientific workflow management systems, it has the advantage that it has minimal configuration and operates within the R programming environment, reducing the barrier to entry for researchers.

II. RELATED WORK

With the growing importance of e-science and computational science, standards for provenance have emerged. In computationally intensive science, large amounts of data are generated. Data provenance allows the scientist to determine all necessary information about the input data, processes, computing environments and contributors involved in the generation of data output in order to be able to reproduce it. Broadly speaking data provenance captures the identities and relationships between **what** was created, **how** it was created and **who** played a role in its creation.

Standard models for how this information should be structured have emerged, such as the *Open Provenance Model* (OPM) developed following the International Provenance and Annotation Workshop in 2006 [1] and PROV-DM a more recent standard, endorsed by the W3C in 2012 [7]. Both these standards list three types of core 'objects' that represent the what, how and who of provenance, albeit with differing terminology: respectively, *Artefacts*, *Processes*, *Agents* in the case of OPM and *Entities*, *Activities* and *Agents* in the case of PROV-DM. They also define a number of causal relationships between them, such as "was derived from", "was controlled by", "was triggered by" amongst others. The standards specify how to model provenance and not how to implement the model.

Most research into the issues of data provenance has been focussed on Scientific Workflow Management Systems (SWfMS) [8]. There is a huge variety of these SWfMS, but generally they have the following functionality: workflow

composition, mapping the workflow onto resources or services, executing the workflow and recording the provenance metadata to allow the final output to be reproduced in the future [5]. As discussed in section I, SWfMS are well suited to large, collaborative, distributed analysis and simulations, a situation where good provenance recording is indeed vital.

Until recently, literature in data provenance tracking *outside* of SWfMS has been scarce. Part of the reason for this may be that research on a scale small enough that a SWfMS is not required means that keeping track of data provenance may be assumed to be trivial. Yet, in our experience, simple, user-friendly tools for automatically maintaining a record of dependencies of data outputs has proved hugely useful.

NoWorkflow [9] is a command line tool for recording detailed provenance metadata from python scripts. Its authors note that outside of a SWfMS, provenance capture is challenging due to the fact that the workflow sequence is encoded by the scripts themselves. NoWorkflow works by using software engineering techniques such as abstract syntax tree analysis, to determine this workflow from the scripts themselves and record provenance information during their execution.

The YesWorkflow [10] toolkit is another tool that offers some of the provenance recording functionality of a SWfMS to users of scripting languages such as R and Python. YesWorkflow allows the scientist to insert annotations as code-comments in the scripts that they write. The toolkit can then parse the codebase and interpret and convert these comments into a form that can be queried to answer questions about data objects created.

These tools share with Datatrack the benefit of minimal intrusion into the programming practices that the user is comfortable with. However, Datatrack comes from a slightly different angle, with emphasis put on provenance information for decision-making during invocation of sections of the workflow in isolation.

III. MOTIVATION CONTEXT

The development of Datatrack was motivated by needs arising in the research undertaken at the Ecoacoustics Research Group at the Queensland University of Technology, which partly entails designing automated and semi-automated methods of acoustic analysis for environmental monitoring. These methods may involve several data processing steps such as pre-processing, feature extraction, silence removal, clustering and sample ranking, for example. Each processing step performs a series of operations on the output of one or more previous steps, and outputs one or more of its own data files. The R language has often been chosen within the Eco-acoustic Research Group at QUT because it offers the power and flexibility of a programming language but is easy to learn and provides very quick methods for manipulating and visualizing data.

A change at any particular point along the pipeline will generate new versions of all intermediate data downstream to the change. A new idea by the researcher might mean creating a new process that shares some upstream data as a dependency, and creates new branches downstream. A particular step may

take multiple input data objects. For example, a classifier will read in both a model on which to base the classification as well as the data points to classify. Both of these inputs will have been generated by one or more earlier processes.

Although such a sequence of steps comprises a “workflow”, it has not been formally specified in a dedicated workflow language. Fig. 1 shows an example of the kind of workflow used.

The workflow is not necessarily run from start to finish; one step of the workflow may be invoked in isolation. This may be because the code for that step may have been modified, or different parameters used. Running a process at a particular point along the pipeline after such changes should not require regeneration of the upstream data, and therefore it is normally desirable to save all intermediate input/output data. Not only does this ‘caching’ of intermediate data save the time needed to compute it, but it allows inspection of the output of each step in order to evaluate and improve the process that created it.

With every modification to parameters of any process in the workflow, the number of saved data objects in this

interdependent network increases, and keeping track of them can become unwieldy. Obviously, in the absence of automated management, diligent manual recordkeeping would be necessary to ensure reliable reproducibility.

But the primary motivator to the development of Datatrack was to assist selection of input data. When invoking a particular processing step in the workflow, the researcher must decide which data to use as input out of a large number of possible inputs available (generated from variations of upstream processes). Without an easy method for the researcher to ascertain the dependencies of a particular data object, their interaction with their software is less user-friendly. Easily accessible provenance metadata at the time that they are selecting the input data makes this interaction quicker and less error-prone.

Git version control is used to track changes in the codebase over time. While this offers a fair degree of retrospective examination of *how* past results were generated, additional metadata of how the data is related to the code in the repository is required.

IV. DATATRACK FRAMEWORK

In this section, we discuss the prototype package that addresses some of the problems detailed above. The package has been developed iteratively to address specific needs that have arisen during our ecological acoustics research.

A. Overview

From the user’s perspective, Datatrack provides wrappers for reading and writing data in R. When these are used instead of the native R functions (such as `write.csv`), additional metadata can be supplied. Datatrack records this supplied metadata as well as some additional information that can be determined from the data and environment. An interactive visual graph representation of the dependencies is available. It is automatically displayed when reading data to allow the user to specify which version of available data objects should be used.

The philosophy behind this simple approach is that the user should not be restricted in their coding style. It does not force the researcher into a particular set of conventions, or to work in an unfamiliar environment.

Notably, there is no formal workflow language or authoring tool. In effect, R is the workflow language as well as the language of the processes that perform the analysis tasks themselves.

Datatrack is completely data-centric. It does not attempt to infer the workflow by interpreting source code. Instead it records the data flow, i.e. the dependencies between data objects. This is in line with its goals to provide the necessary information to the user at the time that data is read in.

B. Writing Data

When Datatrack is used to save the output of an analysis process, several pieces of information are supplied to it:

- Name
- Dependencies

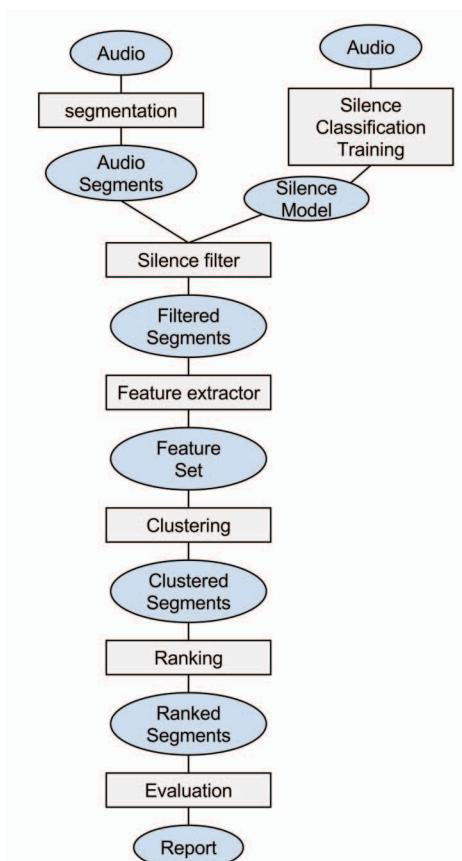


Fig. 1. Example of a workflow for ecological audio analysis experimentation. Rectangles denote processes and ellipses denote data objects that are input and output to these processes.

- Parameters
- Annotations

The **name** is an arbitrary string used to refer to the data at a later stage. It would typically describe the data object that has been generated or the process that generated it, for example “clustering.result” or “ranked.samples”.

Dependencies are a list of names and version-numbers of the data objects that were read in and used in the processing step that created the data being written. This is the mechanism that defines the dependency graph of data output for the workflow.

Parameters are a set of name-value pairs, and refer to the parameters used by the process that created the output data.

Annotations are also name value pairs that the user can supply to be stored as the metadata of the process.

During experimentation, *dependencies* or *parameters* might be altered by the user on separate invocations of a processing step. When writing data of a particular *name*, Datatrack will save a separate version of the output for each unique combination of *dependencies* and *parameters*. The versioning is handled internally to the package. It is not possible to save output data that has the same name, dependencies and parameters as a previous version has, and doing so will result in a user-prompt asking for approval to overwrite the existing file. The distinction between annotations and parameters is that annotations do not affect versioning. Example R code is for writing data is shown in Fig. 2.

C. Reading Data and the Data Dependency Graph

When reading in upstream data in any step of the pipeline, the *name* is specified in a Datatrack function (Fig. 3). There may be many versions of available data objects with a given name, each having been generated with a unique combination of *dependencies* and *parameters*.

Execution is paused until the user inputs the desired *version*, which is chosen from a list of available versions. This is done made with the aid of a visual graph of data

```
params <- list(k = 240, alg = 'kmeans')
dep <- list(segment_features = seg$version)
datatrack::writeDataobject(mydata,
                           name = 'clustering.1',
                           params= params,
                           dependencies = dep,
                           annotations = list())
```

Fig. 2. Example R code for saving data

```
datatrack::readDataobject('clustering.1')
```

Fig. 3. Example R code for reading data

dependencies and associated metadata, which is presented to the user at the time data is being read through Datatrack. It is a directed acyclic graph (DAG) where the nodes are data objects, and the edges are dependencies, with the vertical position denoting the direction of the dependencies (dependencies are above the data objects that depend on them). Nodes are grouped by their *name*. The dependency graph is shown for the purpose of assisting the user in their selection. The user can easily see the versions of the potential inputs available, the parameters used when generating them, and the annotations attached to the data, as well their dependencies. The graph also provides the user with convenient access some information that can be derived directly from the data object, such as the column names of saved CSVs, which can also help in their choice of which version to read in.

Because multiple versions of saved data objects are shown simultaneously for comparison, the number of relevant data nodes on the graph can be quite large. This has motivated the design decisions aimed at minimising clutter in the dependency graph.

Fig. 4 shows an example of a graph. In this example, there are three groups of data objects that do not have any dependencies: “weather”, “radar.wthr” and “audio”. These names are defined by the user as the ‘name’ argument to the `writeDataobject` function (Fig. 2). Each of the numbered cells in the groups below the group name is a node of the graph and represents a data object. For instance, in the example shown in Fig. 4 there are six versions of “event.features.1” available, each created with different dependencies and/or parameters. By hovering the mouse over a node, the user can inspect the metadata attached to that version and easily ascertain its dependencies and dependents. In the example, the mouse pointer is hovering over version 1 of “event.features.1” and the direct dependency (version 1 of “events”) and indirect dependency (version 1 of “audio”) are highlighted. The directly dependent nodes (versions 1 and 3 of “clustering”) and indirectly dependent nodes (version 1 of “ranking”) are also highlighted. This graph is interactive and is displayed within the viewer of R-Studio (Fig. 5), a popular integrated development for R [11].

Nodes can be filtered by name, hiding nodes that do not have any direct or indirect links to any node in the group with the selected name. Fig. 5 shows the same graph as Fig. 4 but filtered by name “event.features.2”. When reading data of a particular name, this filtering is performed to remove nodes that are irrelevant to the choice the user needs to make. Within a group, nodes are ordered by their created date-time, and a range can be specified to filter nodes of the selected group by date-time

Important to note is that the conceptual workflow conceived by the researcher and implemented in R will *resemble* the groups and their relationships in this graph, however the Datatrack data dependency graph is not based directly on any workflow designed by the user, but rather only on the dependencies declared at the time of writing the data through the data object’s *name*. No assumptions are made about the workflow, and no attempt at inferring the workflow through code inspection is made. The user is free to choose

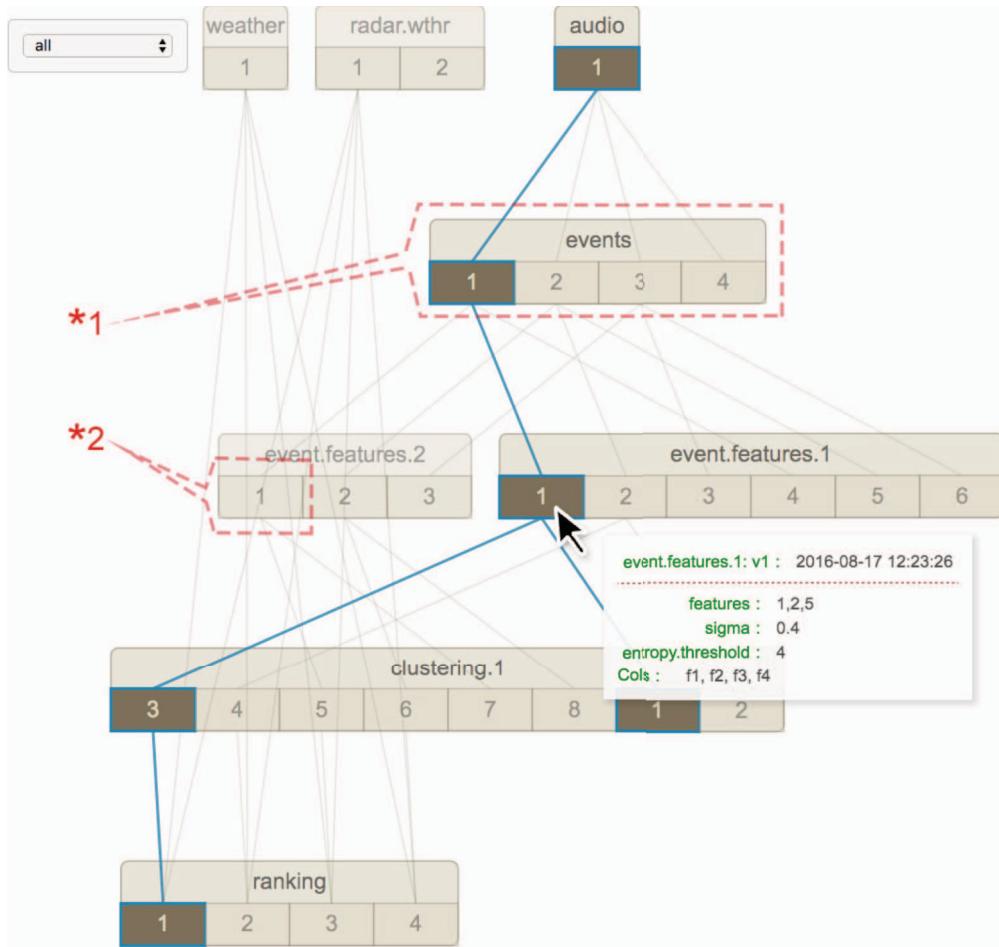


Fig. 4. Example of a graph of data dependencies showing parameters used when generating data.). Each group of nodes (*1) represents a type of data object arbitrarily named by the user. This example shows eight different types of data object. Each node (*2) represents a data object. The nodes are numbered by their version. Dependencies between nodes are represented by the lines. When the mouse hovers over a node, metadata information is for that node is shown, and the dependency chain to and from that node is highlighted. Note: the dashed lines are figure labels and are not part of the graph visualization.

whatever name they like, although normally it would refer in some way to the step of the workflow. This means that the working style of the user is not impacted at all by using Datatrack for reading and writing data.

While giving the user the provenance information at the time that data object is being read in to a process is useful when that step of the workflow is being run in isolation, if multiple steps of the workflow are being executed in sequence or if the workflow is being executed from start to finish, there is no need to prompt the user to select the correct version of data. Unless instructed otherwise, Datatrack will automatically select the last accessed version of a data object of a particular name (read or written), instead of prompting for user selection. This allows an entire workflow of reading and writing data to be run without user input at each step.

D. Considerations and Tradeoffs

Datatrack is not designed to offer all the data provenance features of a Scientific Workflow Management System, because its primary aim is centered around user decision-making assistance and data versioning. In this section, we discuss features and their implications for usability of Datatrack.

1) Tracking of users: the “who” of provenance

The Open Provenance Model and PROV-DM specifications define a number of ‘objects’ that do not show up as nodes on the Datatrack dependency graph, namely the “who” (which users) and the “how” (which processes), leaving only the dependencies between the “what” (the generated data). This difference can be seen by comparing the toy example given in the OPM specification (Fig. 6) [1]. As well as displaying *artefacts* (e.g. ‘cake’), it also shows the *process* ‘bake’ and the *agent* ‘John’.

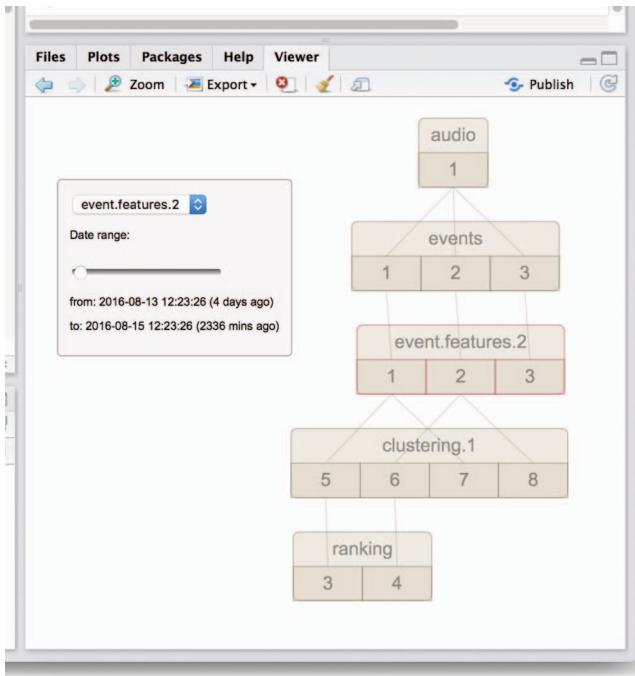


Fig. 5. Data dependency graph within the R-Studio viewer

In making design decisions about what should be recorded and what should be present as nodes in the dependency graph, some trade-offs were made between simplicity and completeness. The aim of Datatrack is to only include nodes in the dependency graph that are central to satisfying the needs for which it was built, namely providing on-the-fly assistance in selecting the correct input when running a process. Other provenance metadata is recorded in the background, such that at a later stage it can be recovered for debugging purposes.

Complete provenance tracking includes information regarding who was responsible for a process that generated intermediate data. This is desirable in collaborative environments, however it requires more configuration by the experimenter. By allowing arbitrary user-supplied metadata to be recorded when data is written (passed to Datatrack's write function as *annotations*), this “who” information can be optionally recorded. This metadata may help a user to navigate through the dependency tree by giving extra information, but is not included as nodes in the dependency graph unlike a fully compliant OPM graph. This decision was made to simplify the graph for readability.

2) Tracking of code versions and environment information

For provenance tracking to be complete, the experiment should be completely reproducible. This requires that all the algorithms that played a role in the creation of the output data, as well as the environment in which they were run, should be recorded in the provenance metadata.

Datatrack keeps things simple and lightweight by simply recording the *date that the data object was generated*, and a *stack trace* of function calls leading to the writing of data. In conjunction with a versioning system (in our case Git), this

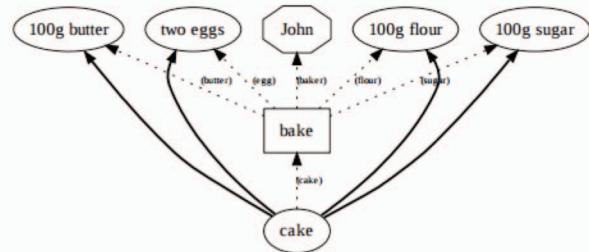


Fig. 6. Example of a workflow graph that adheres to the OPM specification [1]

does reasonably well in allowing the details of the processes responsible for generating output at any later date to be discovered if necessary.

For completeness, Datatrack also records information about the environment including

- The operating system
- The version of R
- The loaded packages and their versions

On the dependency graph, this information is all recorded as annotations on a data node, rather than as their own nodes. Again, while this deviates from provenance graphs favored by the OPM specification, it was primarily done in order to keep the dependency graph as visually uncluttered as possible so as to produce a more useable tool.

The decision to keep the processes (the ‘how’ of data provenance) as annotations of data nodes, rather as their own nodes, was made because Datatrack is essentially data-centric: the user only interacts with it through reading and writing data. For this reason, while the processes are part of the workflow and would conceptually fit into the dependency graph (connecting two data nodes), there actually is no requirement as far as Datatrack is concerned to formally name the processes.

3) Generating versions and overwriting data.

Core to the aim of Datatrack is assisting the researcher in selecting version of the input data for a process they are running. Numerous versions suitable data might be available, each having been generated by the same earlier process but with different parameters or using different inputs.

As detailed in section B, Datatrack will write multiple versions of the same name, provided they have a unique combination of dependencies parameters. Version numbers are automatically assigned. If the same script is run with different parameters, this parameter information will be passed on to Datatrack and a new version will be saved. Similarly, if the same process is run with different input data, this dependency information will be passed on when writing the output data and a new version of it will be saved.

If modifications are made to the code that generates the data, or if the process is run in a different environment, such as with different versions of packages loaded, then this does not constitute a new version.

This is done for the simple reason that changes to the code that defines the process is most likely done to improve output results. Essentially, this coupling of a workflow model with a constantly evolving codebase of processes creates this situation.

However, this decision is certainly debatable, and it may be desirable incorporate changes to the source code and loaded packages and into the data versioning system. However, this presents a number of implementation challenges and may complicate the user's interaction with the data dependency graph. At this stage of the prototype, we have decided to keep it simple and not have changes to code generate new versions of data. If the user decides that the two versions of a process *should* result in two versions of the output, they can consider representing this information as a parameter or modifying the name of the output.

4) Potential for illogical data dependencies

Datatrack's intentional simplicity is designed to allow it to fit into a wide variety of coding styles, but means that there is no inbuilt checking that the dependencies declared by the user actually make sense. Cyclic data dependencies could in theory be generated. It is up to the user to select the correct data to use as dependencies and to name the output data in a way that conforms to their conceptual workflow. If done in a sensible way, dependency cycles will be absent.

E. Technical implementation of Datatrack package

As the Datatrack package is a prototype, these implementation details are neither set in stone nor as important as the design considerations and motivations behind the packages, and as such this description has been kept brief.

A *Datatrack project* is simply a directory containing the provenance metadata and the data objects themselves. The minimum configuration for a Datatrack project is the path to this Datatrack project directory. Configuration is stored as a json file in the working directory.

The package records all provenance and versioning metadata in a CSV (comma separated values) file on disk. A relational database would also have been appropriate and possibly more efficient, however it was deemed that running and interfacing with the database was an unnecessary complexity, and would require greater user configuration. By storing all metadata available as a text file, the user can manually inspect it if need be, using familiar spreadsheet programs rather than requiring database management software or SQL knowledge.

Data output files are saved in a central location, the path to which is specified through user configuration, with file names automatically generated by the Datatrack version numbering system.

The visualization of the dependency graph is written in Javascript, html and css using the D3 Javascript library [12]. The "HTML widgets" R package [13] was used to display this

graph within the RStudio console. For maintainability reasons, the visualization tasks were abstracted to a separate R package, on which Datatrack depends.

V. FUTURE WORK

A. Improving interactive features of the dependency graph

The visual interface of the graph used to help the user choose between potential inputs is currently interactive to a limited degree:

- The user can select a date range to limit the number of nodes shown.
- Only the data nodes, grouped by name, are shown in the graph. The metadata that actually informs the decision-making is displayed through user interaction (in the form of a mouse rollover).

While this works well, greater control over what is shown would be better. The key is to offer as much information as possible while continuing to allow easy comparison between many data objects displayed side-by-side.

Currently, the user can filter nodes by name, which will exclude any group that does not have at least one node linked directly or indirectly to a node in the group with the given name. Nodes of the selected group can be filtered by created date. Finer-grained control of which nodes are displayed may be useful if the graph become very large, however the need for this has not yet arisen in our use cases.

B. Integration with Git

Currently, in order to inspect the source code used to generate a data object after the code has been modified, the Git repository can be used. This requires taking the stack trace and timestamp available in the Datatrack metadata and examining the Git repository, but this is not integrated and is done manually. This is both somewhat tedious and unreliable, since there could be multiple branches and information about the branch used for generation of a data object is currently not available in the provenance metadata.

Integration with Git version control will be added to solve this problem. By allowing Datatrack attach the Git repository version number, branch and commit status data object's metadata, accessing this retrospective provenance information will be smoother and less ambiguous.

C. Invalidating downstream data

Let us consider the following scenario. The source code of a particular step of the process is modified, for example to fix a bug. Currently, if this step is run after these changes, Datatrack will overwrite its output data objects (if the dependencies and parameters are the same).

However, this presents a problem if there are any existing downstream data objects that depend on these overwritten data objects. A modification to the code of a step in the workflow has an effect not only on its output data objects, but also on everything downstream to that step.

A proposed solution is as follows. Upon overwriting, first check if the new data is the same as the existing. If it is

different, check if there are any downstream dependent data objects. If so, they should be either deleted or flagged as invalid and archived, after confirming with the user.

VI. CONCLUSION

This paper has discussed some of the provenance tracking and data versioning requirements of research that uses interactive exploratory scripting involving a data workflow, where the nature of the work does not lend itself to integration with a scientific workflow management system.

Although our solution has been developed to cater for the requirements of our specific ecological acoustics research, solving actual problems as they arise, these requirements are reasonably general.

Datatrack addresses these needs by offering a lightweight alternative to using a fully featured scientific workflow management system to researchers whose computational experiments are contained within the single interactive programming environment such as R, and involve constant revisions to the source code.

By automating the generation of provenance metadata in a way that provides almost no modifications to the normal coding practices of the researcher, it succeeds in its goal to be accessible to users from a wide range of computational experimentation practices. Most importantly, it provides a tool to allow the user to select the appropriate input data for their processes quickly and reliably without needing to leave the scripting environment.

REFERENCES

- [1] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, *et al.*, "The open provenance model core specification (v1. 1)," in Future generation computer systems, vol. 27, pp. 743-756, 2011.
- [2] S. Miles, P. Groth, M. Branco, and L. Moreau, "The Requirements of Using Provenance in e-Science Experiments," in Journal of Grid Computing, vol. 5, pp. 1-25, 2007.
- [3] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo, "VisTrails: visualization meets data management," in ACM SIGMOD international conference on Management of data, 2006, pp. 745-747.
- [4] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock, "Kepler: an extensible system for design and execution of scientific workflows," in 16th International Conference on Scientific and Statistical Database Management, 2004, pp. 423-424.
- [5] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-Science: An overview of workflow system features and capabilities," in Future Generation Computer Systems, vol. 25, pp. 528-540, 2009.
- [6] P. Eichinski, "Datatrack," 1.0.0-beta.1
<https://github.com/peichins/datatrack>
<http://dx.doi.org/10.5281/zenodo.60582>
- [7] P. Missier, K. Belhajjame, and J. Cheney, "The W3C PROV family of specifications for modelling provenance metadata," 16th International Conference on Extending Database Technology, Genoa, Italy, 2013.
- [8] K. A. C. S. Ocana, V. Silva, D. d. Oliveira, and M. Mattoso, "Data Analytics in Bioinformatics: Data Science in Practice for Genomics Analysis Workflows," in IEEE 11th International Conference on e-Science, 2015, pp. 322-331.
- [9] L. Murta, V. Braganholo, F. Chirigati, D. Koop, and J. Freire, "noworkflow: Capturing and analyzing provenance of scripts," in Provenance and Annotation of Data and Processes, ed: Springer, 2014, pp. 71-83.
- [10] T. McPhillips, T. Song, T. Kolisnik, S. Aulenbach, K. Belhajjame, K. Bocinsky, *et al.*, "Yesworkflow: A user-oriented, language-independent tool for recovering workflow information from scripts," in CoRR, vol. abs/1502.02403, 2015.
- [11] RStudio-Team, "RStudio: Integrated Development for R," <http://www.rstudio.com/>.
- [12] M. Bostock, V. Ogievetsky, and J. Heer, "D3: Data-Driven Documents," in IEEE Transactions on Visualization and Computer Graphics, vol. 17, pp. 2301-2309, 2011.
- [13] R. Vaidyanathan, Y. Xie, J. Allaire, J. Cheng, and K. Russell, "htmlwidgets: HTML Widgets for R," R package version 0.6
<https://cran.r-project.org/package=htmlwidgets>

Starting Workflow Tasks Before They're Ready

Wladislaw Gusew and Björn Scheuermann

Computer Engineering Group

Humboldt University of Berlin

{gusewwly, scheuermann}@informatik.hu-berlin.de

Abstract—Today's science is more and more driven by collecting and evaluating increasing amounts of data. Utilizing Scientific Workflows is one suitable method how to organize processing pipelines for this purpose. In this work, we show that performance improvements on the execution of existing workflows can be achieved, if the conditions for starting selected tasks with certain data access characteristics are loosened. We provide a scheme how to identify eligible tasks in a given workflow and demonstrate a technique how an earlier start of tasks can be realized in Pegasus WMS by transforming the workflow DAG and by using a wrapper around the task executable during runtime. Our implemented wrapper handles the reading data accesses for task instances so that existing original workflows can be executed without the need to modify them. We evaluate our approach in simulations and experiments on real distributed computing resources, and are able to observe performance improvements for the Montage workflow by a significant reduction of total execution time.

I. INTRODUCTION

Research in scientific fields, like, for instance, natural sciences, medicine, computer science, or engineering, is often conducted by evaluating large quantities of measured and collected data [1]–[5]. In order to produce experimental results, raw data has to be processed and transformed in several steps, each performing computations regarding specific aspects. The composition of all processing steps of such a workflow forms a *directed acyclic graph* (DAG) where nodes correspond to tasks and edges define the data dependencies between these tasks. The incoming edges into a task are input data objects produced by the *parent* and, similarly, outgoing edges represent generated output by the task that is consumed by the *child*. In order to achieve acceptable total runtimes of data-intensive workflows, parallelism is exploited by utilizing distributed computing resources, such as, for example, grids, computing clusters, or clouds [1], [6]. The magnitude of data transferred between the tasks is steady-increasing in practice so that the network connection between the computing nodes typically constitutes the performance bottleneck [7], [8] during execution.

For distributed execution, a mapping from workflow tasks to available physical resources has to be established such that certain goals, e.g., short runtime, high utilization of resources, or low monetary costs, are met. Common practical scheduler implementations regard a workflow task as ready to run not before all specified input objects have been completely produced and become available [1], [9]–[13]. Waiting until all input data is completely produced can impose a non-negligible constraint for task implementations that do not require all

input for starting data processing. Examples for such tasks are implementations which require two or more input objects and where the inputs are completely read and analyzed one after another, e.g., in order to check the inputs for certain criteria, or in order to concatenate all inputs to one output in a reduce-pattern manner [8]. For such tasks, an earlier start would be possible so that in a first part of progress only a subset of input data objects is processed, whereas the subsequently required remaining input data can be delivered after the moment the task execution has begun. In other words, a given task implementation can be split into multiple virtual tasks that can be scheduled by unmodified existing scheduling algorithms so that an earlier start of the first virtual parts of the task in the workflow can be achieved. Consequently, by supporting this differentiation schedules can be generated that result in shorter workflow runtimes.

In this work we introduce a scheme for automatically investigating eligible tasks for virtual splitting by analyzing their performance and processing characteristics. These tasks are then appropriately split into a group of virtual tasks and the DAG of the scientific workflow is correspondingly transformed. The virtual split of each task is realized by utilizing our developed wrapper, which starts the original and unmodified task executables on the allocated computing node and handles all system calls relating to file open operations. Thereby, the degree of parallelism is not changed because a task which is virtually split into multiple tasks forms a group that is allocated as a whole to a single computing node.

Our main contributions are threefold: (1) we apply a program profiling technique to analyze the accessed input data by the task without relying on additional annotated information, (2) we develop a lightweight wrapper around task implementations in order to handle accesses of tasks to input data objects, and (3) with the profiling information we transform the original DAG, representing the initial abstract workflow, into a refined DAG that can then be used as an input to already existing execution engines.

We evaluated our proposed scheme with the Montage workflow [14] with the simulation framework *WorkflowSim* [12] as well as in a real-world computing cluster with Pegasus WMS [9]. Montage is a collection of tools for processing astronomical images. A composition of these tools, as depicted in Figure 1, is often used for performance evaluations of workflow engines in literature. Our observations show that the total workflow runtimes can be decreased, for instance, by more than 15% for a workflow with 133 tasks. Simulation

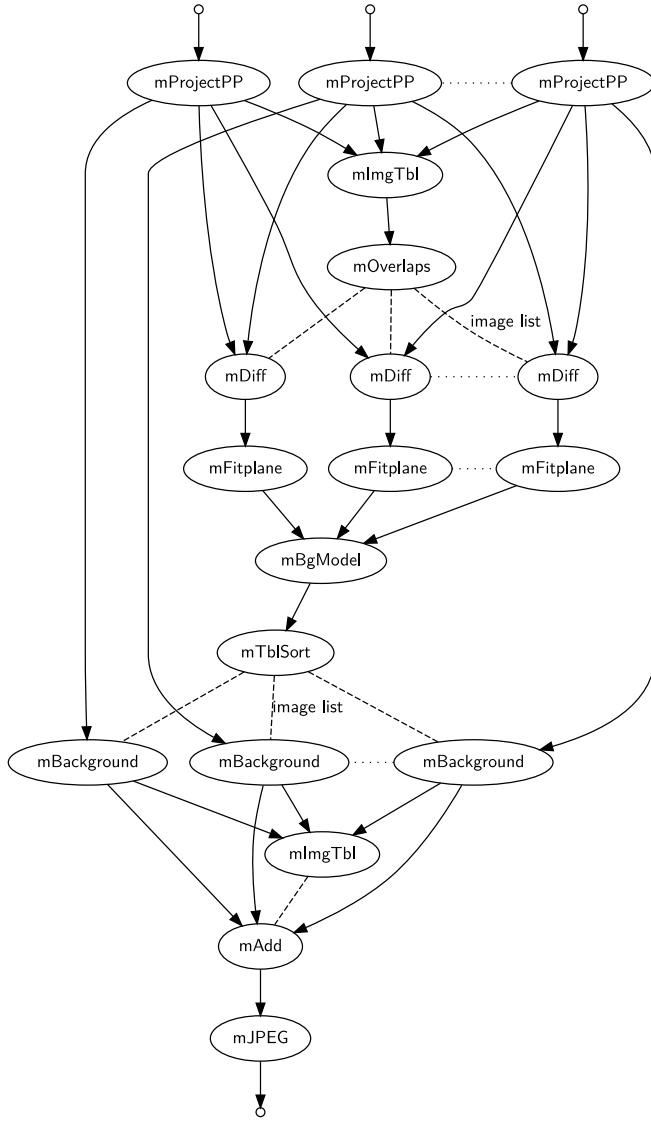


Fig. 1. An example instance of the Montage workflow with 18 tasks. The dotted lines mark the level where additional tasks appear, if the size of the workflow is scaled up. The dashed lines represent the transfer of lists as relatively small data objects which influence the number of subsequent tasks to schedule.

furthermore indicates that the performance gain increases with increasing number of computing nodes and tasks.

The remainder of this paper is structured as follows: starting from a review of related work in Section II, background information on the execution model of scientific workflows is given in Section III where we outline the idea behind our DAG transformation approach. We introduce our methodology for profiling workflow tasks in Section IV and our employed technique for splitting tasks into virtual tasks in Section V. These approaches are used in combination and applied to the Montage workflow in Section VI in order to quantitatively evaluate the decrease of total runtime in a simulation and an experiment, before we conclude this paper in Section VII.

II. RELATED WORK

Related existing methods such as, for example, pipelining or data streaming [15]–[17] show similar effects to our approach, because tasks can then likewise process input earlier. However, it is typically required that the execution engine supports these methods. In addition, the data model of the individual tasks must facilitate a partitioning of the data objects in smaller chunks, a prerequisite that our task split technique does not depend on. Another related approach proposed by Chen et al. [13] is task clustering. By merging multiple tasks into a single job – that is an atomic execution unit for the workflow execution engine – staging overheads for individual tasks can be diminished, because the total number of executed jobs is decreased. In contrast, our proposed technique does not change the degree of parallelism, but still can lead to lower total workflow runtimes because of reduced delays between executed tasks.

We structure our further review of related work into three categories: monitoring and profiling of workflow task executions, conducting workflow DAG transformations, and characterizations of common workflow patterns.

A. Profiling

Juve et al. demonstrate in [18]–[20] that monitoring and profiling real scientific workflows can produce valuable statistical data for deriving estimates of a task’s resource utilization or for detecting inefficient program implementations.

In [18], six different real scientific workflows were profiled under consideration of the runtime, I/O usage, peak memory usage, and CPU utilization of each type of task for each of the workflows. Based on the captured data, a characterization into compute-intensive or data-intensive workflows with low or high resource requirements has been conducted. For the Montage workflow, Juve et al. state that the *mAdd* tasks have been the I/O operationally most intensive tasks in the experiments and spend the majority of the total workflow runtime. Based on this observation, in this work we focus on the *mAdd* task and by starting the execution of this task earlier, we achieve a significant overall performance improvement, as the runtime of the *mAdd* task dominates the total runtime of the Montage workflow.

In [19], [20], the Kickstart tool [21] that can launch tasks and report information about the execution and the environment has been used for monitoring the behavior of the tasks. During a task’s execution, Kickstart periodically queries specific tools, i.e., *procfs*, *stat*, or *ptrace*, that utilize operating system mechanisms in order to respond with fine-grained information on, for instance, CPU and I/O usage, runtime, and peak memory usage. As the overhead increases with higher granularity of the responded data, due to more context switches when, for example, system calls are interposed, Kickstart provides the user with three different monitoring levels so that an acceptable impact on the performance can be configured. The main difference to our approach is that Kickstart is intended to be run during a production execution in the background of the actual workflow run for all of

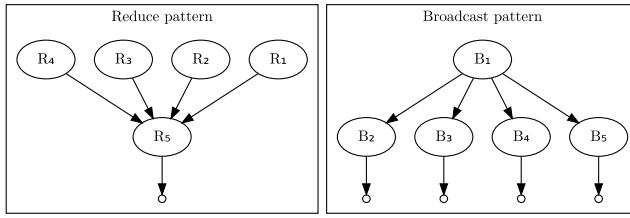


Fig. 2. Examples of the reduce pattern and the broadcast pattern.

the workflow tasks, which means that the overhead has a measurable impact on the total runtime of the workflow. In contrast, we propose a scheme for profiling each type of a workflow task prior to a production execution in order to obtain the task's I/O read and write characteristics which then can be used for the DAG transformation into a task group with virtual sub-tasks.

B. DAG Transformation

The workflow management system Pegasus [9] is designed in such a way that transformations of an abstract workflow, which is independent of the available computing resources, are conducted in a sequence of refinements in order to obtain an executable workflow. This transformations include the detection and prevention of redundant task executions in cases where a required data object has already been produced and is still available in the storage system. Moreover, the Pegasus system is able to cluster the workflow tasks that are assessed as relatively short running tasks with a high granularity. These clustered tasks are composed into a single virtual task which is required to be executed on the same computing node. Task clustering techniques can increase the overall execution performance of the workflow [9], [13], [22], because staging overhead for many short running tasks can be reduced to the overhead for just the encapsulated virtual task.

In contrast to task clustering where only independent tasks are considered, we focus on tasks that depend on input data produced by different parent tasks which is processed with certain access characteristics. Another difference is that we propose to transform the DAG in so far as we split a single task into multiple virtual tasks, instead of merging multiple tasks into a single virtual task.

C. Workflow Patterns

Within the domain of scientific workflows many studies were conducted with an analysis of data access patterns in existing workflows. Costa et al. present an overview of common patterns distilled from five different research articles in [8]. In our proposed scheme, we are mainly interested in the reduce pattern¹ where a single task processes multiple input data objects that have been previously produced by direct parent tasks, as is shown in Figure 2. Beside the reduce pattern, a contrary broadcast pattern exists where one single

¹In literature, *merge* or *aggregation* pattern are synonyms for the reduce pattern.

task produces a set of output data objects for multiple children tasks, as well as the scatter, gather, reuse, and pipeline patterns. While [23] and [8] evaluate potential gains in performance by using an optimized data placement algorithm, we evaluate performance improvements by achieving an earlier start of a workflow task by using partially available input data, in comparison to the policy which requires all input data objects in order to start the task.

III. EXECUTION SEMANTICS OF SCIENTIFIC WORKFLOWS

A scientific workflow is composed of one or multiple tasks which can have data dependencies between each other as expressed by edges in the corresponding DAG. Several workflow execution engines, including Pegasus, consider a data object to be completely produced and retrievable for the child only after the parent has finished execution. This means technically that the main process of the parent task program has returned. Then, the data can be transferred to the location of the child and after the transfers of all required data objects have succeeded, the child program is regarded as ready to start. These semantics have proven to work for many practical workflows as can be noticed by the wide use of workflows in the scientific community, but this scheme simplifies actual operations in two aspects. In the following, we explain both aspects and motivate that by considering them, tasks can be regarded as ready earlier so that performance optimization potential can be provided.

First, during the run of a task it produces its output according to its algorithmic implementation. Especially for tasks with multiple output data objects, a subset of the output may be completely produced significantly earlier than the moment of the task's termination. In these cases, child tasks which need data only from this available subset could be started before the parent is done. In order to utilize this potential, execution engines must be provided with information on the task's data output characteristics for the planning and scheduling of tasks on the available resources. Typically, this information is not annotated to the workflow so that tasks can only be regarded as black boxes. Our proposed profiling technique, as described in Section IV, provides a way for obtaining the data output characteristics.

While the first aspect is concentrated on the output side of parent tasks, the second aspect highlights the input side of children tasks. In Figure 3a, a small example workflow with two parent tasks T_1 and T_2 , and a child T_3 that can be regarded as a reduce task, is depicted. With common execution semantics, the start of T_3 is scheduled after T_1 and T_2 have finished and all data D_1 and D_2 has been transferred to the location where T_3 will be executed. In a scenario with two computing nodes, which we define as *workers*, a possible schedule is shown in Figure 3d, where first, worker W_1 has to execute T_1 in parallel to the execution of T_2 at W_2 . Then, at the site of W_1 task T_3 is scheduled, which requires D_1 as well as D_2 , before T_3 is started according to the common semantic. To this end, D_2 is transferred to W_1 , which takes a certain amount of time, depending on D_2 's size and the achievable

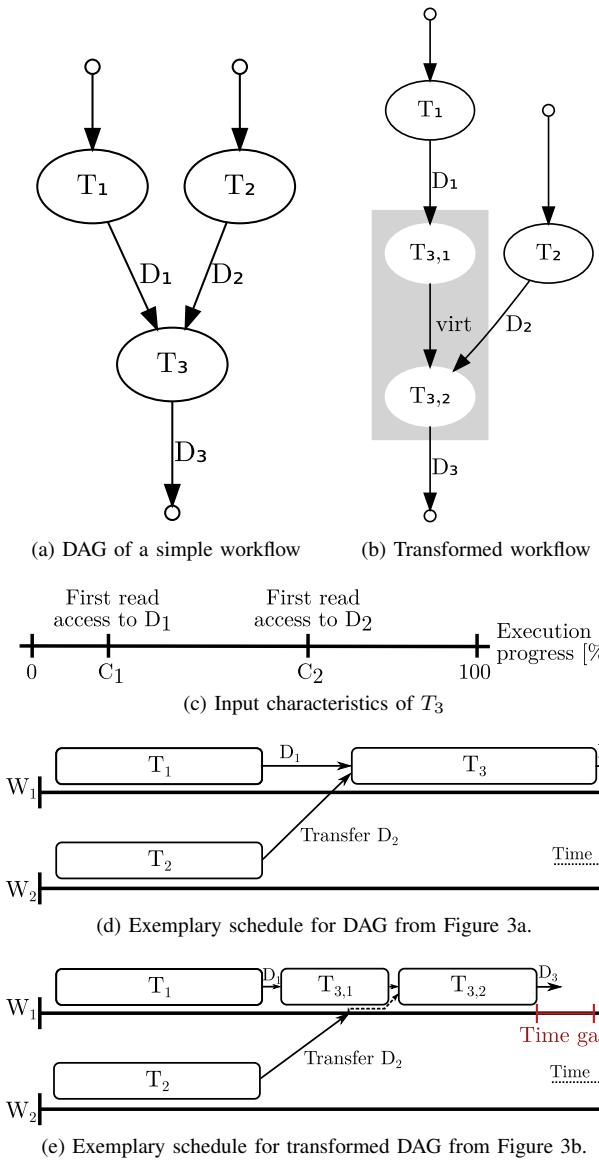


Fig. 3. Virtually splitting the qualified task T_3 at the point in time when D_2 is accessed for the first time. This leads to an earlier start of T_3 and a reduced total runtime of the small workflow example.

data rate on the communication link between W_2 and W_1 at that moment. Finally, T_3 is started and processes both input data objects in order to come up with the final result.

To demonstrate the potential we want to exploit with our approach, a diagram in Figure 3c represents an exemplary T_3 's data input characteristic by relating the first access to each input data object to the execution progress of T_3 . After T_3 is started at 0%, its first access to D_1 is at the progress state C_1 and the task can process D_1 for a while, before it performs the first read access to D_2 at C_2 . Then, between C_2 and C_3 both input data objects are maybe read concurrently until the task is finished at 100%.

In this example case, our scheduler could start T_3 immediately after T_1 has finished, if the data transfer of D_2 to worker W_1 will be completed until the processing reaches C_2 . Based

on the example input characteristics of T_3 , we can transform the DAG from Figure 3a to the DAG depicted in Figure 3b by virtually splitting T_3 into two separate tasks $T_{3,1}$ and $T_{3,2}$ arranged in a sequence. Semantically, this means that $T_{3,1}$ represents the normal beginning of T_3 up until the moment when D_2 is accessed for the first time, which is symbolized by the edge *virt* and start of $T_{3,2}$. A possible schedule of this refined workflow is presented in Figure 3e where the start of $T_{3,1}$ is scheduled before the data transfer of D_2 has finished. In this case, a time gain can be achieved so that the total runtime of the workflow is reduced.

With the presented approach, modifications to the task implementations of the workflow or the execution engine are not required. In order to interrupt the execution of T_3 at the point in execution progress when D_2 is accessed for the first time and later continue T_3 's execution, we employ a wrapper program that controls the task's execution. However, a main prerequisite for a reasonable application of this scheme are applicative access characteristics to the input data objects by the reduce task, as is, for instance, shown in Figure 3c. In the following Section IV, we describe our profiling technique which helps to identify similar characteristics for eligible tasks of the Montage workflow.

IV. PROFILING WORKFLOW TASKS

Scientific workflows are typically composed of tasks which provide different functionalities so that the tasks can be distinguished by their runtime properties, such as, for example, peak memory requirements, CPU utilization, and I/O access statistics [18]. For instance, the Montage workflow we employ in this work consists of ten essential task types, as depicted in Figure 1, where tasks like, e.g., mProjectPP, are computationally intensive, whereas tasks like, e.g., mAdd, are dominated by high I/O loads [18].

A. Profiling Mechanisms

For our runtime optimization approach, the goal is to detect tasks where the read accesses of the task resemble a certain pattern. We define a task as eligible for our DAG transformation, if the task needs to access multiple input files (e.g., file A and file B), and if the read accesses occur consecutively with a significant delay between the first read access to one file (i.e., file A) and the first read access to the next file (i.e., file B).

For detection of eligible tasks, we prefer to utilize a profiling technique based on the tool SystemTap [24] over a manual source code review, as source code of workflow tasks is not always available, and because a manual review needs an extensive effort and becomes error-prone for complex task implementations. In contrast, with SystemTap we can characterize file access behavior of an arbitrary task in an automated process. SystemTap provides operating system instrumentation under Linux and enables intercepting system calls for I/O accesses through the POSIX API. In order to record the trace of read and write accesses to input and output files by an arbitrary task, we developed a script for SystemTap that

stores for each accessed file two counter values: a counter for the number of bytes read so far, and another counter for the number of bytes written to that file, respectively. During execution of the task, these counter variables are continually updated on each I/O access to the corresponding file, and in periodic intervals stored in a log that is afterwards used for the preparation of the profiling results.

With this method, it is possible to track down the point in execution progress when a file is accessed for the first time (i.e., when the corresponding counter value changes to a value greater zero). The execution progress is measured in consumed CPU cycles, an information that SystemTap can provide on many system architectures. We intentionally decided to utilize CPU cycles instead of wall clock time, as in general preemptive schedulers of operating systems can swap a running process of a task out of the CPU, if it is the turn of another concurrent process running on the system. Therefore, the relation could become distorted so that the intervals between data points next to each other would be extended. Hence, we record the CPU cycles consumed only by the task process for computations and processing.

B. Detection of Eligible Tasks

As motivated in Section III, for our optimization approach we primarily consider tasks as eligible which are arranged as reduce tasks in the workflow. However, it is not possible to detect eligible tasks solely from the workflow DAG, as the task's read access characteristics have to match the criteria that we defined above. For this reason, we depict a selection of obtained tasks' file access characteristics in Figure 4.

These diagrams show only I/O operations on the most interesting files in order to provide a better overview. Read and write accesses to relatively small files like, e.g., files with the extensions `*.hdr` or `*.tbl`, as well as all `*_area.fits` files are ignored here. The workflow configuration that we used for the profiling corresponds to the workflow that will be generated by the calls

```
$ mHdr m31 <D> m31.hdr
$ mExec -o m31.fits -f m31.hdr 2MASS J tmp
```

where the degree parameter `<D>` influences the size of the mosaic image and therefore the complexity and total number of tasks in the workflow.

In Figure 4a, it is shown that the re-projection task `mProjectPP` first performs a complete read of the main input file in the Flexible Image Transport System (FITS) format and close to the termination it continually writes the plane-to-plane transformation results into the output file. Between both I/O operation blocks, a relatively long processing interval can be observed where the task performs computations on the input data. This observation supports the findings of Juve et. al in [18] that the `mProjectPP` task is computationally intensive.

In contrast, the characteristics of `mDiff` and `mBackground`, as presented in Figure 4b and Figure 4c, appear to be dominated by I/O operations than by computations, because no recognizably long periods of

unchanged I/O access states are observable. Moreover, we observe that `mDiff`'s characteristics meet our criterion for our optimization approach, because the first read access to the second file occurs significantly later than the first read of the first file. Thus, a split of each `mDiff` task in two consecutive virtual tasks, as described with a similar example in Section III, may potentially increase the execution performance of the workflow.

Instead, we decide to evaluate our approach by splitting `mAdd` tasks. The reason for this decision becomes clear, if we consider the characteristics of `mAdd` tasks in cases when the workflow size increases, as is shown in the diagrams Figure 4d to Figure 4f. While the `mDiff` task always expects two different input files and therefore can only be reasonably split into two virtual tasks, the degree to which a `mAdd` task can be split increases with an increasing workflow complexity. This can be quantitatively expressed by considering the number of different files read in each of the portrayed situations: while `mAdd` accesses only six files in Figure 4d, in Figure 4e there are already 30 different files, and this number is more than tripled for Figure 4f to 106 files. Although, many of the files are read in parallel, still intervals for our task criterion can be recognized even in Figure 4f, as the accesses follow a repetitive pattern. We conclude from this analysis of tasks' characteristics that the `mAdd` task meets our formulated criteria and is an eligible task for the optimization approach.

C. Profiler Overhead

While related approaches of runtime optimization exist where the profiling needs to be activated during all production runs, our proposed approach requires that the tasks' characteristics are obtained once for a certain workflow configuration. By analyzing the characteristics, decisions can be made for which of the tasks the virtual split should be applied. Therefore, the overhead by the profiling mechanism occurs only once during the pre-run analysis, whereas for other profiling methods the overhead would be present in each production run. Moreover, an already obtained task characteristic could be annotated to the task in the workflow description so that for such an enriched workflow the profiling stage could be skipped and the workflow executor could directly decide whether DAG transformation for an earlier start of eligible tasks should be applied. As the obtained characteristics depend on the concrete input files of the tasks, it is not possible to directly translate the behavior on another workflow configuration and different files used. But, as we see in Figure 4d to Figure 4f, tasks are often implemented in a way where some structural patterns concerning the first accesses to files exist, and where a very similar structure is observable, if the number of files as well as the file sizes remain approximately equal.

V. SPLITTING TASKS INTO VIRTUAL TASKS

The previous two sections discussed the idea behind our DAG transformation approach and to which workflow tasks such a transformation can be reasonably applied. This section describes the technological measures for realizing the split

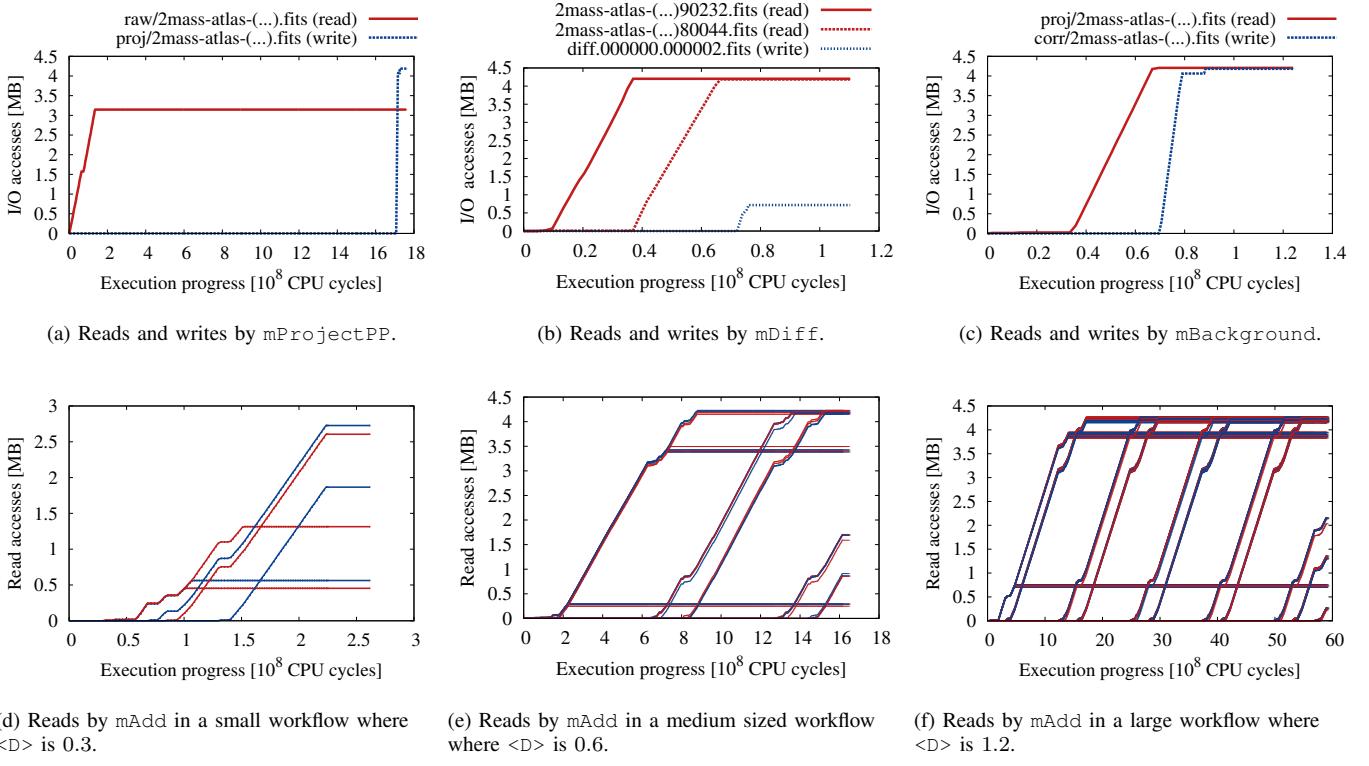


Fig. 4. Recorded traces of I/O operations obtained in exemplary executions of four selected task types. Each line in the `mAdd` diagrams represents read accesses performed to separate files.

of a single task into multiple virtual tasks, which is needed in order to execute the transformed DAG of an optimized workflow on real systems. A core feature of our approach is that modifications to the source code of an existing task implementation are not conducted, instead the task is executed inside a wrapper that handles read accesses to files in the file system.

In order to achieve our goal of a transparent wrapper (from the point of view of the executed task), we utilized the Filesystem in USerSpace (FUSE) interface [25] that enables the setup of a virtual file system without changing kernel modules or code, because the FUSE kernel module is already integrated in many current Unix-based operating systems, such as, e.g., Linux, FreeBSD, or OS X. In this way, during the DAG transformation stage the definition of the executable program to run for the transformed task is replaced by the wrapper program, which receives the task executable as its input argument, and in addition the relative path to the directory where the virtual file system should be located. When the first task of the sequence of virtual tasks is started, the wrapper instantiates a process at the computing node that contains FUSE-specific function callbacks, which are evaluated if the virtual file system is accessed by other processes through the operating system. For instance, the file open operation on a file located inside the virtual directory will lead to an evaluation of the specific callback for opening files. Our implementation considers two different cases: (i) a file needs to be opened,

which is already physically available at the computing node, then the task that accessed that file can proceed, or (ii) the task instance accessed a file which is not available yet, this is equivalent to a transition from one virtual task to the next. In the second case, the execution of the task process is stopped and only continued, when the wrapper of the next virtual task is started, as this means that the priorly accessed file has been produced and transferred, and must be already present at the worker. While from the view of the execution engine, it appears at this transition point that another task has been started at the worker, the wrapped task instance is still active and just changes from the stopped process state back to the running state.

One current limitation of this approach is that all virtual tasks originating from the same real task must be regarded as clustered so that they have to be executed at the same computing node. The configuration option for clustered task execution is inherently provided in several execution engines, in particular in Pegasus WMS with label-based job clustering [26]. Nevertheless, a distribution of virtual task instances to multiple workers may be technically feasible, if specific techniques for the transfer of the whole task execution states at transition points from one virtual task to the next are implemented in the execution engine. The distribution over multiple workers could be reasonable in scenarios where measures are taken against failures of very long running task instances, which have been virtually split according to our

scheme in order to be able to start these tasks earlier, and additionally, the whole cluster of virtual tasks is executed in an over-provisioned way on multiple parallel instances. Thus, if one instance fails due to a system crash, another worker can continue at the last consistent checkpoint by loading the saved state of a finished virtual task and proceed with the next virtual task.

Although we described that during production runs there is no overhead by the profiling mechanism, there is some overhead for the wrapped tasks, because (i) the wrapper needs to handle the interprocess communication with the virtualized task, and (ii) the virtual file system can be regarded as an additional intermediate abstraction layer that has to perform additional context switches of the processes. We quantitatively analyzed this overhead imposed by the wrapper program in comparison to task executions without the wrapper and observed that it never exceeded 3% of the total runtime in our experimental setup that we introduce in Section VI-B.

VI. EVALUATION

We evaluate the proposed task splitting scheme in two stages: first, we utilize the WorkflowSim framework in order to simulate the behavior of the executed Montage workflow under many different configurations. Then, we run two instances of the Montage workflow on a real computing cluster with up to ten distributed computing nodes. In both experiments, we analyze the total runtime of the workflow that is defined as the time interval between the start of the first task and the termination of the last task. Our experimentation goal is to investigate how a split of the mAdd aggregation task into multiple virtual tasks affects the total runtime of the workflow.

A. Simulation with WorkflowSim

The WorkflowSim framework [12] is a simulator for the execution of synthetic workflows in a simulated distributed system. It supports several scheduling algorithms, such as, e.g., min-min, max-min, or HEFT [27], and employs a layered model in order to map different overheads that have an impact on the performance in reality, to the simulation. The WorkflowSim project includes a set of example workflows stored in the XML-based DAX format, such as, for example, four different versions of the Montage workflow that we have used for our evaluation. We refer to these synthetic Montage workflows with M_N where N can take the values $\{25, 50, 100, 1000\}$ and represents the total numbers of tasks in each workflow. Furthermore, we refer to the, accordingly to our proposed scheme, transformed workflow with a virtually split mAdd task with M_N^* .

Before running our simulations, we have to prepare the workflow descriptions in the DAX files in order to conduct our evaluation under comparable conditions. To this end, the simulation framework has to be provided with additional information on the accessed files by each of the virtual mAdd tasks. In the original workflow, this information is implicitly encoded in the dependencies of the mImgTbl task (preceding the mAdd task) and in its annotated runtime as well as the

TABLE I
CONFIGURATION OF THE SIMULATION ENVIRONMENT

Simulated scheduling / planning algorithms	
{ Min-Min, Max-Min, Round-robin, HEFT, DHEFT, Random }	
Clustering	disabled
Failure generator	disabled
Datacenter VM config.	
Type	CondorVM
Storage size / RAM	10 GB / 512 MB
Network bandwidth	10 Mbit/s
File system	local
#CPUs per VM / MIPS	1 / 1000
#VMs	{5, 10, 50, 100}
#Workflow tasks	{25, 50, 100, 1000}

runtime of the mAdd task. We solve this issue with the following modifications to the M_N^* structure:

- 1) Set *runtimes* to the sum of the annotated task runtimes of mImgTbl and mAdd.
- 2) Assign the list *infiles* with all input file dependencies of mImgTbl and assign the list *outfiles* with all output file dependencies of mAdd.
- 3) Set the number of virtual mAdd tasks *#virt* to the number of mBackground tasks plus one.
- 4) Set *runtime_i* to the quotient of *runtimes* and *#virt*.
- 5) Assign to each virtual mAdd task the value of *runtime_i* and add some additional five percent overhead time of *runtime_i* in order to approximately represent the additional overhead when starting our wrapper implementation.
- 6) Split up *infiles* in consecutive two-tuples and divide in half the annotated file sizes of each entry. With this step, we build pairs of the "area" file together with the corresponding normal FITS file.
- 7) Iterate over all resulting tuples and assign each tuple to two consecutive virtual mAdd tasks. Thus, we assume for the simulation that the read accesses of the different file pairs are overlapping at the half of the files when the transition occurs from one virtual task to the next.
- 8) Divide the file sizes of the entries in *outfiles* by *#virt* and assign these output file dependencies to each of the virtual mAdd tasks. Here, we assume that the output file is written continually during the whole execution of the virtualized mAdd task.
- 9) Remove task mImgTbl and mAdd's input file dependency to cimages.tbl, which can safely be done as we have distributed all of mImgTbl's semantic contents over the virtual mAdd tasks.

WorkflowSim enables the configuration of a simulated execution environment by a set of parameters that we list for our experiment in Table I. In order to focus on the impact of workflow transformation on execution performance, we have disabled clustering of other workflow tasks and generation of execution failures. As scientific workflows are typically considered to be data-intensive, which means that data objects with high volumes are transferred between the nodes, we want to simulate a similar scenario. To this end, we could increase

TABLE II
REDUCTION OF TOTAL WORKFLOW RUNTIME WITH SPLIT mAdd TASK COMPARED TO ORIGINAL WORKFLOW

#VMs	#Tasks	Scheduling and planning algorithms					
		Min-Min	Max-Min	Round-robin	HEFT	DHEFT	Random
5	25	10.53%	14.96%	12.88%	12.74%	11.08%	40.52%
5	50	10.40%	13.11%	15.52%	-22.81%	15.20%	39.89%
5	100	10.12%	11.14%	12.05%	8.74%	13.41%	12.60%
5	1000	11.05%	10.33%	10.35%	7.25%	7.49%	10.88%
10	25	14.54%	14.53%	15.66%	11.09%	11.33%	7.69%
10	50	14.70%	18.85%	14.77%	12.05%	13.32%	4.77%
10	100	14.46%	17.17%	21.18%	10.25%	19.46%	11.27%
10	1000	16.95%	16.38%	16.05%	8.57%	10.51%	-0.09%
50	25	14.54%	14.53%	15.66%	11.09%	11.33%	20.56%
50	50	16.10%	18.95%	20.01%	16.28%	13.86%	0.04%
50	100	25.94%	24.61%	25.16%	25.19%	16.74%	-0.80%
50	1000	31.08%	30.72%	31.06%	29.92%	15.08%	20.73%
100	25	14.54%	14.53%	15.66%	11.09%	11.33%	-8.37%
100	50	16.10%	18.95%	20.01%	16.28%	13.86%	1.38%
100	100	24.81%	26.60%	27.57%	25.08%	16.74%	33.91%
100	1000	34.22%	33.54%	33.66%	33.05%	18.84%	4.74%

the file sizes of the exchanged files between the workflow tasks, as the example Montage workflows process relatively small sized files, or alternatively, we can reduce the network bandwidth – as practiced here – in order to cause contention for the network communication so that the network connection becomes the performance bottleneck. Therefore, we throttle the network connection for our simulation to 10 Mbit/s so that we achieve a more realistic situation, similar to scientific workflows that move big amounts of data between the tasks on high-speed connections. Consequently, data transfer between the workers (i.e., VMs) has a crucial impact on the overall execution performance. For evaluation, we vary three dimensions: the number of workers, the number of tasks, and several scheduling algorithms, whose functionality is orthogonal to our proposed workflow transformation scheme.

Our results are presented in Table II where the values are percentages by which the total runtime of the workflow execution has been reduced by M_N^* in comparison to the original workflows M_N . Negative percentage values indicate configurations where execution of M_N^* performs worse than M_N which is, for instance, the case for (5, 50, HEFT).

In general, for simple scheduling algorithms, i.e., min-min, max-min, and round-robin, with our approach we are able to achieve performance gains by approximately between 10% to 30% in the simulation. The tendency towards better performance is observable for increasing numbers of workers, and for constant numbers of nodes the total runtime is reduced sharper for an increasing number of workflow tasks. For instance, from (5, 50, min-min) to (50, 50, min-min) the performance increases from 10.40% to 16.10% by 5.7 percentage points, whereas it increases, for example, from (50, 50, max-min) and 18.95% to (50, 1000, max-min) and 30.72% by 11.77 percentage points. If we compare the values for the same workflow size over different numbers of nodes, we can observe a saturation in runtime reduction. For example, there is an increase from (5, 25, round-robin) and 12.88% to (10, 25, round-robin) and 15.66% by 2.78 percentage points, but if we further increase the number of nodes to, e.g., (50,

25, round-robin) we still achieve a reduction of the runtime by 15.66%. This effect can be observed for the min-min and max-min scheduling due to arriving at a situation where a higher number of workers can not further be exploited by the execution engine, as not enough workflow tasks can be set in the running state at the same time, because of their reciprocal dependencies.

Compared to the simple scheduling algorithms, there is a mixed picture for the HEFT and DHEFT results. In the majority of the results, a beneficial impact of M_N^* can be seen, but for (5, 50, HEFT) the performance is reduced and the total runtime becomes 22.81% longer than for the original workflow. The higher complexity behind HEFT and DHEFT seems to distort the trends that we can observe for the other three previously mentioned algorithms. Particularly noticeable are the values for (5, 100, HEFT/DHEFT) in contrast to (5, 1000, HEFT/DHEFT) where the performance gains relatively decrease while the number of tasks in the workflow becomes ten times higher. The same situation that can be observed for (10, 100, HEFT/DHEFT) and (10, 1000, HEFT/DHEFT) suggests that the behavior of these two algorithms with our task splitting approach is difficult to predict. Furthermore, the relative performance gains for HEFT and DHEFT are in general inferior to the gains achieved with the more simple algorithms. Interestingly, the absolute numbers of the runtimes in this simulation (which are omitted here due to space limitations) show that schedules generated by HEFT or DHEFT lead to longer runtimes for nearly all configurations than the ones generated by simpler algorithms.

The right-most column in Table II contains the results for the trivial *random* scheduling algorithm that assigns each runnable task randomly to one of the available workers. Surprisingly, in the majority of simulated configurations, our task splitting approach leads to performance gains up to 40% for this unpredictable scheduler. This observation may indicate that by splitting the mAdd task the probability of randomly choosing a poor schedule is reduced.

Overall, simulation results show that by achieving an earlier

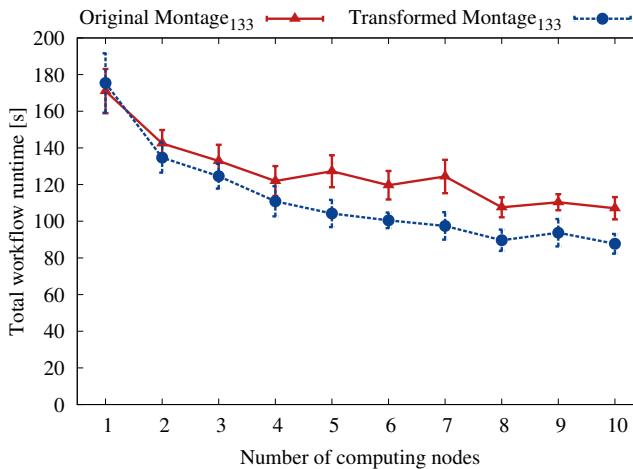


Fig. 5. Workflow execution time on a computing cluster with up to ten nodes.

start of the aggregation task due to its virtualization and a transformation of the workflow DAG, a performance increase of up to approximately 30% can be obtained.

B. Execution on a Computing Cluster

In order to evaluate our scheme in a realistic setting, we executed the Montage workflow containing 133 tasks in the Pegasus WMS on a small cluster with ten computing nodes. Each node is running the Debian GNU/Linux 7.0 operating system on a machine with an Intel Core i7 CPU providing a clock rate of 2.5 GHz and 4 cores, 8 GB of main memory. All nodes are connected to a network switch with a bandwidth of 1 Gbit/s. According to our experimentation goal, we throttle the network bandwidth for all workers on the application layer to 10 Mbit/s, as explained above in Section VI-A. For file transfers between workers, we configured Pegasus to use condorio with a local storage at each computing node. The cluster was used exclusively for our experiment so that negative side effects by other users on the nodes or network activity could be confined to a minimum.

During the experiment we incremented the number of attached computing nodes with one active HTCondor worker instance on each, from one to up to ten. Similar to the simulation in Section VI-A, two variants of the Montage workflow were created: one with the original workflow, and the second with the conducted virtualization of the `mAdd` task. Each configuration was repeated ten times in order to quantify the impact of statistical fluctuations and the arithmetic average as well as the standard deviation were calculated. The measured results of the total workflow runtimes are depicted in Figure 5 including error bars with 95% confidence intervals. We show here the numbers with utilizing the min-min scheduling algorithm in Pegasus, however, the results for the max-min algorithm were not significantly different.

The graph shows that the transformed workflow outperforms the original workflow in all configurations, except when only one computing node is utilized. This can be explained by the

additional overhead which our specific task wrapper provides, as discussed in Section V. In this case, executing the transformed workflow takes four seconds longer which is a relative performance loss of approximately 2%. When the number of workers is increased, both workflow executions can benefit from the increased parallelization potential. While the runtimes decrease for both up to four available workers with a similar gradient, from five computing nodes on the execution times of the transformed workflow drop sharper so that the gap between the curves increases. In the range from eight to ten nodes the curves seem to approach a saturation zone, where more available parallel nodes do not benefit that much, because the workflow structure does not provide any more parallelization potential.

These observable results underline our findings in the simulation that the transformation of a workflow in order to start aggregation tasks earlier can have a measurable impact on the execution performance. If we roughly compare the relative reduction of execution time for five computing nodes in the real experiment, i.e., approximately 18%, with the simulation results where we could achieve for 5 workers and *Montage*₁₀₀ around 10% to 12% with the simple scheduling algorithms, then we see that this simulation result comes close to reality. An explanation for the few percent more may be the different size and structure of the workflows, as the simulation utilized a synthetic workflow with 100 tasks, while the cluster experiment worked with a real workflow with 133 tasks.

VII. CONCLUSION

We presented our scheme for splitting a workflow task into multiple virtual tasks in order to be able to start the virtualized task earlier without the need to modify the implementation of the task or the execution engine. To this end, we utilized three different techniques that we describe in this work in combination, which are first, profiling black-boxed workflow tasks and infer from the profile the eligibility of the task for an earlier start, second, transform the workflow DAG by refining the virtualized task as a cluster of tasks which represent calls to our specific wrapper, and third, a wrapper program which creates locally a virtual file system that provides the interface between the wrapper and the actual task executable. By using inter-process communication methods, the running state of the virtualized task process is controlled in order to map the execution of a virtual task instance from the view of the workflow execution engine to the actual continually executed task instance.

We evaluated our proposed approach with the Montage workflow in a simulation as well as a real experiment and observed that on the one hand the results from simulation were comparable to the real results, and on the other hand we could achieve a reduction of the total workflow runtime by up to 15%. Moreover, simulation shows that the potential of outperforming the original workflow by the transformed workflow where an earlier start of the aggregation task is achieved, grows with an increasing computing cluster and workflow size.

The proposed scheme can be regarded as a contrary approach to the already established task clustering technique that groups small independent tasks into one single virtual task in order to reduce the execution staging overhead. In contrast, by splitting data intensive aggregation tasks, unnecessary delays during the execution can be reduced and the overall performance of the workflow increased. Similar to task clustering, task splitting is not limited only to Pegasus WMS, which we used in our evaluation, but can also be applied to other workflow execution engines that base on the same execution semantics.

ACKNOWLEDGMENTS

Wladislaw Gusew is funded by the DFG through grant GRK 1651. We would like to thank Olaf Menzel and Steffen Tschirpke for helpful comments and discussions. We further acknowledge the anonymous reviewers for their feedback and suggestions.

REFERENCES

- [1] J. Yu and R. Buyya, "A taxonomy of workflow management systems for grid computing," *Journal of Grid Computing*, vol. 3, no. 3-4, pp. 171–200, 2005.
- [2] Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, and J. Myers, "Examining the challenges of scientific workflows," *IEEE Computer*, vol. 40, no. 12, pp. 26–34, 2007.
- [3] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, "Characterization of scientific workflows," in *WORKS '08*, Nov. 2008, pp. 1–10.
- [4] T. Hey, S. Tansley, and K. Tolle, Eds., *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, 2009.
- [5] D. De Roure, C. Goble, S. Aleksejevs, S. Bechhofer, J. Bhagat, D. Cruickshank, P. Fisher, N. Kollara, D. Michaelides, P. Missier, D. Newman, M. Ramsden, M. Roos, K. Wolstencroft, E. Zaluska, and J. Zhao, "The evolution of myExperiment," in *E-SCIENCE '10*. IEEE Computer Society, Dec. 2010, pp. 153–160.
- [6] D. Thain, T. Tannenbaum, and M. Livny, "Condor and the grid," in *Grid Computing: Making the Global Infrastructure a Reality*, F. Berman, G. Fox, and A. Hey, Eds. John Wiley & Sons Inc., Apr. 2003.
- [7] A. Mandal, P. Ruth, I. Baldin, Y. Xin, C. Castillo, M. Rynge, and E. Deelman, "Evaluating I/O aware network management for scientific workflows on networked clouds," in *NDM '13*. ACM/IEEE, Nov. 2013.
- [8] L. B. Costa, H. Yang, E. Vairavanathan, A. Barros, K. Maheshwari, G. Fedak, D. Katz, M. Wilde, M. Ripeanu, and S. Al-Kiswany, "The case for workflow-aware storage: An opportunity study," *Journal of Grid Computing*, vol. 13, no. 1, pp. 95–113, 2014.
- [9] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, B. Vahi, K. Berrian, J. Good, A. Laity, J. Jacob, and D. Katz, "Pegasus: a framework for mapping complex scientific workflows onto distributed systems," *Scientific Programming*, vol. 13, no. 3, pp. 219–237, 2005.
- [10] J. Yu and R. Buyya, "Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms," *Scientific Programming*, vol. 14, no. 3-4, pp. 217–230, 2006.
- [11] H. Kim, I. Cho, and H. Yeom, "A task pipelining framework for e-science workflow management systems," in *CCGRID '08*, May 2008, pp. 657–662.
- [12] W. Chen and E. Deelman, "WorkflowSim: A toolkit for simulating scientific workflows in distributed environments," in *E-SCIENCE '12*. IEEE, Oct. 2012, pp. 1–8.
- [13] W. Chen, R. F. Da Silva, E. Deelman, and R. Sakellariou, "Balanced task clustering in scientific workflows," in *E-SCIENCE '13*. IEEE, Oct. 2013, pp. 188–195.
- [14] J. Jacob, D. Katz, B. Berrian, J. Good, A. Laity, E. Deelman, C. Kesselman, G. Singh, M.-H. Su, T. Prince, and R. Williams, "Montage: a grid portal and software toolkit for science-grade astronomical image mosaicking," *International Journal of Computational Science and Engineering*, vol. 4, no. 2, pp. 73–87, 2009.
- [15] T. McPhillips and S. Bowers, "An approach for pipelining nested collections in scientific workflows," *ACM Sigmod Record*, vol. 34, no. 3, pp. 12–17, 2005.
- [16] S. Bowers, T. McPhillips, S. Riddle, M. Anand, and B. Ludäscher, "Kepler/pPOD: Scientific workflow and provenance support for assembling the tree of life," in *IPAW '08*. Springer, Jun. 2008, pp. 70–77.
- [17] C. Liew, M. Atkinson, J. van Hemert, and L. Han, "Towards optimising distributed data streaming graphs using parallel streams," ACM, Jun. 2010, pp. 725–736.
- [18] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," *Future Generation Computer Systems*, vol. 29, no. 3, pp. 682–692, 2013.
- [19] G. Juve, B. Tovar, R. da Silva, D. Król, D. Thain, E. Deelman, W. Allcock, and M. Livny, "Practical resource monitoring for robust high throughput computing," in *CLUSTER '15*. IEEE, Sep. 2015.
- [20] R. da Silva, G. Juve, M. Rynge, E. Deelman, and M. Livny, "Online task resource consumption prediction for scientific workflows," *Parallel Processing Letters*, vol. 25, no. 03, 2015.
- [21] E. Deelman, G. Mehta, J.-S. Vöckler, Y. Zhao, and M. Wilde, "Kick-starting remote applications," in *GCE '06*, Nov. 2006.
- [22] W. Chen, R. F. da Silva, E. Deelman, and R. Sakellariou, "Using imbalance metrics to optimize task clustering in scientific workflow executions," *Future Generation Computer Systems*, vol. 46, pp. 69–84, 2015.
- [23] Z. Zhang, D. Katz, J. Wozniak, A. Espinosa, and I. Foster, "Design and analysis of data management in scalable parallel scripting," in *SC '12*. IEEE, Nov. 2012, p. 11.
- [24] "SystemTap project as free software (GPL) with documentation and examples," <https://sourceware.org/systemtap/>, 2016, accessed: 2016-08-18.
- [25] Filesystem in Userspace, "Reference implementation of the Linux FUSE interface," <https://github.com/libfuse/libfuse/>, 2016, accessed: 2016-08-18.
- [26] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. Maechling, R. Mayani, W. Chen, R. F. da Silva, M. Livny, and K. Wenger, "Pegasus, a workflow management system for science automation," *Future Generation Computer Systems*, vol. 46, pp. 17–35, 2015.
- [27] H. Topcuoglu, S. Hariri, and M. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, 2002.

Automatic Glomerulus Extraction in Whole Slide Images Towards Computer Aided Diagnosis

Yan Zhao, Edgar F. Black, Luigi Marini
and Kenton McHenry

National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign
{yanzhao3, lmarini, mchenry}@illinois.edu

Norma Kenyon
Diabetes Research Institute
University of Miami
NKenyon@med.miami.edu

Rachana Patil, Andre Balla
Amelia Bartholomew
Department of Surgery and Bioengineering
University of Illinois at Chicago
{rpatil3, ambart}@uic.edu

Abstract—Renal biopsies form the gold standard of diagnostic and prognostic assessments of renal transplants. With the addition of new quantitative strategies to supplement renal biopsy interpretation such as gene array and metabolomics, the capability to incorporate all quantitative measures for clinical interpretation will require multi-dimensional analyses. Currently, renal biopsies are analyzed manually; the quantitative features of pathology observed on the biopsies are limited to hand counts. Standardized, automated detection of pathology observed in a kidney transplant biopsy will enable the input of these digital images alongside other quantitative measures of new technologies, with potential gains in precision in patient care. We investigate a learning framework to detect pathological changes in biopsy image that addresses two main issues: the inadequate training set and the significant diversity of color and tissue shape on whole slide images. Two case studies, automatic detection of interstitial inflammation and tubular cast, are presented in this work. Afterwards, we propose a fully automated glomerulus extraction framework on micrograph of entire renal tissue, focusing on extracting Bowman's capsule, the supportive structure of glomeruli. Statistical approaches are also introduced to further improve the performance. Human expert annotations of interstitial inflammation and tubular casts in 10 H&E stained renal tissues of nonhuman primates and more than 100 glomeruli are used to demonstrate the superior performance of the proposed algorithm over existing solutions.

I. INTRODUCTION

Renal biopsies provide diagnostic and prognostic assessments of the kidney transplant, playing a critical role in the prevention of graft loss [1], [2]. Detection of the wide variety of pathologic features on biopsy can be accomplished by traditional morphologic techniques including light microscopy, immunofluorescence, and immunohistochemistry [3], [4]. However, emerging, potentially complementary technologies are being developed to improve accuracy and predictive power [5]. Strategies being combined with renal biopsy have included gene expression profiling [6], metabolomics [7], and whole slide scanning for morphometric attributes [8]. The magnitude of data which can now be obtained per biopsy in combination with patient attributes requires increasingly more sophisticated modes of analyses to incorporate the contributions of each data type. Gene array data has discrete quantitative data sets, however examination of the renal biopsies using traditional morphologic techniques has relied upon manual

annotation and quantitative measurement of structures by expert pathologists. Input of morphologic data into this multidimensional analysis will be challenging without digital, automated, quantifiable attributes. Our goal is to develop a fully automated histologic annotation of glomeruli, interstitial inflammation, and tubular casts using digital whole slide scans of post-transplant renal biopsies as an initial framework for the development of multi-dimensional analysis platforms.

Glomeruli are critical in the identification of renal pathology; accuracy in assessment is predicated on examination of sufficient number per biopsy. Below the minimum number of 25 glomeruli per biopsy, correlation with clinical outcomes becomes diminished [3], hence comprehensive ability to assess all glomeruli, as can potentially be accomplished in a whole slide morphology assessment, can be beneficial to patient outcomes. A glomerulus can be well identified by routine hematoxylin and eosin (H&E) staining based on its supportive structure, Bowmans capsule, and small knots of capillaries in the middle. Color-based segmentation has been proven successful in distinguishing potential Bowman's capsule from background [9]–[11].

Despite these capabilities, whole slide medical image analysis as a research field remains in its infancy; digital scans of whole slide pathology, especially with unique parameterization, has encountered obstacles, impeded by irregular, varied shapes and blurred boundaries. In comparison with photo recognition technology, these features of tissue images serve as challenges for direct application of classical image analysis for computational pathology.

A shift from rule-oriented expert systems to learning-oriented statistical models presents in pattern recognition in general and computational pathology in particular [12], [13]. In this direction, supervised classification was trained after iterative morphological operations for cell detection [14]. Geometric features, in conjunction with color features, were fed into Support Vector Machine (SVM) [15] to capture the properties of a nucleus. In [16], Maximally Stable Extremal Region Detector (MSER), a low computational cost method, was applied to non-overlapping regions for cell detection, together with binary classification. In [17], M. DiFranco *et al.* systematically evaluated texture features

selection on computer-aided diagnosis of prostate cancer.

With the objective of lowering expertise requirement for content retrieval, collaborative biomedical image-mining frameworks can be created. Theodosis Goudas *et al.* proposed DWMF ontology [18], a commonly adopted framework, and extensively studied the output of three most widely utilized classifiers in the field of computational pathology. Glomerulus segmentation, as an example of the applications of DWMF ontology, was implemented to demonstrate the usability and accuracy of the image-mining framework.

This work extends the Clinical Research Informatics (CRI) toolbox¹ towards the automated ingestion of provided project data into a formal database towards broad accessibility, use, and preservation. On top of this, the CRI toolbox utilizes a number of web-based visualizations and analysis tools to aid domain scientists in their navigation and exploration of the data. We have previously applied such data mining techniques in medical research, using a combination of numerical values of physical signs and blood analytics, to test the immunoregulatory effects of mesenchymal stem cells in a cynomolgus monkey model of islet transplantation [19]. A Hidden Markov Model (HMM) [20] was applied as a means of modeling temporal aspects of control of diabetes in nonhuman diabetic primates after undergoing isolated islet cell transplant. In line with our prior work in predicting outcomes post-transplant, we extend our work to examine post-transplant renal biopsies, considered the gold standard of kidney transplant diagnostic and prognostic measures [1]. Our learning-oriented framework is proven to achieve the state-of-the-art accuracy on 10 testing whole slide images (WSIs). In addition, large-scale images, whose size runs up to the order of 300M pixels, are utilized to empirically evaluate the efficiency.

In summary, the contributions of this paper are the following:

- We design a standardized, fully automated histologic annotation of glomeruli, interstitial inflammation, and tubular casts framework for digital whole slides.
- The proposed algorithm is less sensitive to the shape and size of glomeruli and the thickness of Bowman's capsule compared to existing algorithms.
- Our high-throughput framework involves no recursive operations or computational intensive algorithms in context of WSIs. Further acceleration can be achieved by parallelizing the most computational expensive components.
- The framework consists of techniques with high maturity, and can be easily implemented with available packages. This work is also implemented in Clowder², a scalable research data management system, as web service.

In this paper, the remainder is organized as follows: Section II reviews related work on glomerulus identification

¹<https://opensource.ncsa.illinois.edu/confluence/display/CRI/Clinical+Research+Informatics>

²<https://clowder.ncsa.illinois.edu/>

algorithms. Section III offers the proposed two-step learning framework for Computer Aided Diagnosis (CAD) and details the characteristics of interstitial inflammation (ITIF) and tubular cast (TC). Section IV details our design of glomerulus extraction algorithm. Section V reports our experimental evaluations, Section VI overviews CRI toolbox and Section VII concludes the paper.

II. RELATED WORK

In this section, we discuss related work on computer-aided glomerulus detection.

There have been some initial studies on automated glomerulus extraction [9], [21] since 2008. Jun Zhang *et al.* [9] described glomerulus extraction that first convoluted image with LOG filter to define the boundary and then optimized fitting curve through modified cubic spline interpolation. Jiaxin Ma *et al.* [21] developed an edge patching method with a genetic algorithm.

Continuing in this direction, color-based segmentation, utilized in [10] [11], was reported to be successful in differentiating potential Bowman's capsule from the background. Siddharth Samsi *et al.* [11] executed perceptual pairwise grouping for glomerulus identification after Bowman's capsule was extracted. Such grouping adopted three metrics, proximity, co-circularity and orientation, and needed to skeletonize Bowman's capsule. In the latest study, Taras Kotyk *et al.* [10] described an approach to measure the diameter of glomeruli beyond glomerulus detection. Obvious difference of glomerular diameters between the experimental and control groups was reported. However, the accuracy of these approaches dropped greatly when they were applied on whole slide histological images. This experimental result was primarily explained by irregular shape of glomeruli and interference of interstitial renal space.

Histogram of oriented gradients (HOG) [22], a widely used feature descriptor, was first introduced in [23] to locate glomeruli on WSIs. Rectangular Histogram of Oriented Gradient (R-HOG) and Segmental Histogram of Oriented Gradient (S-HOG) were involved to capture the physical characteristics, especially the boundary, of glomeruli. HOG descriptor is a theoretically color-free descriptor, avoiding the problem of color inconsistency of WSIs. With appropriate tuning strategies, the novel descriptors were capable of dealing with the issue of various sizes and irregular shapes of glomeruli as well. Tsuyoshi Kato *et al.* presented promising result that both classification steps could achieve recall as high as 90% on experts annotated dataset. Nevertheless, the requirement that a tile precisely centered the specific glomerulus in S-HOG necessitates a much smaller step size of sliding tiles in R-HOG.

By closely studying the computational components of existing works, we develop an adaptive glomerulus annotation framework for whole slide digital scanning of entire renal tissues in Section IV. To our best knowledge, this is the first work that empirically study the performance of fully automated glomerulus extraction on WSIs.

III. COMPUTER AIDED DIAGNOSIS

In this section, we first identify the challenges and formulate the two-step learning workflow for digital whole slide scans of post-transplant renal biopsies, then discuss the applications in two scenarios, interstitial inflammation and tubular cast.

A. Basic Workflow

Ghaznavi *et al.* [24] summarized the challenges of Computer Aided Diagnosis (CAD) caused in scanning procedure, such as tissue fold and dirty media. Apart from these challenges, two main issues are listed as follows:

- Lack of training set. Compared with benign or normal histopathological images, the number of H&E stained slides with a specific pathological change is limited, and this situation is worsened by the limited share between institutions. The inadequate number of samples for classification causes problems such as overfitting, i.e. the classification yields high precision and recall for the dataset examined but losses the general functionality on tissues from other dataset.
- Diagnosis-cross problem, i.e. the unclear effect of appearance of Disease A on the automatic detection of Disease B. For example, infiltration of lymphocytes or plasma cells potentially influence the digital measurement of tubules.

With the objective to address the issues mentioned above, we formulate our two-round classification solution for CAD.

1) *Pre-Screening*: Initially, a combination of tiles of image in sliding manner and machine learning technique [25] [26] is employed to preliminarily discard undesired area. We use axis-parallel, square tiles for simplicity in this work.

Note that hematoxylin is to stain nuclei blue, while eosin stains cytoplasm and the extracellular connective tissue matrix pink, we consider four basic GLCM features, contrast, correlation, homogeneity and energy in blue and red channel respectively. GLCM Tool³ calculates of the eight texture descriptors.

2) *Search the Biomarkers of Diagnosis*: In the second round of learning system, a diagnosis-specified classification is employed to further filter the positives of Pre-Screening.

First, to capture the texture information in a wider context, we utilize bigger tiles than those in Pre-Screening. The size of tile is essentially determined by the characteristic of different biomarkers. For example, the size of normal tubule is generally small than a typical glomerulus, thus the tiles for intra-tubular disease detection are smaller, or narrower than those for intra-glomerular disease. In contrast, interstitial disease always spreads in a large area and surrounds other biological components, requiring big tiles.

Furthermore, the optimized features selection is considered based on closely studying the difference of histological structure between the false positives from Pre-Screening and the real biomarkers.

³<http://www.mathworks.com/matlabcentral/fileexchange/22187-glcmt-texture-features>

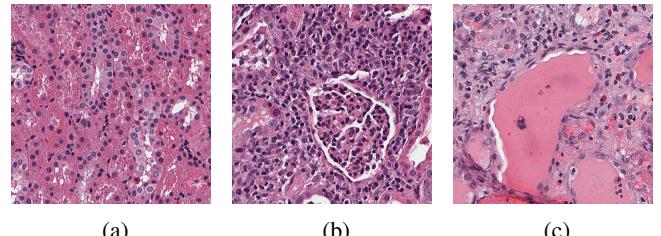


Fig. 1: Representative examples of pathological biomarkers. (a) Normal renal tissue from CN8450; (b) Interstitial inflammation biomarker surrounding a glomerulus from CN8452; (c) Tubular cast biomarker from CN8349. Each sample is 500x500 pixels.

We name the general classifier in Pre-Screening as G-SVM and the specific classifier in this phase as S-SVM. It is noteworthy that rather than mapping a particular tile into different categories, the ultimate goal of the learning system is to accurately report pathological changes in the context of the whole slides.

B. Interstitial Inflammation

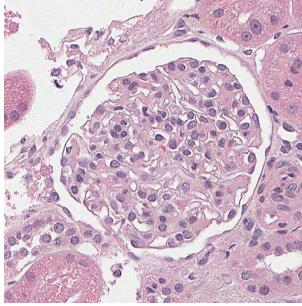
Renal interstitium refers to the area of the kidney in which nephrons and vessels are embedded, and can contain inflammatory cells, extracellular matrix and interstitial fluid [27]. The Banff classification of renal transplant rejection includes a semiquantitative grading of interstitial inflammation, defining the type of rejection based on the percent of interstitial infiltration of inflammatory cells [28]; in addition, standardized descriptions of the degree of inflammation of the tubules (tubulitis, t-score) and the arteries (arteritis, v-score) contribute towards a more uniform interpretation of the post-transplant renal biopsy.

As demonstrated in Fig.1b, interstitial inflammation typically presents as small knots of capillaries tightly organized, similar to the internal structure of glomeruli. First, the histological structure of interstitial inflammation yields high energy and low homogeneity values. Second, the hue channel of inflammation area in H&E stained slides is around 240, the blue color. Third, for severe interstitial inflammation area, renal interstitium is hardly observed.

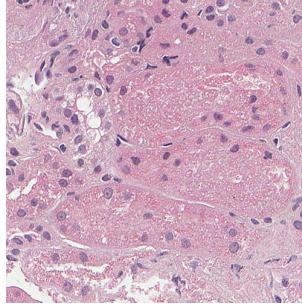
C. Tubular Cast

As an example of a tubular abnormality that can be readily detected by this method, we examined tubular casts which can be observed following the administration of rapamycin [28].

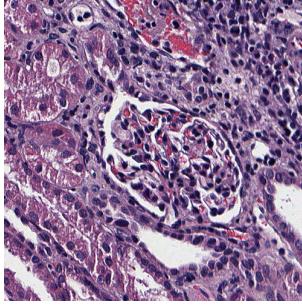
Fig.1c demonstrates an example of renal tubular casts occupying a distal convoluted tubule. The histological structure of hyaline casts exhibits smooth and pick, which yields small local range of color changes. The intra-tubular casts are surrounded by cells, which form a dot boundary and separate the cast from interstitial foreground. Some or many of surroundings are macrophages and multinucleated giant cells, essential for diagnosis. Apoptotic cell debris appears as the dark knot inside the hyaline cast. The cast progressively grows



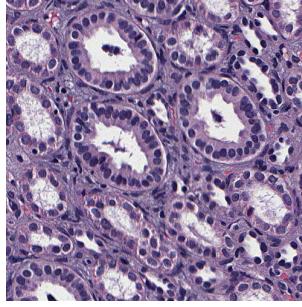
(a) Positive sample from CN8621



(b) Negative sample from CN8621



(c) Positive sample from CN8376



(d) Negative sample from CN8376

Fig. 2: Nonhuman primates glomeruli in two H&E stained tissues compared with non-glomerular counterparts. Each sample is 500x500 pixels.

until breaking and destroying the tubule. Dense intratubular casts as well as the large size of casts in CN8349 slide, as shown in Fig.1c, indicate the severity of damage of the tubules and renal interstitium.

IV. GLOMERULUS EXTRACTION

In this section, we detail the underlying design of glomerulus extraction. Statistical approaches are introduced to extract Bowman's capsules, the white outer lines of glomeruli in H&E stained image.

Fig.2 shows examples of nonhuman primates glomeruli in two H&E stained tissues compared with non-glomerular counterparts. The variation of color of the slides and inconsistency of the shape, size and orientation of glomeruli are depicted.

A. Object Generation

Color segmentation is commonly employed as the first step to annotate Bowman's space and binarize color image. To perform color segmentation, K-means clustering [29] using two color channels, a^* in La* b and magenta in CMYK color space, was trained in [11]. However, K-means replicates clustering to avoid local minima and the computational time becomes unbearable when extending the algorithm onto WSIs. As a result, we adopt a simple and efficient threshold system that uses green channel in RGB color space, as suggested in [10]. The threshold is the sixth level of green channel. In particular, the computation of threshold merely considers foreground pixels. We name connected positive pixels on the binary image as a object.

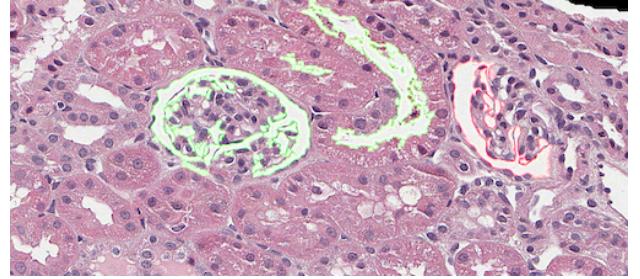


Fig. 3: Detection of the capsule coordinates. The image takes from CN8621. The candidate objects satisfy conditions $AspectRatio > 1.3$, $Circularity < 0.1$, $Solidity < 0.4$ are depicted in green line. Red outlined object is a real Bowman's capsule failing the conditions.

Enhancement in the procedure of Object Generation has been tried to identify the Bowman's capsule with insignificant thickness. Top-hat Filtered image is fed into color segmentation in conjunction with the normal gray image. Then, morphological closing on small-sized objects is applied prior to discard those with extremely small size, whereas large-sized objects with the length out of the range of a normal glomerulus are globally filtered out. We name the kept objects as candidate objects.

B. Morphological Classification

To discard undesired objects, [10] filtered candidate objects by quantifying three morphological properties as follows:

- **Aspect Ratio (AR)**, denoting the ratio between the major and the minor axis of the object;
- **Solidity**, denoting the ratio between the number of pixels of the object compared to the associated convex hull;
- **Circularity**, equivalent of $4 \times \text{area}/\text{perimeter}^2$.

Unfortunately, the rule-based method does not allow more complex capsule shape with varied length and thickness. Take Fig.3 as example, the green outlined objects in the middle are actually noises while the red outlined real Bowman's capsule is dropped. By empirically studying the morphological descriptors of Bowman's capsule associated objects, it is found that the solidity ranges diversely from 0.1 to 0.4, and the eccentricity is inferior to 0.4. In addition, the shorter the maximum length of a candidate object is, the smaller fragment of the ellipse's boundary it represents and the higher solidity and eccentricity value the candidate object has. When the maximum length of Bowman's capsule object approximately equals to the typical glomerular diameter, the higher eccentricity of a candidate object is, the smaller fragment of the boundary it represents and the higher solidity value the candidate object has. Consequently, an SVM is trained to address the issues of the threshold method. The input feature vectors of SVM_{morph} include maximum length, solidity and eccentricity, a similar measurement to circularity. We name this SVM as SVM_{morph} and the positives of SVM_{morph} as selected objects.

Despite SVM classification yields higher precision than threshold method, optimization is still required prior to identify glomeruli. We describe the grouping of selected objects in the next subsection.

C. Perceptual Grouping

In accordance with [11], perceptual pairwise grouping captures the properties of glomeruli in terms of proximity and orientation. Nevertheless, pairwise grouping in the context of WSIs is time-consuming and unnecessary. Therefore, we consider a simple alternative.

The selected object with the maximum length longer than a threshold is treated as the basis and perceptual grouping casts a binary vote for other selected objects being inside or outside the group. We name this selected object as BWmax. A Region of Interest (ROI) centered at the centroid of BWmax is generated.

For other selected objects in this ROI, we first filter them based on their proximity to BWmax. Proximity is computed as the distance between the centroid of BWmax and the centroid of another selected object. The selected objects with distance longer than a threshold are eliminated.

The round shapes of glomeruli make orientation an important metrics for grouping selected objects [11]. However, it is deserved to notice that the orientation is difficult to measure for short fragment of Bowman's space. As shown in Fig.2c, the skeleton of fragment of Bowman's space in bottom-left is nearly a straight line. In this sense, we employ a novel approach that only considers the orientation of BWmax.

Specifically, define $\overline{C_{BWmax}C_{object}}$ is the link of the center of BWmax and the convex hull center of another selected object. We verify the relative position between BWmax and $\overline{C_{BWmax}C_{object}}$ to eliminate the selected objects outside of the potential glomerulus. A fundamental assumption is that BWmax faces away from selected objects if the link intersects with BWmax. Otherwise, BWmax fronts the selected object. An implementation provided by Kanchi⁴ computes the links.

D. Glomerulus Segmentation

All objects selected in previous subsection are considered as the input of glomerulus segmentation. The glomerulus in Fig.2c exhibits an irregular histological structure, which may be caused by renal disease or distortion of scanned tissue. To this end, glomeruli cannot be simply described as circles or ellipses. A simple representation of a glomerulus would be the convex hull of all selected objects in an ROI.

Finally, we eliminate the convex hulls with small size or low AR according to geometric characteristics of glomeruli.

E. Summary of Method

Algorithm 1 sketches an overview of glomerulus extraction and Fig.4 depicts the steps of workflow using Fig.2 as example. Associated parameters in Algorithm 1 are optimized via cross-validation on training dataset. Also, ROIs in Fig.4

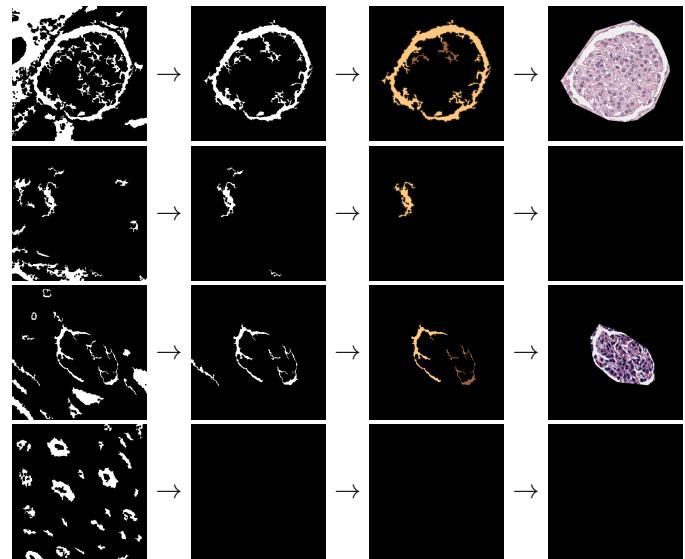


Fig. 4: Steps of glomerulus extraction using Fig. 2 as example. Column (1) Object Generation; Column (2) Selected objects in Morphological Classification; Column (3) Perceptual Grouping, objects with light brown are BWmax, with dark brown are in-group selected objects; Column (4) The extracted glomerulus from Glomerulus Segmentation.

Algorithm 1 Glomerulus Extraction

Input: RGB image, trained *SVMmorph*
Output: Binary image or centroids of glomeruli

- 1: Convert color image to gray image G
- 2: Convert gray image to binary image:
 $I = G > \text{threshold} \mid \text{Top-hat}(G) > \text{threshold}$
- 3: ImageA = objects of I with Area > 5000 ,
 ImageB = I - ImageA
- 4: Discard objects with $\text{MaxLength} > 650$ on ImageA
- 5: Morphological closing with radius = 3, disk-shaped structuring element on ImageB
- 6: Discard objects with #Pixel < 200 on ImageB
- 7: Input MaxLength, Solidity and Eccentricity of each object to *SVMmorph*
- 8: **for all** selected objects with $\text{MaxLength} > 120$, e.g. BWmax **do**
- 9: Generate a 500x500 ROI centers at BWmax
- 10: **for all** selected objects in ROI **do**
- 11: C_{object} = centroid of the selected object
- 12: **if** $|C_{BWmax} - C_{object}| < \max(200, \text{MaxLength} - \text{MinorLength}/2 \text{ of BWmax}) \&$
 $C_{BWmax}C_{object}$ not intersect with BWmax **then**
- 13: Keep this object
- 14: Generated convex hull in ROI
- 15: Report a glomerulus if the convex hull satisfies the condition: $\text{AR} < 2.2$, $\text{Area} > 14000$

⁴<http://www.mathworks.com/matlabcentral/fileexchange/4211-connect-two-pixels>

are actually the original samples, while in our experiment ROI should center at the centroid of BWmax.

V. EXPERIMENT

A. Experimental Setup

Our experiments are conducted on a desktop with Intel i7 CPU, 2.5GHz. The operating system is OS X 10.11.3 64-bit with MATLAB R2015b installed. The entire evaluation program is conducted with MATLAB. SVM, Principal component analysis (PCA) [30] and the morphological features mentioned in Section III and IV are implemented by using MATLAB package *fitcsvm*, *pca* and *regionprops*, respectively, with default parameters without specified otherwise. The kernel function of SVM is Radial Basis Function [31] with automatic fitting scale. Executional time is the average of three repeated measurements conducted by the MATLAB *profile* function.

The experimental work is performed using a dataset of images scanned on a ScanScope virtual slide light microscope scanner from Aperio Technologies Inc. A lens with a magnification of 20x is used, which results in a per pixel resolution of $0.5\mu\text{m}$. During preprocessing phase, slides are converted from *svs* to *tiff* format with VIPS⁵ tool and the background of micrographs is removed with MATLAB image segmentation tool⁶.

The training and testing sets in the following evaluation are whole slides of nonhuman primate kidneys. Both biopsy and necropsy images are utilized. Specifically, CN8376 and CN8454 biopsies are retrieved from rejected kidneys with interstitial inflammation, CN8452 demonstrates acute rejection with hemorrhage and lymphocytes infiltrating the tubular membrane, CN8349 demonstrates rejection with tubular casts and interstitial inflammation. CN8450, CN8621 and CN8383 are biopsies taken from normal kidneys, excised from the recipient at the time of the renal transplant. The size of images ranges from 7M to 100M pixels and the variety of color is demonstrated in Fig.1 and 2. Expert annotations are provided along with the images.

B. Evaluation of Interstitial Inflammation Detection

In this subsection, we analyze the automated detection of Interstitial Inflammation (ITIF) by evaluating the two systems of classifications separately. We reiterate the similar pathological structure between internal glomeruli and ITIF, thus only the outputs absolutely outside of glomeruli are considered.

A tile with positive vote is considered to be a true positive (TP) if its center sits on the true category. Otherwise, it is a false positive (FP). Precision is mathematically defined as the ratio of the number of TPs to the number of all selection elements (or positives), whereas recall is the ratio of the number of TP to that of relevant elements (true category).

We define G-SVM with default parameters and 50-pixel tiles as baseline, compared with three methods as follows:

⁵<https://github.com/openslide/openslide>

⁶<http://www.mathworks.com/examples/image/2099-detecting-a-cell-using-image-segmentation>

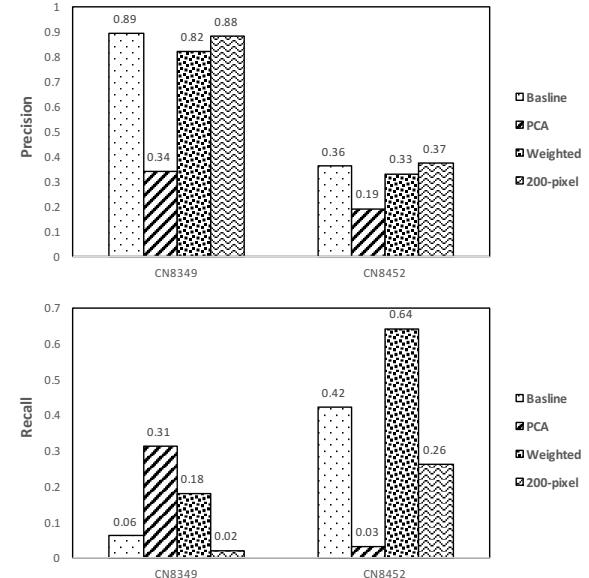


Fig. 5: Comparison of precision and recall for G-SVM of ITIF detection with respect to the three parameterizations: PCA, twofold weighted positive and 200-pixel tile input.

- **PCA** Principal component analysis [30] is commonly employed before feeding feature vectors into SVM in medical image analysis [24]. On one hand, PCA decreases the complex and increases the efficiency of SVM by linear dimensionality reduction. On the other hand, the last principal component has the smallest inconsistency, thus represents the most common feature information of the positives. In this sense, PCA enhances generalization capability of the classifier. We employ the last three vectors generated by PCA.
- **Weighted** To boost the recall for cell detection, positive training set is given double unit weight. The rationale for this parameterization is that the subsequent classification allows filtering false positives of G-SVM.
- **200-pixel** We use tiles with 200x200 pixels for comparison.

The corresponding precision and recall are summarized in Fig.5. In our experiment, PCA turns out to be the method with the lowest precision, as low as 4% for CN8454. This unsatisfied result is primarily explained by generally low correlation between texture features. Therefore attempting to improve the classification using linear transformation techniques is not advisable.

Contrary to PCA, double-weighted-positive yields more positives, leading to higher recall by marginally sacrificing the precision. Consequently, we take Weighted as G-SVM.

The FPs of training set generated from G-SVM are selected as the negative samples for S-SVM, as shown in the first row of Fig.6. First, Fig.6 clearly reveals relatively dense distribution of nuclei in positive samples. Therefore, a natural descriptor of cell distribution, the number of pixels with weak

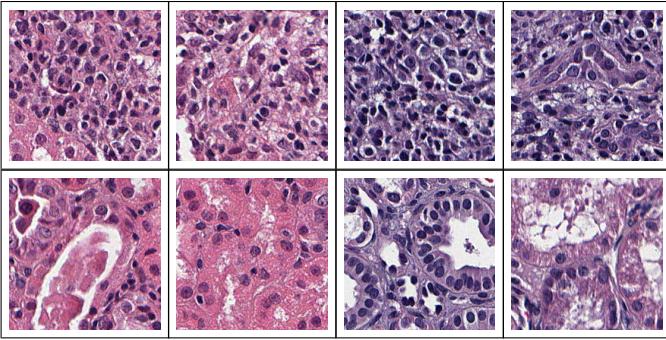


Fig. 6: Examples of training set for S-SVM of ITIF detection. Each sample is 200x200 pixels, the tiles in the first row are positive samples and in the second row are negative samples.

color intensity with maximum length ranging from 3 to 30 pixel, is fed into the second classifier. Second, histological structures in negative samples have relatively light color whereas renal interstitium in the positive samples is infiltrated by lymphocytes or neutrophils. Thus, the other feature vector is defined as the percentage of light color area above 100 pixels. We also adopt information measure of correlation [32], cluster prominence [33], maximum probability [33] and inverse difference moment normalized [34] in S-SVM and use 200-pixel tiles for S-SVM.

With proper selection of the feature space, the precision of S-SVM can reach as high as 98%. We then test the model on four normal WSIs. In most cases, FPs appear separated whereas TPs are concentrated. A small proportion, less than 1%, of FPs are reported, and mostly are discarded.

In terms of computational efficiency, the tiling of the whole-slide images allows the coarse-grained parallelism in processing each image tile independently, whereas fine-grained parallelism, which can be implemented on GPUs or heterogeneous platform, can be achieved through processing each pixel independently.

C. Evaluation of Tubular Casts Detection

In this subsection, we first quantitatively evaluate Pre-Screening step of computer-aided TC detection in this subsection by comparing the baseline with PCA, double weighted positive and 100-pixel samples. Fig.7 depicts the corresponding result, drawing a conclusion in agree with that for ITIF detection. In most cases, the precision and recall of 100-pixel are lower than the baseline. This result justifies our assumption in Section III.A that smaller tiles are suitable for intratubular disease learning and detection.

The positive samples for S-SVM are compared with negative samples in Fig.8. By investigating the differences, optimal feature selection is studied to precisely capture the properties of TC in terms of color, shape and size. First, TC samples generally exhibit a substantial variation of color intensity compared with the negative samples. Accordingly, the first two features are the median of local range of color among 7*7 neighborhood in red and blue channel, respectively.

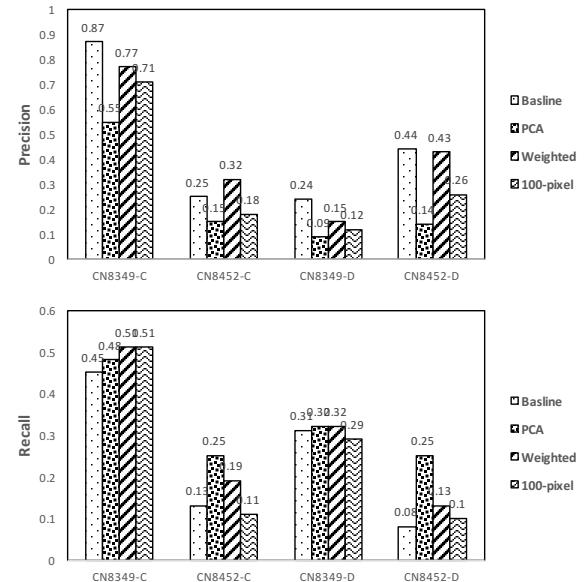


Fig. 7: Comparison of precision and recall for G-SVM of TC detection with respect to the three parameterizations: PCA, twofold weighted positive and 100-pixel tile input.

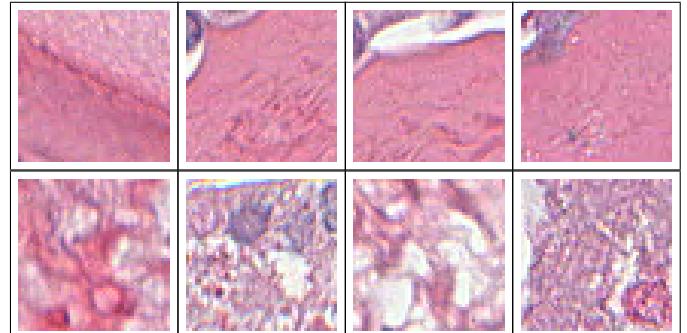


Fig. 8: Examples of training set for S-SVM of TC detection. Each sample is 50x50 pixels, and the tiles in the first row are positive samples and in the second row are negative samples.

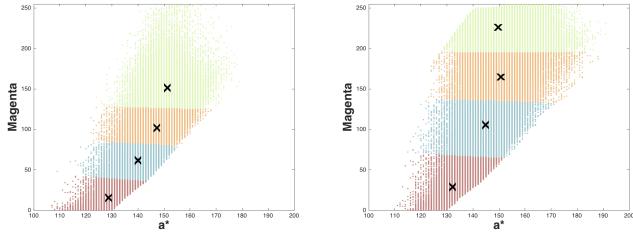
In addition, note that the color saturation of hipline tubular casts is higher than that of non-cast area, another two features are the median of hue and saturation channels. We also adopt information measure of difference variance [32], difference entropy [32] and sum entropy [32] in S-SVM.

The precision of S-SVM in four testing set can reach as high as approximate 98%. Continuing in this direction, we experimentally test on two slides of benign tissues. Merely one or two false positive tiles in each WSI are reported, which appear negligible with respect to the large scale of the slides and usually are discarded.

D. Evaluation of Glomerulus Extraction

In this subsection, we empirically study glomerulus extraction.

We first train K-means clustering on color channel of a* and magenta as previously proposed in [11]. Fig.9 illustrates



(a) A tile of CN8621

(b) A tile of CN8376

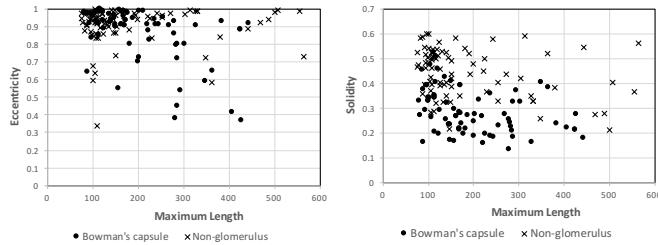
Fig. 9: The result of K-means cluster in the color space of a^* and magenta.

Fig. 10: The 2D scatter plot of eccentricity and solidity of objects along with their maximum length.

TABLE I: Overall Performance

Parameters	CN8454	CN8452	CN8450	CN8383	CN8376
# Real Glomeruli	8	6	37	80	9
# Glomeruli Detected	5	4	23	73	5
Percentage	80.6%	100%	91.6%	96.9%	73.0%
# False objects	0	0	1	15	13
Time(s)	9.3	3.5	10.2	65.6	12.3

the clustering result on Fig.2a and 2c, in which intersects represent the centers of clusters. On one hand, despite of the variation of hue and intensity of the two images, it turns out that both clustering methods largely depend on magenta channel whereas a^* channel plays an insignificant role. On the other hand, the execution of K-means takes 292s for 24M-pixel CN8621 slides, directly hindering an application in a high-throughput context. As a result, K-means clustering can be replaced by threshold method.

On top of color segmentation, we continually study the morphological classification step in glomerulus extraction. *SVMmorph* uses 50 positive and negative samples from 4 slides, respectively. Each sample consists of one connected object generated by the method demonstrated in Section IV.A. In order to increase the accuracy of the SVM, all connected object samples are longer than 80 pixels.

As depicted in Fig.10, the eccentricity of Bowman's capsule is closed to 1 when the maximum length is low than 200 pixels, whereas the solidity is nearly inverse correlation to the maximum length. The supervised classification utilized as the basis for filtering non-glomerular objects is advantageous compared to rule-based approaches, as this technique delivered consistent results despite a variation in the shape of Bowman's

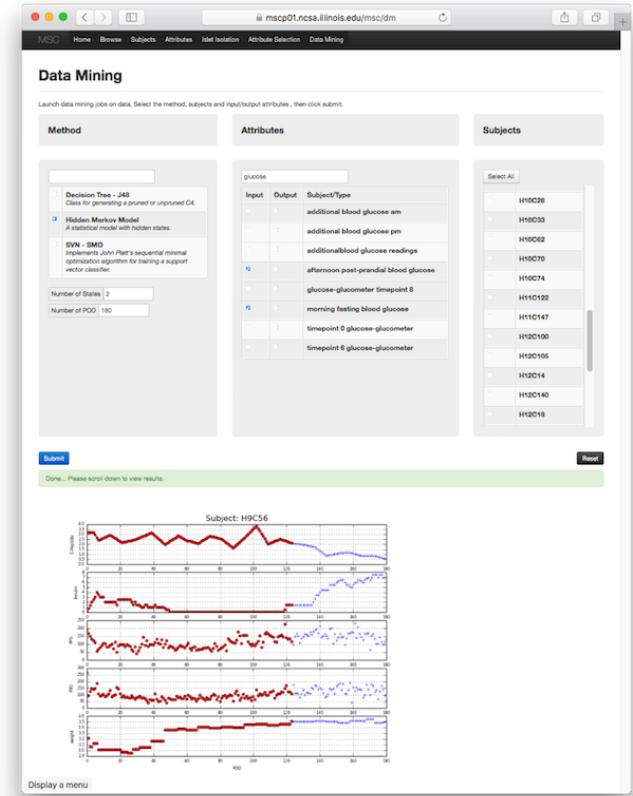


Fig. 11: The web interface of mining numerical values of physical signs regarding MSC. The bottom graph is the history of state changes produced by 2-state HMM.

capsules.

Table I summarizes the overall performance of the proposed algorithm. Percentage is defined as the ratio of the number of pixels of TPs to the actual number of pixels of corresponding glomeruli. The experimental results of the detection pipeline shows a precision of 78% and a recall of 79% in overall. Completeness of glomerulus segmentation can reach as high as 90%. Improved morphological closing is critical in annotating potential Bowman's capsule. In contrast, the undesired performance on CN8376 is primitively explained by its light color and limited texture information conveyed. The threshold in Algorithm 1 Line 2 for CN8376 is 0.886, whereas for the other four is 0.822 ± 0.025 . Moreover, glomeruli attached to the boundary of the tissues are hard to detect, since we eliminate the candidate objects with maximum length longer than 650 and the Bowman's capsule is discarded along with the boundary of the tissue.

VI. CLINICAL RESEARCH INFORMATICS

As part of the CRI toolbox first-order Hidden Markov Model (HMM) was being applied as a means of modeling temporal aspects of the post-surgery health state of nonhuman primates after undergoing isolated islet cell transplant

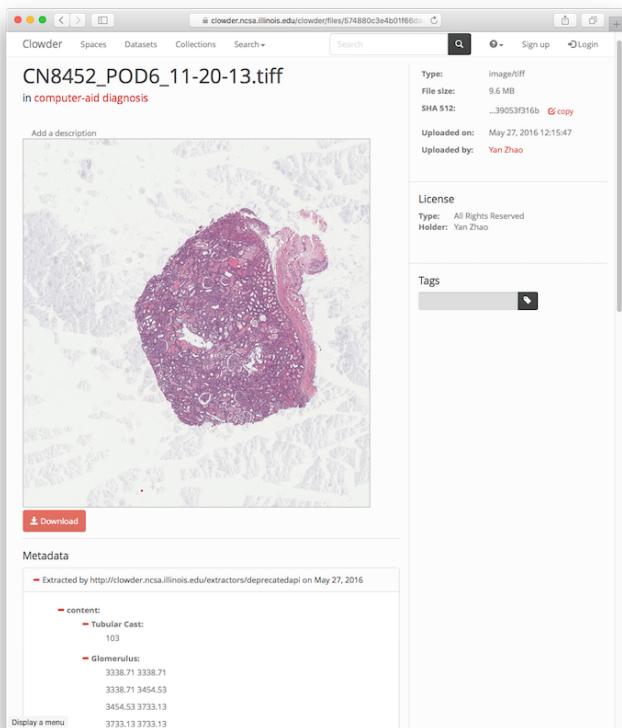


Fig. 12: Clowder interface of a single file with preview and project-specified metadata provided.

from healthy donor pancreas. The specific goal was to develop a tool capable of automatically determining the postoperative day (POD) corresponding to Rejection Free Survival, a sign of transplant rejection, on nonhuman primates after isolated islet cell transplant. The effectiveness of the tool was established by comparing its prediction of the date corresponding to Rejection Free Survival to the one determined by the collaborate experts based on their experience. This work was in support of research conducted to test the graft-promoting effects of mesenchymal stem cells (MSCs) in a cynomolgus monkey model of islet/bone marrow transplantation. Mesenchymal stem cells were used as a means of regulating immune response during organ and tissue transplant, theoretically reducing or even eliminating the need for immunosuppressants.

With the object of broad accessibility, visualizations and analysis of the project data, our system was integrated into a data management framework, which involved 1) Clowder [35], [36], an extensible web-based content management system to store the original Excel workbooks with raw data; 2) a collection of extractors to automatically extract information from the workbooks and ingest it into MongoDB; 3) a web application for large scale biomedical time-series data visualization and querying and 4) a series of machine learning algorithms with customized parameterization to study the postoperative day. Decision Tree and SVM were provided in conjunction with HMM as learning methods (Fig.11 and 12).

The work described here has been added to this toolbox as another tool in the form of an extractor that fires on uploaded biopsy images to identify/tag portions of the images with the trained classes allowing researchers to manage, navigate, and explore collections of such images (alongside other data sources, e.g. tabular subject data). Further, as extractors within the Clowder framework, these classifiers are also available^{7,8} as part of the Brown Dog data transformation service [37] for broader potential use among other scientific domains as well as archiving of the tools themselves.

VII. CONCLUSION

With the advent of digital scanner, postoperative prognosis is seeking assistance from pathological analysis of biopsy micrographs. We are convinced that the availability of fully automatic diagnosis framework is of immense benefit in leveraging the expertise and preventing graft loss.

In this work, we first discuss the possibility to learn and detect interstitial inflammation as a prelude to the development of interstitial fibrosis [38]; with increased breadth and sensitivity of interstitial fibrosis detected on whole slide imaging combined with molecular analyses it may be possible to predict the development of fibrosis, allowing the clinician to intervene at an earlier time point. To our best knowledge, this is the first study directed at automated renal allograft biopsy analysis. Besides formulating the two-round classification workflow that allows learning based on a relatively small amount of samples, we detail the proper feature selection and tile size for interstitial inflammation and tubular casts. With appropriate tuning strategies, the proposed framework achieves precision as 90% in average. In terms of computational efficiency, the two-round classification framework takes approximate 46.1s for an 112MB-pixel-foreground image, which allows routine clinical assessment of post-transplant renal biopsies.

As for glomerulus extraction, 110 out of 140 glomeruli from five WSIs are correctly extracted with average completeness over 90%. We are confident that our learning-oriented framework outperforms the existing algorithms in terms of accuracy and efficiency, and benefits to the subsequent measurement of renal slides. Efforts have been undertaken to annotate Bowman's capsule with insignificant thickness in this work, and further enhancement can be achieved from biological object segmentation based on texture information.

Nonhuman primate cynomolgus monkey kidneys are highly homologous to human kidneys [39] hence findings in this model can be directly translated to humans. The entire framework is integrated in Clowder as web service and demonstrated in CRI dataset⁹.

⁷<https://opensource.ncsa.illinois.edu/bitbucket/projects/CATS/repos/extractors-msc/browse/Kidney>

⁸<https://opensource.ncsa.illinois.edu/bitbucket/projects/CRI/repos/msc-kidney/browse>

⁹<https://clowder.ncsa.illinois.edu/clowder/datasets/57471bbe4b01f66da440fa0>

ACKNOWLEDGEMENT

Whole slides of renal images were obtained in Departments of Surgery and Bioengineering, University of Illinois at Chicago. This work has been supported by the National Institutes of Health under grant 1P01AI089556-01A1. Yan would like to thank Peishi Jiang for his help in MATLAB programming.

REFERENCES

- [1] W. W. Williams, D. Taheri, N. Tolkoff-Rubin, and R. B. Colvin, "Clinical role of the renal transplant biopsy," *Nature Reviews Nephrology*, vol. 8, no. 2, pp. 110–121, 2012.
- [2] V. Nickeleit, "Foretelling the future: Predicting graft outcome by evaluating kidney baseline transplant biopsies," *Journal of the American Society of Nephrology*, vol. 24, no. 11, pp. 1716–1719, 2013.
- [3] H. J. Wang, C. M. Kjellstrand, S. M. Cockfield, and K. Solez, "On the influence of sample size on the prognostic accuracy and reproducibility of renal transplant biopsy," *Nephrology Dialysis Transplantation*, vol. 13, no. 1, pp. 165–172, 1998.
- [4] S. Hara, "Banff 2013 update: Pearls and pitfalls in transplant renal pathology," *Nephrology*, vol. 20, no. S2, pp. 2–8, 2015.
- [5] B. Adam and M. Mengel, "Transplant biopsy beyond light microscopy," *BMC nephrology*, vol. 16, no. 1, p. 1, 2015.
- [6] M. Sarwal, M.-S. Chua, N. Kambham, S.-C. Hsieh, T. Satterwhite, M. Masek, and O. Salvatierra Jr, "Molecular heterogeneity in acute renal allograft rejection identified by DNA microarray profiling," *New England Journal of Medicine*, vol. 349, no. 2, pp. 125–138, 2003.
- [7] R. Bohra, J. Klepacki, J. Klawitter, J. Klawitter, J. M. Thurman, and U. Christians, "Proteomics and metabolomics in renal transplantation-quo vadis?" *Transplant International*, vol. 26, no. 3, pp. 225–241, 2013.
- [8] A. B. Farris, C. D. Adams, N. Brousaides, P. A. Della Pelle, A. B. Collins, E. Moradi, R. N. Smith, P. C. Grimm, and R. B. Colvin, "Morphometric and visual evaluation of fibrosis in renal biopsies," *Journal of the American Society of Nephrology*, vol. 22, no. 1, pp. 176–186, 2011.
- [9] J. Zhang and J. Hu, "Glomerulus extraction by optimizing the fitting curve," in *Computational Intelligence and Design, 2008. ISCID'08. International Symposium on*, vol. 2. IEEE, 2008, pp. 169–172.
- [10] K. Taras, D. Nilanjan, S. A. Amira, B.-T. Dana, C. Sayan, S. A. Ahmed, and M. T. Joo, "Measurement of glomerulus diameter and Bowman's space width of renal albino rats," *Computer Methods and Programs in Biomedicine*, vol. 126, pp. 143–153, apr 2016.
- [11] S. Samsi, W. N. Jarjour, and A. Krishnamurthy, "Glomeruli segmentation in H&E stained tissue using perceptual organization," in *Signal Processing in Medicine and Biology Symposium (SPMB), 2012 IEEE*. IEEE, 2012, pp. 1–5.
- [12] M. N. Gurcan, L. E. Boucheron, A. Can, A. Madabhushi, N. M. Rajpoot, and B. Yener, "Histopathological image analysis: a review," *Biomedical Engineering, IEEE Reviews in*, vol. 2, pp. 147–171, 2009.
- [13] T. J. Fuchs and J. M. Buhmann, "Computational pathology: Challenges and promises for tissue analysis," *Computerized Medical Imaging and Graphics*, vol. 35, no. 7, pp. 515–530, 2011.
- [14] T. J. Fuchs, T. Lange, P. J. Wild, H. Moch, and J. M. Buhmann, "Weakly supervised cell nuclei detection and segmentation on tissue microarrays of renal clear cell carcinoma," in *Pattern Recognition*. Springer, 2008, pp. 173–182.
- [15] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [16] C. Arteta, V. Lempitsky, J. A. Noble, and A. Zisserman, "Learning to detect cells using non-overlapping extremal regions," in *Medical image computing and computer-assisted intervention-MICCAI 2012*. Springer, 2012, pp. 348–356.
- [17] M. D. DiFranco, G. OHurley, E. W. Kay, R. W. G. Watson, and P. Cunningham, "Ensemble based system for whole-slide prostate cancer probability mapping using color texture features," *Computerized medical imaging and graphics*, vol. 35, no. 7, pp. 629–645, 2011.
- [18] T. Goudas, C. Doukas, A. Chatzioannou, and I. Maglogiannis, "A collaborative biomedical image-mining framework: Application on the image analysis of microscopic kidney biopsies," *Biomedical and Health Informatics, IEEE Journal of*, vol. 17, no. 1, pp. 82–91, 2013.
- [19] E. F. Black, L. Marini, A. Vaidya, D. Berman, M. Willman, D. Salomon, A. Bartholomew, N. Kenyon, and K. McHenry, "Using hidden Markov models to determine changes in subject data over time, studying the immunoregulatory effect of mesenchymal stem cells," in *e-Science (e-Science), 2014 IEEE 10th International Conference on*, vol. 1. IEEE, 2014, pp. 83–91.
- [20] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *The annals of mathematical statistics*, vol. 37, no. 6, pp. 1554–1563, 1966.
- [21] J. Ma, J. Zhang, and J. Hu, "Glomerulus extraction by using genetic algorithm for edge patching," in *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*. IEEE, 2009, pp. 2474–2479.
- [22] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.
- [23] T. Kato, R. Relator, H. Ngou, Y. Hirohashi, T. Kakimoto, and K. Okada, "New descriptor for glomerulus detection in kidney microscopy image," *arXiv preprint arXiv:1506.05920*, 2015.
- [24] F. Ghaznavi, A. Evans, A. Madabhushi, and M. Feldman, "Digital imaging in pathology: whole-slide imaging and beyond," *Annual Review of Pathology: Mechanisms of Disease*, vol. 8, pp. 331–359, 2013.
- [25] C. Papageorgiou and T. Poggio, "A trainable system for object detection," *International Journal of Computer Vision*, vol. 38, no. 1, pp. 15–33, 2000.
- [26] A. Homeyer, A. Schenk, J. Arlt, U. Dahmen, O. Dirsch, and H. K. Hahn, "Practical quantification of necrosis in histological whole-slide images," *Computerized Medical Imaging and Graphics*, vol. 37, no. 4, pp. 313–322, 2013.
- [27] K. V. Lemley and W. Kriz, "Anatomy of the renal interstitium," *Kidney international*, vol. 39, no. 3, pp. 370–381, 1991.
- [28] L. C. Racusen, K. Solez, R. B. Colvin, S. M. Bonsib, M. C. Castro, T. Cavallo, B. P. Croker, A. J. Demetris, C. B. Drachenberg, A. B. Fog et al., "The Banff 97 working classification of renal allograft pathology," *Kidney international*, vol. 55, no. 2, pp. 713–723, 1999.
- [29] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.
- [30] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.
- [31] I. Maglogiannis, H. Sarimveis, C. T. Kiranoudis, A. A. Chatzioannou, N. Oikonomou, and V. Aidinis, "Radial basis function neural networks classification for the recognition of idiopathic pulmonary fibrosis in microscopic images," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 12, no. 1, pp. 42–54, 2008.
- [32] R. M. Haralick, K. Shanmugam, and I. H. Dinstein, "Textural features for image classification," *Systems, Man and Cybernetics, IEEE Transactions on*, no. 6, pp. 610–621, 1973.
- [33] L.-K. Soh and C. Tsatsoulis, "Texture analysis of SAR sea ice imagery using gray level co-occurrence matrices," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 37, no. 2, pp. 780–795, 1999.
- [34] D. A. Clausi, "An analysis of co-occurrence texture statistics as a function of grey level quantization," *Canadian Journal of remote sensing*, vol. 28, no. 1, pp. 45–62, 2002.
- [35] J. Myers, M. Hedstrom, D. Akmon, S. Payette, B. A. Plale, I. Kouper, S. McCaulay, R. McDonald, I. Suriarachchi, A. Varadharaju et al., "Towards sustainable curation and preservation: The sead project's data services approach," in *e-Science (e-Science), 2015 IEEE 11th International Conference on*. IEEE, 2015, pp. 485–494.
- [36] L. Marini, R. Kooper, J. Futrelle, J. Plutchak, A. Craig, T. McLaren, and J. Myers, "Medici: A scalable multimedia environment for research," in *The Microsoft e-Science Workshop*, 2010.
- [37] S. Padhy, G. Jansen, J. Alameda, E. Black, L. Diesendruck, M. Dietze, P. Kumar, R. Kooper, J. Lee, R. Liu et al., "Brown dog: Leveraging everything towards autocuration," in *Big Data (Big Data), 2015 IEEE International Conference on*. IEEE, 2015, pp. 493–500.
- [38] X.-M. Meng, D. J. Nikolic-Paterson, and H. Y. Lan, "Inflammatory processes in renal fibrosis," *Nature Reviews Nephrology*, vol. 10, no. 9, pp. 493–503, 2014.
- [39] C. R. Abebe, K. Mansfield, S. D. Tardif, and T. Morris, *Nonhuman Primates in Biomedical Research, Two Volume Set*. Academic Press, 2012.

Improving Data Provenance Reconstruction via a Multi-Level Funneling Approach

Subha Vasudevan, William Pfeffer, Delmar Davis, Hazeline Asuncion

School of Science, Technology, Engineering, and Mathematics
 University of Washington Bothell
 Bothell, Washington, United States
 {subha, wpfeffer, davisdb1, hazeline}@uw.edu

Abstract— The ease with which data can be created, copied, modified, and deleted over the Internet has made it increasingly difficult to determine the source of web data. Data provenance, which provides information about the origin and lineage of a dataset, assists in determining its genuineness and trustworthiness. Several data provenance techniques record provenance when the data is created or modified. However, many existing datasets have no recorded provenance. Provenance Reconstruction techniques attempt to generate an approximate provenance in these datasets. Current reconstruction techniques require timing metadata to reconstruct provenance. In this paper, we improve our multi-funneling technique, which combines existing techniques, including topic modeling, longest common subsequence, and genetic algorithm to achieve higher accuracy in reconstructing provenance without requiring timing metadata. In addition, we introduce novel funnels that are customized to the provided datasets, which further boosts precision and recall rates. We evaluated our approach with various experiments and compare the results of our approach with existing techniques. Finally, we present lessons learned, including the applicability of our approach to other datasets.

Keywords— *data provenance; provenance reconstruction; Latent Dirichlet Allocation; Genetic Algorithm; Longest Common Subsequence; Statistical Re-clustering; Silhouette Coefficient*

I. INTRODUCTION

Data provenance assists in tracking the source of and changes to a dataset over the course of its lifetime. Provenance recording techniques enable provenance to be captured *while* data are created or modified, within provenance-enabled tools such as scientific workflows [7, 34, 20], databases [28], analysis tools [8, 30], or environments such as operating systems [26], distributed environments [32], or monitoring environments [21]. However, many current datasets exist for which provenance was not originally captured (e.g., datasets available on the Internet). These types of datasets are out of scope for the aforementioned tools and environments.

To address these challenges, provenance reconstruction techniques have been proposed to generate *approximate* provenance from available data. Since the exact provenance is not captured, one can simply approximate what may have occurred. In a previous work, we generated an approximate provenance based on knowledge of existing research practices

and conventions [14]. In cases when this knowledge is not available, we started examining the semantic content of the files to reconstruct provenance [6].

Provenance reconstruction techniques are applicable for the following scenarios: (1) Researchers working on multiple projects may not have time to explicitly label their notes and/or data. Provenance reconstruction can assist in determining which notes and data belong to a project and in automatically grouping together the related notes and data. (2) When software companies acquire other software companies, they are left with miscellaneous files (e.g., developer notes, test notes) that are not necessarily saved in a version control system. In addition, if some of the software engineers leave the company, it becomes difficult to determine the files related to a software product. Provenance reconstruction can assist in grouping these related files together as well as connecting them to a software product.

In general, current provenance reconstruction techniques measure similarities between files. One technique uses several similarity measures (e.g., text similarity, image similarity, domain-specific similarity, metadata similarity) and edit distance algorithms [31]. Another technique measures similarity between files through named entities represented in a Vector Space Model [16]. The most closely related work uses Vector Space Modeling with cosine similarity and TF-IDF weights [35]. All these techniques assume that timing information (i.e., date of creation) is available for all the files. Our technique, meanwhile, does not require the presence of timing information. The only metadata our technique uses, for one of the datasets, are the file type and file size.

Our provenance reconstruction technique focuses on reconstructing provenance on two types of datasets similar to those provided by the 2014 IPAW Provenance Reconstruction Challenge: human-generated and machine-generated datasets [2]. The human-generated dataset is a collection of Wikinews articles [5] and source files with all association information deleted. The reconstruction task is to identify “had Primary Source” links between the two sets of files. The machine-generated dataset contains all files that were maintained in a version control system, with all version history removed. The goal is to establish “was Derived From” relation between files.

To reconstruct provenance for both types of datasets, we pursue the multi-level funneling approach we previously introduced [6]. This funneling approach is distinctive since we measure the similarities between files using their semantic content and we leverage state-of-the-art techniques from areas such as machine learning. We also introduce novel funnel techniques, such as statistical re-clustering and file clustering to determine provenance relationship between files. Finally, we demonstrate that our approach achieves higher precision and recall rates than existing techniques and is applicable to other datasets.

The paper is organized as follows. Next section covers works related to our approach. Section 3 covers our approach. We present our evaluation results in Section 4. Section 5 contains lessons learned and we conclude with future work.

II. RELATED WORK

Generating an *approximate* provenance for datasets for which provenance was not originally captured, i.e., provenance reconstruction, is a fairly new area of study in the provenance community [35]. Provenance reconstruction is especially applicable in the Web environment where much data already exist but the provenance has not been recorded. Since the generated provenance is approximate, there is some level of uncertainty [15]. Use cases and requirements for provenance on the Web have also been presented by the provenance community [22].

There are attempts to reconstruct provenance of different types of datasets. Table I shows a summary of techniques proposed within the data provenance community, which have reported results in terms of precision and recall. The table also shows types of data and the use case for the provenance reconstruction.

One approach makes use of signal detectors, signal filters, and aggregators. The signal detectors analyze different types of data such as text, metadata, images, or domain-specific information. The signal detectors then determine similarities between each type using existing techniques, such as Lucene [1] to determine text similarity. The signal filters filter dependencies based on a criteria (e.g., time of document creation or a threshold). Finally, an aggregator provides weighting to the scores provided by the signal detectors [31].

Another approach aims to restore provenance based on semantic content [16]. Similarity between files is determined using named entities as words in a Vector Space Model.

The closest related work is by De Nies et al. who have also investigated reconstructing provenance in human-generated datasets [35]. The method proposed by the authors clusters the documents and then uses the assumption that the oldest document is the primary source of all other documents. The approach produced results with 86% precision and 59% recall. Since the approach tries to solve the same problem addressed in this paper, these values can be used for direct comparison with the results in this paper (see Section IV.A.4).

While these techniques are sound, they rely on metadata (such as timing metadata), which are not always present. Our

technique for human datasets only requires the semantic content to be present within HTML pages to reconstruct missing provenance without any metadata being present. In processing machine-generated dataset, however, we do rely on basic metadata, file size and file type. We posit that these metadata are always available.

Other techniques for provenance reconstruction include metadata annotation. One technique provides metadata annotation on datasets via artificial intelligence (AI) planning and edit distance to determine the sequence of changes on the original dataset [23]. Another technique annotates historical datasets with entries from a domain ontology, which are then used to semantically associate subsets of ontological graphs [39].

Provenance reconstruction has also been used for various applications such as assessing the genuineness of information on social media [17, 37], determining the provenance of tuple data [38], or in analyzing strains of malware code [19].

TABLE I. PROVENANCE RECONSTRUCTION TECHNIQUES WITH REPORTED RESULTS

Data and Use Case	Techniques	Results
Deriving source articles of Wikinews articles (i.e., human-generated dataset in the IPAW Provenance Reconstruction Challenge) [35]	Vector Space Model and cosine similarity with TF-IDF weighting	Precision: 86% Recall: 59%
Reconstruct provenance for documents in shared folder setting [31]	Signal detectors, signal filters, and aggregators	Precision: 63% Recall: 80%
Determine primary sources of news articles [16]	Vector Space Model & Named Entity Recognition	Precision: 68.2% Recall: 73%

III. PROVENANCE RECONSTRUCTION TECHNIQUE

In order to reconstruct provenance for these two types of datasets [2], we used a multi-level funneling approach, which we introduced in [6]. We provide an overview of our approach, present the funnels we used, and discuss how we combine the funnels to form funnel sets.

A. Overview

We refer to techniques that determine some type of relationship between files in a dataset as funnels. Because the datasets we examine in this paper have semantic content, we selected funnels that analyze this semantic content, to determine “had Primary Source” and “was Derived From” relationships, as defined in the W3C PROV [4].

Prior to applying any funnels, the entire dataset is pre-processed. This pre-processing includes obtaining all relevant files and removing any unnecessary contents, such as punctuations, tags, and stop words.

Funnel sets are applied to datasets in an increasing level of detail (i.e., from coarse- to fine-grained relationships), as shown in Fig. 1, in a multi-level fashion. First, course-grained

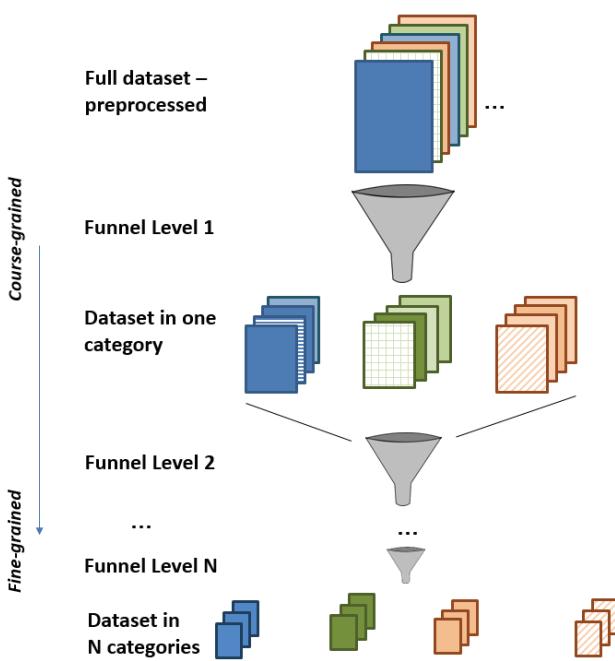


Fig. 1. Multi-level Funnel Approach

funnels are applied to the entire dataset to categorize the data in one dimension. Another funnel is then applied to each category, iteratively, to find finer-grained relationships between files. This process is repeated until provenance relationships such as “had Primary Source” or “is Derived from” can be inferred within each group of files that are created.

B. Funnels

In this section, we discuss the different funnels we used and the rationale for selecting them.

1) Latent Dirichlet Allocation (LDA)

Topic modeling is a set of algorithms that are capable of extracting abstract topics from a group of documents and clustering the documents based on those topics. Topic Models are used to discover the main themes that are spread across a large and otherwise unstructured data [12]. It is highly useful in organizing and indexing huge electronic archives, where human annotation is impossible.

The most commonly used algorithm in topic modeling is Latent Dirichlet Association (LDA). The basic assumption of LDA is that a document can be a part of multiple topics. A topic is modeled as a distribution over words. A document is modeled as a distribution over topics. LDA has been popularly used in various applications such as web page tagging [29], text image and music classification [27], spam filtering [11], email classification [36] and others.

Rationale: Similarities between files based on their textual content can be detected using information retrieval techniques such as the Vector Space Model (VSM) [33] and Latent Semantic Indexing (LSI) [18]. In both VSM and LSI, each document in the corpus is represented as a word count vector

of length W , where W is the number of words in the corpus vocabulary. (Note: other weights may also be used aside from counts, such as TF-IDF [33].) When the vectors of all D documents are placed side by side, one obtains a $W \times D$ matrix of counts. LSI addresses a shortcoming of VSM, which is the ability to handle synonyms and polysemes [33]. Probabilistic LSI (PLSI) is a version of LSI which is able to achieve better results than LSI [25]. Meanwhile, Latent Dirichlet Allocation (LDA) is a fully generative Bayesian model [12] which addresses overfitting issues associated with PLSI. Since LDA addresses shortcomings of previous text processing techniques, we selected LDA as one of our funnels.

Usage: LDA is an unsupervised machine learning technique; thus, no training data is required. It only requires the dataset to analyze and the number of topics to be learned. In some implementations, the number of iterations on which the algorithm will run is also required.

2) Latent Dirichlet Allocation with Genetic Algorithm (LDA-GA)

Genetic Algorithm (GA) is an optimization algorithm resembling the process of natural selection [24]. GA works towards finding the most optimal solution for problems that do not have a definite answer such as NP hard problems. The algorithm starts with an initial population. Each individual in the initial population is a chromosome. Each chromosome is evaluated for its fitness. The fitness of a chromosome is determined based on conditions that are unique to every application. The chromosomes with the best fitness value are made a part of the next generation. The other chromosomes in the new generation are called offsprings. Offsprings are constructed by performing crossover and mutation on the fittest chromosome of the previous generation. The process of evaluating fitness is repeated for the new generation. The cycle repeats itself until certain conditions are met or the algorithm runs for a predefined number of iterations.

Rationale: Since the parameters used in LDA (number of topics and number of iterations) could significantly impact the accuracy of the document categorization, we use GA to dynamically determine the optimal number of topics and iterations for each dataset. In contrast, a systematic sweep technique to find the optimal parameters will include weak inputs and, thus, will be more time consuming. GA eliminates weak inputs progressively and stops when thresholds are met.

Usage: The initial population of the genetic algorithm starts with k chromosome with two genes or characteristics in each. The first gene represents the number of topics and the second gene represents the number of iterations. These two values are randomly populated for all the k chromosomes. Each of these k chromosomes is used as an input configuration of LDA. LDA runs once with the number of topics and the number of iterations from each of the chromosomes. The LDA output for each chromosome is a topic distribution file, which contains a list of documents and its topic distribution. This distribution is converted into a document topic matrix. The documents are virtually plotted in an n dimensional space where n is the number of topics. The documents with a higher percentage of a certain topic appear closer to each other in the plot. They form a virtual cluster.

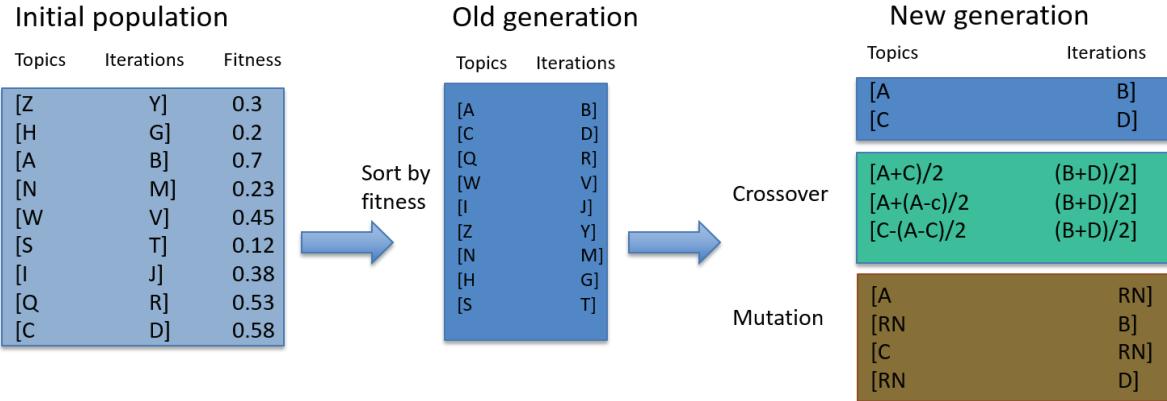


Fig. 2. Genetic Algorithms: Sorting, Crossover, and Mutation

As mentioned previously, GA uses a *fitness function* to determine whether it has reached the optimal number of topics. Silhouette Coefficient is used to determine the quality of the clusters that are formed [10]. Silhouette coefficient calculates the distance between documents using the Euclidian distance:

$$Sil(i) = \frac{(b(i)-a(i))}{\max(b(i),a(i))} \quad (1)$$

where,

$a(i)$ – average dissimilarity between i and the other documents of the cluster to which i belongs.

$D(i,C)$ – For all other clusters C , average dissimilarity of i to all observations of C .

$b(i)$ - minimum of $D(i,C)$ – dissimilarity between i and the neighbor cluster C .

Silhouette coefficient is a numerical value ranging between -1 and 1. This numerical value indicates how close a document is to other documents in its cluster and how far it is from the center of other clusters. The higher the Silhouette Coefficient, the better coupling between documents in a cluster leading to higher quality of clusters.

The fitness of a configuration is determined by calculating the average of the Silhouette coefficient of all the documents. In order to select chromosomes with only good Silhouette Coefficient, a quality threshold is set. The quality threshold ranges are the same as that of the Silhouette Coefficient. The higher the quality threshold, the more iterations the algorithm is likely to run to meet the threshold. The Silhouette coefficient of all the k chromosomes is compared with the expected quality. If the Silhouette Coefficient of one of the chromosomes surpasses the quality threshold, the genetic algorithm loop exits and the specific chromosome is identified as the optimal configuration.

If none of the chromosomes' Silhouette Coefficient meets the quality threshold, then the chromosomes are sorted based on their fitness value. The first $k/4$ chromosomes with the best fitness values are added to the new generation. The rest of the

new generation is populated using crossover and mutation functions, as shown in Fig. 2, where $k = 9$.

Crossover and mutation are steps in GA that attempt to find a better solution. When a good configuration is identified, crossover searches for a better configuration close to it and $k/3$ new configurations are created. To minimize the chance of the genetic algorithm staying in a local optimum, mutation introduces new random numbers to make sure that $k/2$ other configurations are also considered.

GA runs again using the new generation of chromosomes. The process continues until one of the chromosomes reaches the threshold fitness or until GA finishes a predefined number of iterations, which we set to i . If it is the latter case, the chromosome with the best fitness discovered throughout all the iterations is considered as the best configuration. The Silhouette coefficient of each configuration is saved in a hashmap for later reuse.

3) Statistical Re-clustering (SR)

Statistical Re-clustering is a novel funnel technique we used specifically for the human-generated datasets. Due to the nature of human datasets, it is necessary to associate each Wikinews article with its source files. Thus, the dataset must be grouped such that each group has only one Wikinews article and one or more source files.

Rationale: Even though the LDA-GA is capable of determining the optimal configuration (number of topics and iterations), this funnel is not aware whether a specific document is an article or a source file. Hence, the output from LDA-GA funnel may be one of three types of clusters (see Fig. 3). Clusters of type A have one Wikinews article and zero or more source files, clusters of type B have more than one article and one or more source files, and clusters of type C have all source files and no articles.

Usage: To determine the “had Primary Source” PROV relation, SR checks for these three types of cluster. SR considers Type A Cluster as a perfect cluster because one can conclude that the one Wikinews article in the cluster is derived from n source articles that are in the cluster. Thus, no additional processing is necessary. Meanwhile, one is not able

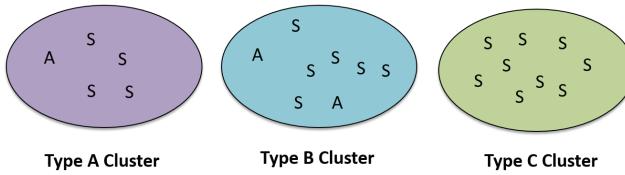


Fig. 3. Types of Clusters

to arrive at this conclusion for Type B Cluster which contains n Wikinews articles. SR processes this cluster by creating n new clusters which only contains one Wikinews article in each cluster, as shown in Fig. 4. Each word from a source file is matched with the words of each article. The source file is assigned to the cluster which has the maximum percentage match with the Wikinews article. Hence a big cluster of Type B is converted to multiple smaller clusters of Type A. Finally, SR processes clusters of Type C by matching each source file with articles in each Type A Cluster, and assigning these files to the cluster with the highest percentage match. Once SR is complete, all clusters are of Type A, where the article in a cluster has a “had Primary Source” relationship with the source files in the cluster.

4) Longest Common Subsequence

The longest common subsequence (LCS) problem finds the longest subsequence common to all sequences in a set of sequences. Note that a subsequence is different from a substring, for the terms of the former need not be consecutive terms of the original sequence.

Rationale: Another way to determine relationships between files is by checking the order of occurrences of the text within files. The LCS algorithm has been used in comparing two different versions of a file (“diff”). Thus, the LCS algorithm fits the reconstruction task from the Provenance Challenge well since we are trying to determine whether a file came from another file.

Usage: In our approach, we use the atomic unit of the LCS algorithm. In other words, we find the longest common word sequence, whose order is the same in both files. For example, say that we will find the LCS between file A and file B. There are n words in file A, respectively A_1, A_2, \dots, A_n and m words

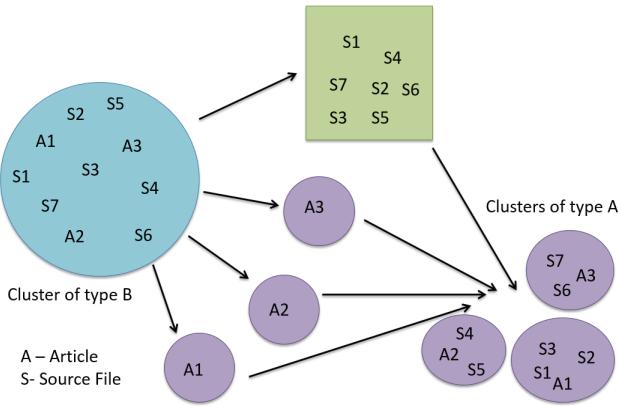


Fig. 4. Statistical Re-clustering to create Type A Clusters

in file B, respectively B_1, B_2, \dots, B_m . The algorithm will find the largest size s and two S -length increasing sequence $1 \leq a_1 < a_2 < \dots < a_{s-1} < a_s$, $1 \leq b_1 < b_2 < \dots < b_{s-1} < b_s$, so that for each i in range $[1, s]$, $A_{a_i} = B_{b_i}$.

The LCS problem is solved by dynamic programming [9] because of the properties of overlapping subproblems and optimal substructure. We use $LCS(n, m)$ to represent the size of the longest common subsequence of file A and file B.

To calculate the similarity, we use the following metric:

$$\text{Similarity} = (\text{number of words in the longest common subsequence of two files}) / (\text{the total number of words in smaller file}) \quad (2)$$

The similarity value will be between 0 and 1 inclusive. If there is nothing in common between the two files, the value will be 0; if the smaller sized file is completely a part of the bigger sized file, the value will be 1.

5) File Clustering

File clustering is another novel funnel technique we created specifically for the machine-generated dataset. Due to the nature of machine-generated dataset, each group should only contain the different versions of one file. Thus, this funnel groups files by their file types.

Rationale: The other funnels we selected are agnostic to the type of file analyzed. Thus, it is necessary to have a funnel that clusters by file type.

Usage: This funnel categorizes together all files with the same type. In addition, this funnel checks for file sizes to determine the order of versions of the same file. Our technique assumes that smaller sized files are earlier versions of larger sized files. The larger files are assigned a “was Derived From” relationship with the immediate smaller sized file.

C. Funnel Sets

In this section, we discuss how we form funnel sets from the previously mentioned funnels. Table II shows a summary of the funnel sets we used.

TABLE II. SUMMARY OF FUNNEL SETS AND DATASETS

Funnel Set	Dataset
LDA + LCS	Human
LDA + SR	Human
LDA-GA + SR	Human
LDA + FC	Machine

1) Funnel Set #1 - Human: LDA+LCS

In this funnel set, LDA is the first funnel followed by LCS. Since sources are directly mapped to articles based on topics, there must be at least as many topics as the number of articles. For this funnel set, we set the number of topics to twice the number of articles. Based on past experience, topics generally converged before 2000 iterations. Thus, we used this number as the number of iterations for all our LDA runs.

After LDA categorizes the documents, LCS identifies the longest sequential order of words that are present in two documents. This step allows us to identify very closely related files, enabling us to connect the Wikinews articles with the source article(s). Because LCS considers the order of words and because of the diversity of news articles, LCS is able to create clusters of type A without using an additional funnel like SR.

2) Funnel Set #2 - Human: LDA + SR

In this funnel set, LDA is the first funnel followed by SR, to ensure that each Wikinews article is mapped to one or more source articles. The parameters for LDA were set similar to that of the LDA+LCS funnel set. Hence the number of topics is twice the number of files and the number of iterations is 2000.

3) Funnel Set #3 - Human: LDA-GA + SR

In this funnel set, we use the hybrid LDA-GA funnel as the first funnel. GA automatically determines the best number of topics and number of iterations. The parameters for GA (k and i), meanwhile, are determined by the type of hardware used. For example, experiments we discuss in this paper were conducted on a machine with 4 cores (Intel Core i5) and 8 GB of memory (DDR3). We determined that $k = 9$ and $i = 50$ would be ideal for this machine's configuration. The hybrid algorithm would run for 90 minutes in the worst-case scenario.

LDA is executed once more for the best configuration and clusters are created for each topic. Each cluster has a set of keywords that are relevant to the topic.

Once LDA-GA is completed, SR is run to ensure each Wikinews article is mapped to one or more source articles.

4) Funnel Set #4 - Machine: LDA + FC

For the machine-generated dataset, we used the LDA funnel to cluster them based on the semantic content of the files. Since the number of files here could be large, we simply fixed the number of topics to 100. Once the files are clustered according to topics, then each topic cluster is fed into the file clustering funnel. Thus, we can determine the file derivations.

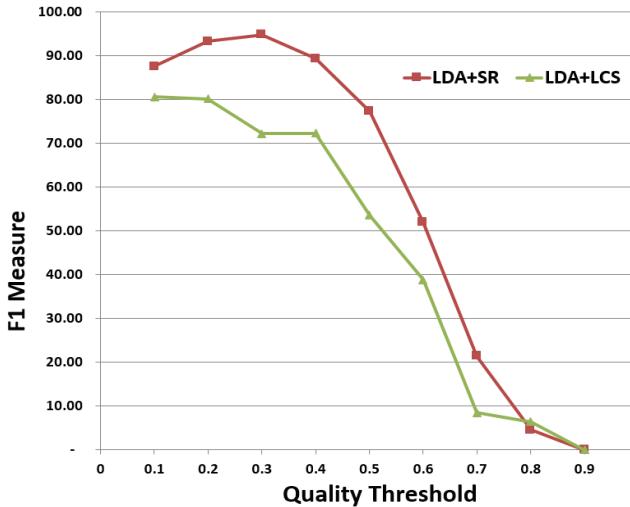


Fig. 5. Comparison of two funnels across various quality thresholds

IV. EVALUATION

We conducted experiments (1) to determine the accuracy of our funnel sets across different quality threshold and data sizes, (2) to compare our results with an existing technique, and (3) to measure execution times across different dataset sizes. Since the ground truth file is available, we are able to calculate precision and recall. We also use the F1 measure to equally weigh the precision and recall values. Briefly, these values are calculated as follows [33]:

$$\text{Precision} = \frac{\text{Number of relevant items retrieved}}{\text{Number of Retrieved items}} \quad (3)$$

$$\text{Recall} = \frac{\text{Number of relevant items retrieved}}{\text{Number of relevant items}} \quad (4)$$

$$F1 \text{ measure} = (2 * \text{precision} * \text{recall}) / (\text{precision} + \text{recall}) \quad (5)$$

In addition to determining the general applicability of our funnels, we generated additional datasets similar to those provided by the IPA Provenance Reconstruction Challenge [2]. These datasets are available here [3].

A. Evaluating Human-Generated Dataset

1) Dataset Generation and Pre-processing:

We generate our own dataset by randomly scraping the Wikinews websites [5] for news articles. The Wikinews articles are downloaded as HTML files. Source article files are also downloaded. A ground truth file is created to save the mapping between the articles and the sources.

Prior to applying any funnels, we pre-process the generated dataset. The HTML files are converted into text files, all HTML tags are removed, and the files are saved in a single directory. The references of Wikinews articles are also removed. Finally, stop words are removed from the text files.

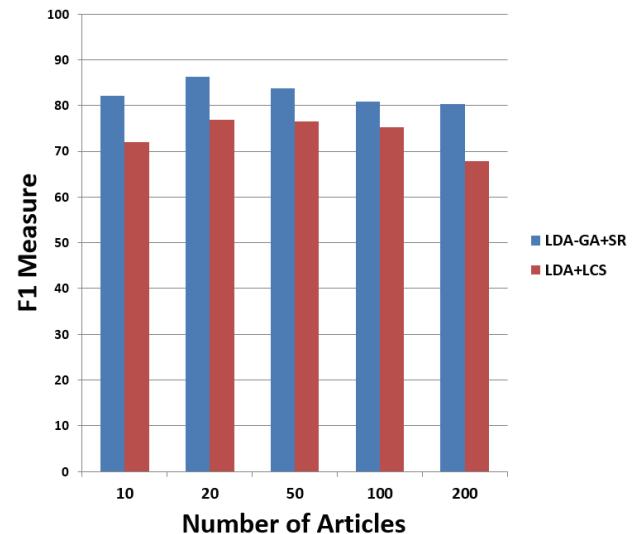


Fig. 6. Comparison of two funnels across different number of Wikinews articles

2) Experiment #1: Different Quality Thresholds

The purpose of our first evaluation is to determine how well our funnel sets perform at different quality thresholds. For this evaluation, we compared the F1 measure of three funnel sets: topic modeling with longest common subsequence (LDA+LCS), topic modeling with statistical re-clustering (LDA+SR), and topic modeling with genetic algorithm and statistical re-clustering (LDA-GA+SR). We ran these funnels on 30 Wikinews articles where the ground truth was withheld. After processing, we compared the output with the ground truth to calculate precision, recall, and F1 measure.

The quality thresholds are as follows. In LDA-GA+SR the quality threshold is the condition that terminates the Genetic Algorithm, which is the Silhouette Coefficient. For LDA+LCS and LDA+SR, the quality threshold is a factor that determines whether a document belongs to the cluster of a certain topic or not. If the distribution of a document to a specific topic is higher than the quality threshold, then the document is assumed to belong to the cluster of that specific topic.

Since the quality thresholds for LDA+LCS and LDA+SR are similar, we plotted their F1 measures (see Fig. 5). As the graph shows, LDA+SR (red line) performs slightly better than LDA+LCS (green line), with both dropping off at 0.9 quality threshold. This is reasonable, since a very high threshold, 0.9, is used when creating the topic clusters. In this case, recall is, or is very close to, 0. In comparison, the LDA-GA+SR funnel has an average F1 measure of 89.93 across all quality thresholds, in comparison to LDA+SR average F1 measure of 57.80 and LDA+LCS of 45.86. Based on this result, we conclude that LDA-GA+SR funnel set is able to achieve much higher precision and recall rates than LDA+SR or LDA+LCS.

3) Experiment #2: Different Data Sizes

The purpose of the second evaluation is to determine how well the funnels perform across different dataset sizes. For this evaluation, we compared two funnel sets: LDA-GA+SR and LDA+LCS. Because the genetic algorithm can result in varying results for the same dataset, we ran the LDA-GA+SR three times on the same dataset and took their average. We use the same datasets for each funnel set.

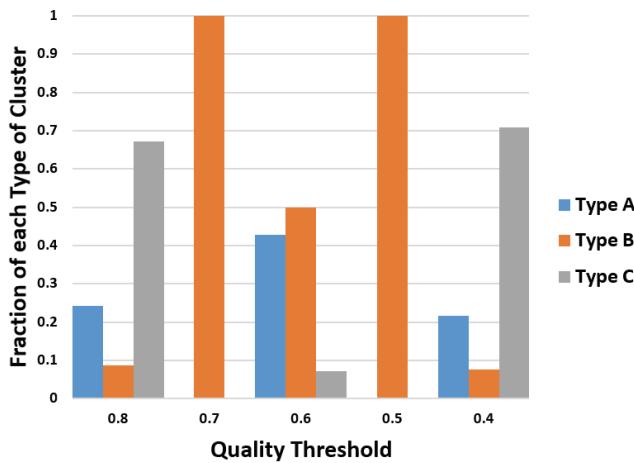


Fig. 7. Cluster type distributions after running LDA-GA funnel on IPAW human dataset

Fig. 6 shows that LDA-GA+SR funnel set (blue bar) consistently performs better than the LDA+LCS funnel set (red bar). It also shows that both funnels slightly drop in F1 measure for larger number of articles (100 and 200).

4) Experiment #3: Comparison with Related Work

As mentioned in Section 2, the work by De Nies et al. is the closest related work to this paper since they also seek to determine the “had Primary Source” relationships for the human-generated dataset [35]. They reported that they were able to achieve a precision of 86% and recall of 59%. Since we have demonstrated in the previous experiments that LDA-GA + SR funnel set outperforms the other funnel sets we created, we ran this funnel set against the human-generated dataset provided by the provenance challenge [2]. Table III shows our results across different quality thresholds (QT). Since the IPAW dataset had only 20 articles, there is not much difference in the precision and recall values across QTs. The gradual decrease in precision and recall values with the increase in QT is more prominent in larger datasets.

To illustrate how SR assists with achieving these precision and recall rates, Fig. 7 shows a distribution of the three types of clusters after running LDA-GA funnel across various QTs. This graph shows that the cluster distributions are based on the best configuration that meets the given quality threshold. Statistical re-clustering restructures the distributed cluster types to clusters of type A.

5) Experiment #4: Execution Times

Fig. 8 depicts the maximum execution times for datasets ranging from 250 to 2708 files. The size of each dataset is the sum of the number of articles and source files. QT was set at 0.8. The execution time is based on one of the following scenarios: (1) the length of time for reaching the best configuration, or (2) the length of time processing the entire size of the dataset (in case the best configuration is not reached by the predefined number of iterations, in this case, 50). Most of the execution times depicted in the graph are of scenario 2 because of the high QT. One of the exceptions is the execution time for dataset size 1352, which is less than the execution time of dataset size 1078. LDA-GA reached a configuration that produced clusters meeting the QT before the predefined number of iterations.

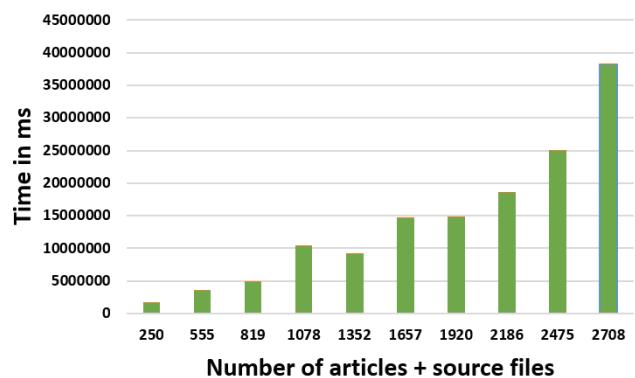


Fig. 8. LDA-GA+SR execution time on human-generated dataset

B. Evaluating the Machine-Generated Dataset

1) Dataset Generation and Pre-processing:

The second set of data is version controlled documents, which is referred to as the machine dataset. The Provenance Reconstruction Challenge provided a dataset, which contains 7,723 files from various GitHub repositories [2]. In addition to this provided dataset, we also extracted our own machine-generated datasets, which were gathered directly from popular GitHub projects. Git repositories store commit objects that reference the commits that came immediately before them. We first traverse through these Git repositories and make a copy of every version of a file that ever existed. In an attempt to make it more difficult to link together the data lineage, file names are also randomized, similar to the challenge dataset.

Like the human-generated set, pre-processing is applied to every file that has been gathered. Unlike the human-generated dataset, the machine-generated dataset contains many file types. All non-text based documents are removed as part of pre-processing. This prevents files, such as images, from being processed, since these do not have any easily readable semantic content stored in text within them.

2) Experiment: Large Datasets

For the machine-generated datasets, we calculated the precision and recall of the following funnel set: topic modeling and file clustering (LDA+FC). The purpose of this evaluation is to determine how well this funnel set performs on large-scale data (up to 100K+ articles). In addition, since we are currently not able to scale our other funnels, we are unable to compare this funnel set with another funnel set.

Fig. 9 shows that while LDA+FC funnel is able to process 100K+ articles, the F1 measure decreases with larger dataset sizes. This may be due to the topic numbers being pre-determined based on the number of files, and not on the optimal number of topics, as the GA funnel is able to do. Also, it may be that this pre-determined topic number is better suited to the dataset with 107K files than the smaller datasets.

V. LESSONS LEARNED

The following are lessons learned based on our experience of using funnels to reconstruct provenance.

1) General purpose and customized funnels:

General purpose funnels, such as the techniques borrowed from machine learning, are useful in that they can be applied generally to other similar types of datasets and they are able to reconstruct some provenance. A drawback to these techniques, however, is that there is a limit to the level of accuracy they can achieve. Thus, in order to boost precision and recall, customized funnels, which are infused with specific knowledge about the structure of the dataset (e.g., one Wikinews articles can have multiple sources), are needed. For both the human-generated and machine-generated dataset, we used funnel sets that contain both general purpose funnels (e.g., LDA) and customized funnels (e.g., Statistical Re-clustering for human-generated dataset, File Clustering for machine-generated dataset).

TABLE III. LDA-GA+SR ON IPA'W'S HUMAN-GENERATED DATASET [2]

QT	Precision	Recall
0.5	91.61	69.40
0.6	91.07	71.37
0.7	88.98	67.37
0.8	88.50	69.53

2) Order of funnels in a funnel set:

It is also important to determine the correct order of funnels in a funnel set. As we mentioned, funnels are ordered based on the level of granularity of provenance relationships they can detect. For both the human and machine-generated datasets, we determined that identifying relationships of files based on topics is a coarse-grained relationship. Thus, LDA was used as a first-level funnel. Meanwhile, customized funnels, which have specific knowledge about a dataset, are able to detect fine-grained provenance relationships: "had Primary Source" or "was Derived From". Thus, these funnels are applied last.

3) Complimentary techniques:

General purpose funnels that have limitations should be accompanied with other funnels that overcome those limitations. For example, VSM represents documents as a vector of word counts (or inverse document frequencies). However, a VSM may represent strings with the same root as two different words (e.g., reads, reading). Thus, prior to using a VSM funnel, a natural language processing funnel, such as stemmers, should be applied on the dataset first to group together words with the same stem as one word in VSM.

Funnels could also be composed of a hybrid of techniques to overcome limitations. For example, LDA-GA is a hybrid funnel that we used on the human-generated dataset. Because the results of LDA are highly subject to the number of topics selected, GA enables the automatic selection of an optimal number of topics. Once the optimal number of topic is selected, LDA is run one more time to obtain the topic relationship between files.

4) Necessity for scalability:

While we are able to achieve higher accuracy than previous techniques, the funnels we use are computationally expensive (see Fig. 8). Thus, we have been unable to apply some of the funnels (e.g., GA and LCS) to large datasets (i.e., machine-generated dataset). We are currently examining running the funnels in a parallel environment to address this issue.

5) General applicability of the funnels:

The general purpose funnels discussed in this paper are well suited to process unstructured data, such as news articles. Thus, they will work well with other unstructured data. Reconstructing source files or file derivations of structured data may require additional metadata, such as timing information, and may benefit from techniques provided by the database community [28, 38].

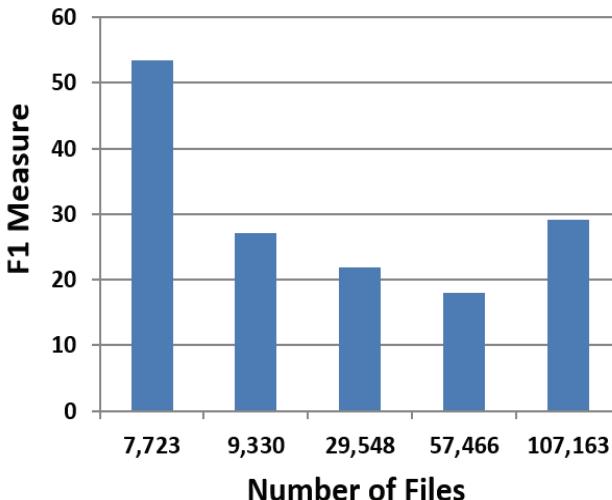


Fig. 9. Performance of LDA + FC on machine-generated dataset

Source code files could be considered semi-structured data, with semantic information limited to variable names, method names, class names, and comments. For these types of files, we also found that additional pre-processing is required prior to using the general purpose funnels [13].

VI. CONCLUSION

Reconstructing provenance is necessary in cases when a dataset has already been processed in the past with no recorded provenance. In this paper, we have demonstrated that a multi-level funneling approach is effective in connecting Wikinews articles with their sources (human-generated dataset) and in connecting files with different versions (machine-generated dataset). Our funnels strongly rely on the semantic content of the files for both datasets. For the machine-generated dataset, we use the additional information regarding file type and file size, while other existing techniques rely on additional metadata. Our experiments demonstrate that our technique can reconstruct a majority of the provenance for the human dataset and can process large numbers of files for the machine-generated dataset. Our experiments also show higher precision and recall rates for the human-generated dataset compared to an existing technique. Finally, we discussed lesson learned in using funnels for reconstructing provenance.

In the future, we plan to address the limitations of the current implementation of our technique. To overcome scalability issues, we plan to parallelize our technique over multiple computing nodes. Along with the number of topics and iterations, we plan to optimize the Dirichlet parameters with GA. In addition, we plan to use GA and LCS in reconstructing provenance for the machine-generated datasets to improve accuracy. We also plan to use other general purpose funnels such as natural language processing techniques. We also plan to further assess our techniques on other types of data.

ACKNOWLEDGMENT

The authors wish to thank Qi Zhang, Ailifan Aierken, and all the students in the Spring 2014 CSS590 Graduate Data Provenance course, for their initial work on the provenance reconstruction funnels. We thank Arthur U. Asuncion for his insights on LDA and providing the CVB0 implementation of LDA. We also wish to thank Hoa Vo for running various experiments and Xiaolin Ma for his help in initial parallelization work of the Genetic Algorithm. The authors thank Dr. Min Chen and Dr. William Erdly for their insights and feedback. This work is based in part upon work supported by the US National Science Foundation under Grant No. ACI 1350724. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

REFERENCE

- [1] Apache lucene. <https://lucene.apache.org/>.
- [2] Provenance reconstruction challenge 2014. <http://www.data2semantics.org/prov-reconstruction-challenge/>.
- [3] Provenance reconstruction wiki. <http://www.provenancereconstruction.org/>.
- [4] W3C PROV. <https://www.w3.org/TR/prov-overview/>.
- [5] Wikinews. <https://www.wikinews.org/>.
- [6] Ailifan Aierken, Delmar B. Davis, Qi Zhang, Kriti Gupta, Alex Wong, and Hazeline U. Asuncion. A multi-level funneling approach to data provenance reconstruction. In *Proceedings of the 2014 IEEE 10th International Conference on e-Science - Volume 02*, E-SCIENCE '14, pages 71–74, Washington, DC, USA, 2014. IEEE Computer Society.
- [7] Ilkay Altintas, Oscar Barney, and Efrat Jaeger-Frank. Provenance collection support in the Kepler Scientific Workflow System. In *Proc of the Int'l Provenance and Annotation Workshop (IPAW)*, Chicago, 2006.
- [8] Hazeline U. Asuncion. *In Situ* data provenance capture in spreadsheets. In *Proc of the 7th Int'l Conference on e-Science*, 2011.
- [9] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.
- [10] Pavel Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.
- [11] István Bíró, Jácint Szabó, and András A. Benczúr. Latent dirichlet allocation in web spam filtering. In *Proc of the 4th Int'l Workshop on Adversarial Information Retrieval on the Web*, pages 29–32, 2008.
- [12] David M. Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, April 2012.
- [13] Namita Dave, Karen Potts, Vu Dinh, and Hazeline U. Asuncion. Combining association mining with topic modeling to discover more file relationships. *International Journal on Advances in Software*, 7(3&4), 2014.
- [14] Delmar B. Davis, Hazeline U. Asuncion, Ghaleb M. Abdulla, and Christopher W. Carr. Towards recovering provenance with experiment explorer. In *Fifth International Conference on Information, Process, and Knowledge Management (eKNOW)*, 2013.
- [15] Tom De Nies, Sam Cottens, Erik Mannens, and Rik Van de Walle. Modeling uncertain provenance and provenance of uncertainty in W3C PROV. In *Proc of the 22Nd Int'l Conf on World Wide Web*, pages 167–168, 2013.
- [16] Tom De Nies, Sam Cottens, Davy Van Deursen, Erik Mannens, and Rik Van de Walle. Automatic discovery of high-level provenance using semantic similarity. In *Provenance and Annotation of Data and Processes*, pages 97–110. Springer, 2012.
- [17] Tom De Nies, Io Taxidou, Anastasia Dimou, Ruben Verborgh, Peter M. Fischer, Erik Mannens, and Rik Van de Walle. Towards multi-level provenance reconstruction of information diffusion on

- social media. In *Proc of the 24th ACM Int'l on Conf on Information and Knowledge Management*, pages 1823–1826, 2015.
- [18] S. Deerwester, S. Dumais, T. Landauer, G. Furnas, and L. Beck. Improving information retrieval with latent semantic indexing. In *Annual Meeting of the American Society for Information Science*, 1988.
- [19] Tudor Dumitras and Iulian Neamtiu. Experimental challenges in cyber security: A story of provenance and lineage for malware. In *Proc of the 4th Conf on Cyber Security Experimentation and Test*, pages 9–9, 2011.
- [20] Juliana Freire, Cláudio T. Silva, Steven P. Callahan, Emanuele Santos, Carlos E. Scheidegger, and Huy T. Vo. Managing rapidly-evolving scientific workflows. In *Proc of the Int'l Provenance and Annotation Workshop (IPA)*, 2006.
- [21] James Frew, Dominic Metzger, and Peter Slaughter. Automatic capture and reconstruction of computational provenance. *Concurrency and Computation: Practice and Experience*, 2007.
- [22] Paul Groth, Yolanda Gil, James Cheney, and Simon Miles. Requirements for provenance on the web. *Int'l Journal of Digital Curation*, 7(1):39–56, 2012.
- [23] Paul Groth, Yolanda Gil, and Sara Magliacane. Automatic metadata annotation through reconstructing provenance. In *Third Int'l Workshop on the role of Semantic Web in Provenance Management, ESWC*, 2012.
- [24] Pengfei Guo, Xuezhi Wang, and Yingshi Han. The enhanced genetic algorithms for the optimization design. In *3rd Int'l Conf on Biomedical Engineering and Informatics*, volume 7, pages 2990–2994, Oct 2010.
- [25] Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1), 2001.
- [26] David A. Holland, Margo I. Seltzer, Uri Braun, and Kiran-Kumar Muniswamy-Reddy. PASSing the provenance challenge. *Concurrency and Computation: Practice and Experience*, 20:531–540, 2008.
- [27] Diane J Hu. Latent dirichlet allocation for text, images, and music. *Research Exam, University of California, San Diego*, 2009.
- [28] Robert Ikeda, Junsang Cho, Charlie Fang, Semih Salihoglu, Satoshi Torikai, and Jennifer Widom. Provenance-based debugging and drill-down in data-oriented workflows. In *Proc of the International Conference on Data Engineering (ICDE)*, 2012.
- [29] Ralf Krestel, Peter Fankhauser, and Wolfgang Nejdl. Latent dirichlet allocation for tag recommendation. In *Proc of the Third ACM Conf on Recommender Systems*, pages 61–68, 2009.
- [30] Barbara Lerner and Emery Boose. RDataTracker and DDG explorer capture, visualization and querying of provenance from R Scripts. In *5th International Provenance and Annotation Workshop*, 2014.
- [31] Sara Magliacane. Reconstructing provenance. In *Proc of the 11th Int'l Conf on The Semantic Web - Volume Part II*, pages 399–406, 2012.
- [32] Tanu Malik, Ligia Nistor, and Ashish Gehani. Tracking and sketching distributed data provenance. In *Proceedings of IEEE Sixth International Conference on e-Science, 2010*, 2010.
- [33] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. 2008.
- [34] Paolo Missier, Stian Soiland-Reyes, Stuart Owen, Wei Tan, Alexandra Nenadic, Ian Dunlop, Alan Williams, Tom Oinn, and Carole Goble. Taverna, reloaded. In *Proc of Scientific and Statistical Database Management*, 2010.
- [35] Tom De Nies, Erik Mannens, and Rik Van de Walle. Reconstructing human-generated provenance through similarity-based clustering. In *6th International Provenance & Annotation Workshop, Poster*, to appear.
- [36] Cagri Ozcaglar. *Classification of email messages into topics using latent dirichlet allocation*. PhD thesis, Rensselaer Polytechnic Institute, 2008.
- [37] Io Taxidou, Peter M. Fischer, Tom De Nies, Erik Mannens, and Rik Van de Walle. Information diffusion and provenance of interactions in twitter: Is it only about retweets? In *Proc of the 25th Int'l Conf Companion on World Wide Web*, pages 113–114, 2016.
- [38] Jing Zhang and H. V. Jagadish. Lost source provenance. In *Proc of the 13th Int'l Conf on Extending Database Technology*, pages 311–322, 2010.
- [39] Jing Zhao, Karthik Gomadam, Viktor Prasanna. Predicting missing provenance using semantic associations in reservoir engineering. In *Fifth IEEE International Conference on Semantic Computing (ICSC)*, pages 141–148, Sept 2011.

Plans and Performances: Parallels in the Production of Science and Music

David De Roure, Graham Klyne,
 Kevin R. Page, John Pybus, David M. Weigl
 Oxford e-Research Centre
 University of Oxford
 Oxford, UK
 david.deroure@oerc.ox.ac.uk

Matthew Wilcoxson, Pip Willcox
 Bodleian Libraries and
 Oxford e-Research Centre
 University of Oxford
 Oxford, UK

Abstract— Whether in the science lab or the music studio, we go in with a plan, we perform, and we make a record of that performance for distribution, consumption, and reuse. Both domains are increasingly data-intensive, with the adoption of new technology, and also socially intensive with democratised and growing citizen engagement. The music industry has embraced digital technology throughout the lifecycle from composition to consumption; scientific practice, and scholarly communication, are also undergoing transformation. Is the music industry more digital than science? We suggest that comparing and contrasting these two systems will provide insights of mutual benefit. Our investigation explores the notion of the Digital Music Object, analogous to the Research Object, for rich capture, sharing and reuse of both process and content.

Keywords—Digital Music Object; Reproducible Research; Research Object; provenance; scholarly communication; workflow

I. INTRODUCTION

The music industry has ‘gone digital’ almost end to end, from composition and recording, through production and distribution, to consumption via downloads and devices, and in some cases to reuse by professional and amateur alike. This is not just the course of the audio recording itself, but also of the associated information, from musical scores to front-of-house materials, and social media in promotion and in response to the performance. This transformation has not been without challenge, notably how established intermediaries endeavour to sustain business when digital copying and transmission can occur at near-zero cost, and the digital technology enjoys widespread and pervasive adoption.

Research, and research communication, are also undergoing a digital shift, as we conduct research at new

This work is supported by: *Fusing Semantic and Audio Technologies for Intelligent Music Production and Consumption* funded by the UK Engineering and Physical Sciences Research Council (EPSRC) under grant number EP/L019981/1, a collaboration between Queen Mary University of London, University of Nottingham and University of Oxford; *Transforming Musicology*, funded by the UK Arts and Humanities Research Council under grant AH/L006820/1 in the Digital Transformations programme, in collaboration with Goldsmiths University of London.

computational and social scales, and our academic papers are born and consumed digitally. The academic publishing industry debates its future while acting to sustain business. The peer-reviewed scientific article, a format now over 350 years old, mirrors the audio recording in moving from physical artefact to digital download. Libraries, which traditionally created and curated physical collections and facilitated their use, have additionally become distributors and curators of born-digital content.

What can science learn from the music industry, and *vice versa*? In this paper we provide a report on this investigation, as we develop a series of tools which bridge these two endeavours. We open in Section II by looking at the common backdrop, as expert and citizen alike adopt new technologies. Section III focuses on description of process, as this underlies the increasingly automated production methods in both domains. The focus switches from process to object in Section IV, as we compare the lifecycles of article and music. Based on this we discuss Digital Music Objects (DMOs) in Section V. Some of our tools and scenarios are described in Section VI, highlighting the intersections, as these are developed to inform and ultimately to realise the DMOs. We draw conclusions and identify future directions of work in Section VII.

II. AFFORDANCES OF THE DIGITAL

The ‘turn’ of data-intensive science is characterised by new computational capability and by working with the capture, analysis, and curation of large datasets [1]. This is realised as we adopt the increasing capacities of our computational, storage, and network hardware in the pursuit of accelerated research, by innovating in data-intensive methods, and establishing new research practice.

The increasing utility and affordability of the hardware has also put it in the hands of citizens. The result is massive growth of citizen engagement in the digital world, thanks to the Web and the smartphone, and with it access to digital cultural goods. Naughton [2] observes the convergence and blurring of boundaries that is occurring through digital technology because artefacts are reduced to the lowest common denominator—the bitstream.

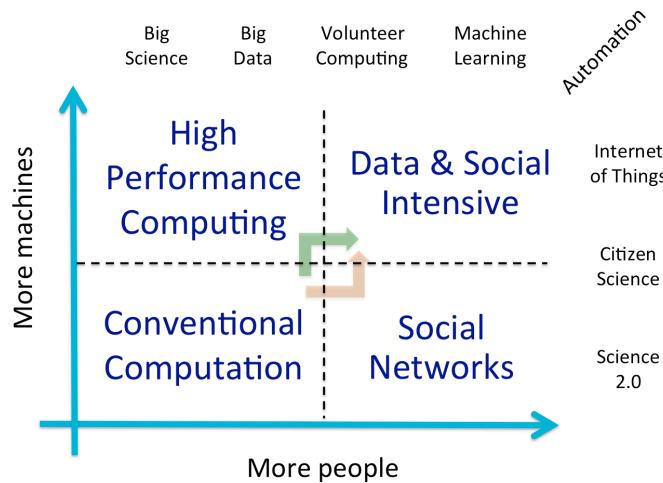


Fig. 1. Computational capability and democratization come together in the fourth quadrant, with increasing automation ahead.

This democratisation is a key digital affordance in our discussion: it means science is not just data-intensive but “social-intensive” too. Content is routinely born digital in lab or home, by professional and amateur. Significantly reuse, repurposing, and republishing is also democratized—empowering the consumer as producer. Distributed teams of empowered individuals form fluidly around content, and new social processes are easily created.

e-Science (and e-Research) today is innovating at the intersection of these trends; for example, massive datasets are generated by online interactions, while other datasets are analysed by citizen science. e-Science has followed a trajectory into high performance and high throughput computing and towards the social, while Science 2.0 and citizen science build on mass citizen engagement and become more computational. This is depicted in Figure 1.

With this comes another key trend and affordance of digital: the rise of automation. We handle the scale, and/or the velocity, of the massive datasets by automating the analytic process. This batch processing is becoming smarter: today citizen science systems use smart algorithms for task allocation, and next we might anticipate machine learning capturing some of the human analytic practices in order to scale to the data volumes of tomorrow’s science.

In this study we aim to concentrate consideration on the *process* as well as data, especially as this underpins the established scientific method—it is how we ‘stand on the shoulders of giants’. We note a significant growth in attention to research data, as stakeholders come to terms with data management plans and data sharing policies and as citizen concern grows for the responsible use of data, but with this comes at least an equal rise in the methods and process of data handling, processing, and analytics. We suggest that issues of sharing and responsible use of process are at least as important as those of data, evidenced through prior work in sharing scientific workflows [3].

III. PROCESS DESCRIPTIONS

A. Descriptions in Laboratory and Studio

A program describes the process that the computer is to execute; we can view the execution as the *performance* of that process. Similarly a computational workflow describes a process and is ‘enacted’. Many decisions are made at runtime as the process is accomplished: which processor core, where to allocate memory and disk, which remote service to use. Exception handling may occur due to events that arise dynamically during execution. While the exceptional decisions may be logged, the routine execution is not usually recorded unless we are debugging. More recently the importance of such provenance traces is recognised in reproducible research [4].

Compare this with a play, musical work, or scientific experiment being performed, from script, score, or lab protocol. In this case the execution is enacted by humans, and unexpected events may be handled by agreed procedures, or creative problem-solving solutions.

The scientist in the lab is aided today by digital capture of data from lab equipment, be it a balance or a mass spectrometer. Logging work via laboratory notebooks is mandatory practice, and increasingly these are electronic lab notebooks (ELNs). Hence the scientists go into a lab with an experiment plan, perform the experiment, and the output of that experiment consists of a diverse array of data, some direct and some contextual, some automated and some reported by human. Crucially, essential problem solving may occur during the experiment. The rich information capture is needed to interpret results and ultimately to support reproducible research. The notion of capture in the lab to facilitate interpretation has previously been articulated in the chemistry domain as *publication at source* [5], even including contextual information such as the status of air conditioning and door interlock, and semantic tooling has been developed [6].

Compare this with the music studio or live recording, which provide a rich digital record of a performance, with high quality audio capture but also the records of recording setup, mixer and virtual instrument parameters and perhaps the self-reported notes of the studio engineer. Musicians are also using digital equipment, their playing supported by scores, notes, and annotations, perhaps using samples and sequences. The compositional phase before the studio, like experiment planning, might also be digitally assisted.

B. Data-Intensive Science

Having established a correspondence between the studio and the laboratory, we might reasonably ask: what about data-intensive science—is there data-intensive music? Digital music datasets are large and created all the time, for example in every performance recorded ‘off the desk’ and uploaded to large curated corpuses such as the Live Music Archive (etree) in the Internet Archive. Multichannel high quality audio capture rates have kept pace so that recording continues to challenge local storage, network, and processing capacity, demanding gigabit per second live data rates. The music studio is comparable to high end laboratory apparatus, with real-time processing.

There are computational challenges in processing the digital audio, especially in real time, and also in analysis and discovery. Techniques established by the vibrant Music Information Retrieval (MIR) community [7] have utility throughout the recording, production, distribution, reuse, archiving, and curation phases of the digital music lifecycle. Feature extraction tools are increasingly set to feature in the toolkit for audio capture and reuse. As in science, the MIR community has developed computational workflow solutions, and a workflow paradigm is proving useful [8].

C. Provenance

In all these examples we can identify (a) a *prospective* plan or script, and (b) a *retrospective* record of what actually happens. These descriptions vary in completeness and specificity. At the comprehensive end of the spectrum, a computational workflow provides a detailed plan, and its provenance trace reports the runtime decisions (e.g. which service was selected?) At the underspecified end, we might have found a few enduring parts of a historical record and piecing it together is itself a process of scholarship.

This is illustrated in Figure 2, where the vertical axis represents decreasing determinism or increasing creativity. We note that the introduction of digital methods brings a potential improvement in completeness of the process record, be it in digital archaeology or digital studios.

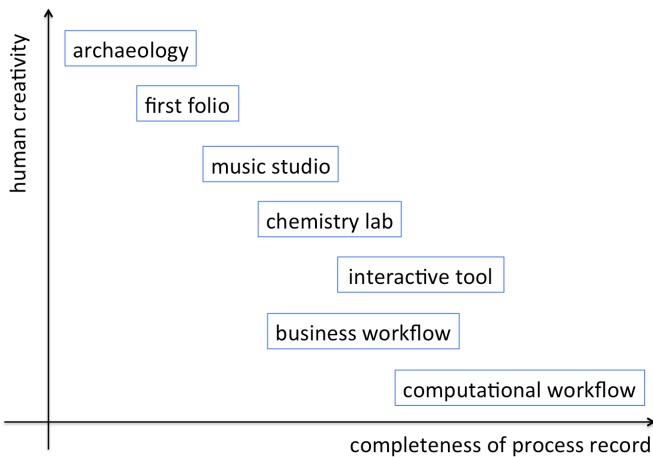


Fig. 2. Illustrating a variety of process descriptions, with more detailed descriptions currently available when machine-generated.

In many cases today, comprehensive descriptions of provenance occur when we are working largely by machine: we see provenance information being ‘born-digital’ alongside content. However it is possible to capture provenance information when humans are involved, such as in a business workflow or a citizen science experiment. The standards and tooling for this exist, notably the W3C PROV recommendations which are motivated by examples of digital systems with human agents [9], as long as the provenance information is substantially complete. We can anticipate further application of PROV in this space, partly motivated by the push to facilitate reproducible research.

IV. LIFECYCLES IN THE ECOSYSTEM

The outcomes of the processes we have described are pieces of recorded music and research results. At first glance, these may seem quite distinct products, that we obtain and use in quite different places for contrasting purposes, with differing scale and demographic of audience. But they are both products of a publishing process. Indeed, before audio distribution at scale was possible, music was distributed as scores—they share the early history of print, and distribution of printed materials.

With product in mind, tangible or digital, one perspective we might usefully take is that of the consumer. Figure 3 is based on the customer activity model of Vandermerwe [10], which we have extended beyond its adaptation presented in [11]. This captures the phases of discovery, use, and post-use from the perspective of the consumer. It enables us to see many similarities, and to tease out differences. For example, discovery, acquisition, and storing in a personal library (stages 1 through 6) have clear commonality. One observed difference might be that citation (5) does not have a direct analogy in music listening from a user viewpoint—however we would argue that unique references to recordings are as much a means to reputation and income in music as they are in scholarship.

Here we are particularly concerned with the way in which these sociotechnical systems have evolved, or indeed reconstructed, due to the affordances of the digital technology. Perhaps the most obvious affordance, and driver of change, is the reduced cost of replication and distribution, where vinyl LP or paper article are replaced by digital download. At the same time this ease of copying facilitates redistribution, a benefit for open content where reuse is encouraged, but a challenge for copyright material such that the industry has sought digital solutions for rights management. Hence stage 8 of the cycle in Figure 3 now demands particular attention due to the ease of reuse, repurposing, and republishing of the digital product.

This product perspective reminds us that the output, be it musical or scholarly content, is also a social object—by which we mean it is the subject of sharing and discourse, around which social networks form and reputation is built. While the CD is a physical object for digital content, today’s social objects can be entirely digital, easily shared and annotated, and they interlink social networks.

We turn then to the lifecycle of the product, with its digital origins: studio practice has changed profoundly due to digital technology [12], and so too has science, for example in chemistry research [13]. We can consider the research cycle diagram in Figure 4, due to Neylon [14], in both contexts.

What are the differences? We might highlight reproducibility, fundamental in the established scientific method, whereby another scientist in another lab should be able to reproduce an experiment *independently* based on the record of this one. At first glance, the recording industry does not rely on such a philosophy. However, we note (a) performers typically wish to replay performances, especially in live settings, and (b) reuse and reinterpretation of audio content underpins the productivity of the industry. Hence the incentive for better capture could be to maximise the reuse of the audio content. We address this in the next section.

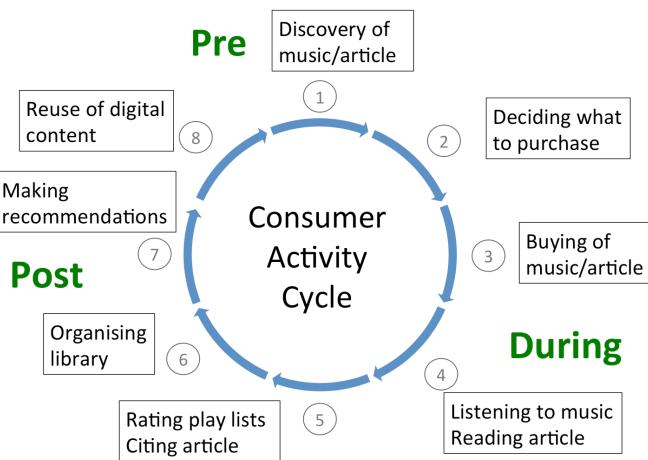


Fig. 3. The Customer Activity Cycle (due to [10]) adapted for music and scholarly articles.

V. DIGITAL MUSIC OBJECTS

The object flowing through the production process in music, from recording through production, distribution, and consumption, was originally a physical object containing an encoding of the audio signal—be it shellac, tape, or vinyl. Now it is the digital audio file. As with tape, the studio generates a multitrack recording (known as *stems*, which are subgroups of mixer channels) and this can be used as the basis for multiple mixes. The consumer receives an audio file, in WAV, FLAC or MP3 format, probably stereo or binaural, possibly in surround sound, for various modes of playing.

Similarly, research data is recorded in the laboratory (or in the field) and leads to publication. Once a printed article, this has been largely replaced by the PDF, very much a digital version of the physical media; i.e. electronic paper. The PDF is produced and distributed for download, like an MP3. Where the research is publicly funded there may be an obligation to publish the research data that supports the results in the paper.

In both cases, then, the content recorded at source is rich and reusable, and the more the richness can be preserved then the better the object can be interpreted, reused, and repurposed. This line of thinking in scientific publishing has led to the notion of the *Research Object*, essentially an aggregation of experimental description and data into a single identifiable and shareable bundle [15].

The goal of Research Objects is "...to create a class of artefacts that can encapsulate our digital knowledge and provide a mechanism for sharing and discovering assets of reusable research and scientific knowledge". Significantly a Research Object is also a social object in that it can be shared and is a subject of discourse, like the physical thing [16].

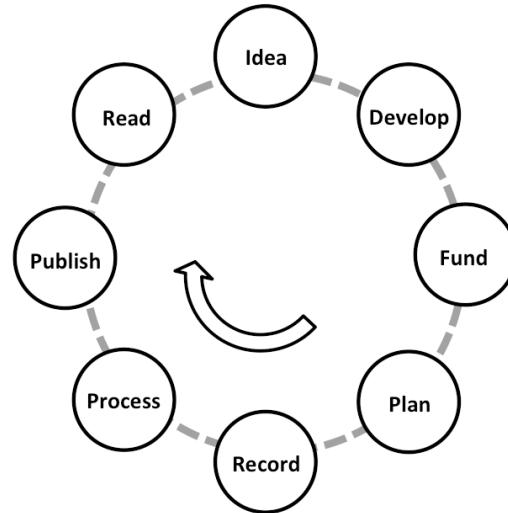


Fig. 4. The Research Cycle (due to [14]).

By analogy we can define the *Digital Music Object* (DMO), which bundles the rich recorded content of music performance for reuse. In this case, the content is of significant size and may be stored in various places and formats for production use. Furthermore, it is temporal in nature. DMOs could be the interchange between studios and production steps, or a DMO could be seen as a mutable object that evolves through the workflow. While the notion of Research Object has been intended to embrace all kinds of content, this has not been the emphasis in today's realizations which more often capture data supporting a published article in a repository, perhaps delivering zip archive or tarball. The DMO must provide a means of gathering distributed content (e.g. a descriptive manifest) and work live as well as in the archive.

The notion of DMO presented here is very much about the object as container, but equally we could be looking at the semantic representation or augmentation of its content. Recent work with the HathiTrust Research Center in the EIEPHÄT project has demonstrated the need to construct and interact with complex research collections or worksets. These require extensions to existing bibliographic ontologies in order to represent the items and their relationships comprehensively [17]. Further consideration of DMO as workset is an important line of enquiry which is set to bring insights from digital scholarship back into the music domain. Semantic audio techniques will also inform content representation.

To conceptualize DMOs we are conducting a series of experiments in performance capture and reuse. We introduce five of these in the next section. Together they will deepen our understanding of process descriptions, and inform the design and practice of DMOs.

VI. UNDERSTANDING THE REQUIREMENTS FOR DIGITAL MUSIC OBJECTS

A. Capture “in the wild”

We have conducted an experiment to understand requirements for DMOs based on current democratised recording practice in the pro-am arena. Using portable digital recording equipment (Focusrite Scarlett 18i20) we have captured live audio in three venues and made the multitrack recordings and “off the desk” mixes available for reuse. Here we distributed the stems in multitrack audio format and different people have developed alternative mixes. We have made some of the audio available for annotation via Music Circle [18].

This in-the-wild exercise has demonstrated the multitrack recording as a social object (physical too, as it was also passed around by USB stick). Outputs have been multiple mixes, including a surround mix, and annotations. Additionally we are experimenting with research-object tooling as applied to this music content, in order to ‘see what breaks’ and to influence the design use cases of a new Research Object system which can be used for recordings of musical performance.

The DMOs produced by this project are stems in standard multitrack audio formats. A “production demonstrator” is currently being designed which focuses on the labelling and grouping of the channels in studio recordings, with the computer suggesting labels based upon inferences from features extracted by signal processing and by other people, perhaps from other stages of the workflow. It aims to show how labels can be derived automatically, shared, and renegotiated between people.

Making the stems available for mixing, and sharing the mixes, is a way of building on human experience and creativity; it also informs future automated mixing by machine.

B. Annalist

Annalist is a software system for individuals and small groups to reap the benefits of using RDF linked data, supporting them in easily creating data that participates in a wider web of linked data [19]. It presents a flexible web interface for creating, editing and browsing evolvable data, without requiring the user to be familiar with minutiae of the RDF model or syntax, or to perform any programming, HTML coding or prior configuration. Requirements centre particularly on achieving low activation energy for simple tasks, flexibility to add structural details as data is collected, access-controlled sharing, and ability to connect private data with public data on the web.

Annalist is designed as a web server application, presenting an interface for defining data structure and managing data. Data is stored as text files that are amenable to access by existing software, with the intent that a range of applications may be used in concert to gather, manage and publish data. During its development, Annalist has been used in a range of applications, which have informed decisions about its design and proven its flexibility and robustness in use.

One of these applications has been the capture of data during a live music performance, including “Carolan Guitar” which reports its own life story [20]. Thus Annalist is serving as an on-ramp to capturing many aspects of the DMO beyond the digital audio.

C. Transforming Musicology

Performance of a musical work potentially provides a rich source of multimedia material for future investigation, both for musicologists’ study of reception and perception, and in improvement of computational methods applied to its analysis. This is particularly true of music theatre, where a traditional recording cannot sufficiently capture the ephemeral phenomena unique to each staging.

In the Transforming Musicology project (see <http://www.transforming-musicology.org/>) we introduced a toolkit developed with, and used by, a musicologist throughout a complete multi-day production of Richard Wagner’s *Der Ring des Nibelungen* (“Ring Cycle”) in Birmingham, UK, in November 2014 [21]. The toolkit is centred on a tablet-based score interface through which the scholar makes notes on the scenic setting of the performance as it unfolds, supplemented by a variety of digital data gathered to structure and index the annotations. Subsequently we structured the data for reuse and further investigation using semantic web technologies, and considered the utility of our tooling from both a user perspective and through an initial quantitative investigation of the data gathered. Additionally we captured physiological data (galvanic skin response, heart rate, micro-movements in three dimensions) from multiple participants—in some ways like additional recording channels.

This illustrates the increasing richness of performance capture. In this regard the requirement of DMOs for musicology are similar to those of the science lab and the studio: multiple and complex data sources, possibly stored in multiple locations, temporally synchronized, richly described and linkable. Today music is seen as a continuous cultural practice, perceived in its context, as opposed to the traditional abstraction that is the score [22].

D. Numbers into Notes

December 2015 saw the 200th anniversary of the birth of Ada Lovelace. A major symposium was held to mark the occasion [23], including the discussion of a thought experiment: had Ada Lovelace lived longer, and had Charles Babbage successfully built the Analytical Engine, what might have happened to pursue this observation by Lovelace:

“Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent.” (note A in [24]).

As part of this we developed an interactive tool for people to generate music from integer sequences. The workflow of the tool mirrors our hypothesised workflow involving the Analytical Engine: the machine runs a parameterised program

to generate a number sequence, and parts of this sequence are then given to different instruments. Inspired by the use of punched cards in the Jacquard loom and the proposed analytical engine, we generate virtual ‘piano rolls’. Hence this is an exercise in algorithmic music generation but with moments for creative human intervention.

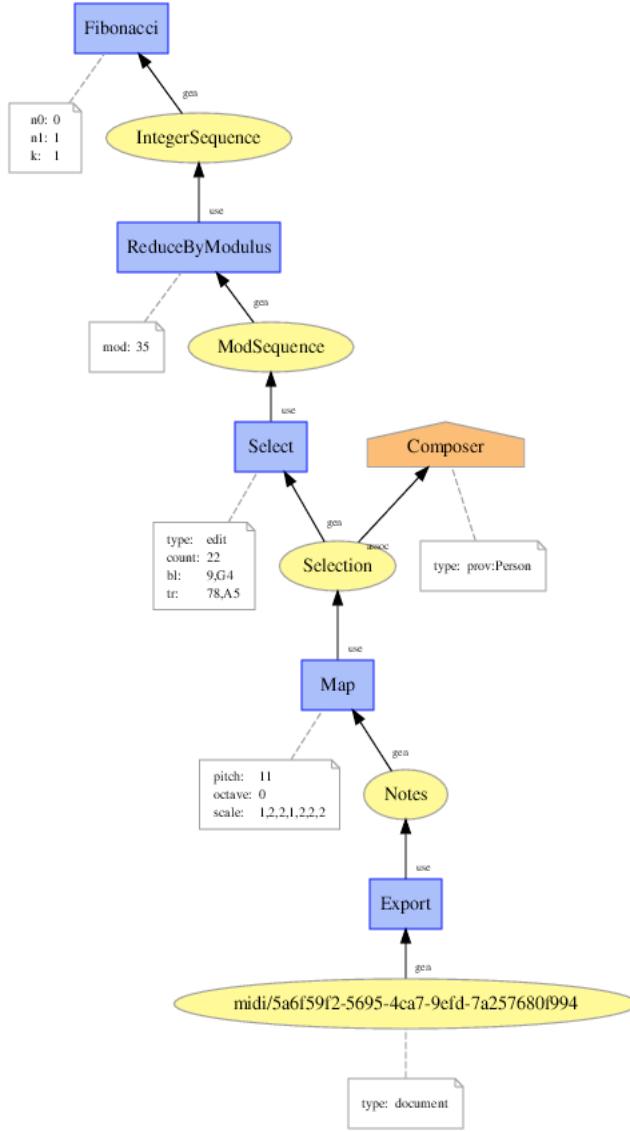


Fig. 5. The Provenance graph describing the output from the “Numbers into Notes” tool, using a Fibonacci sequence. A detailed rendering of this image may be obtained by using the tool.

The programmer and operator (or ‘attendant’) were not allowed to change the numbers generated by the machine, but had full control of the mapping from numbers to notes and then from notes to instruments. The interactive tool, a single page web app (<http://demeter.oerc.ox.ac.uk/NumbersIntoNotes/>), provides several algorithms which illustrate the mathematics of the early 19th century. The primary example involves generalized Fibonacci sequences, reduced by modular arithmetic to produce periodic sequences.

The final stage of the workflow is to export the musical fragment in various formats, one of which is metadata with an automatically generated natural language description of the algorithm parameters, mapping, and selection. We did this to enable someone at a later stage to be able to understand how the fragment was generated or indeed to regenerate the fragment using different tooling; i.e. to reproduce the results of the experiment. For this same reason, one of the output formats is W3C PROV-N, from which an SVG visualization is generated as shown in Figure 5.

E. Lifecycle

We have found it helpful to consider the object as container of digital content but also having a social life—a social object with metadata and annotation, subject of social sharing and of social networks, for consumption by human or machine.

Figure 6 illustrates the lifecycle of the multimedia content associated with a novel performance of the operatic work *Ada sketches* by composer Emily Howard. This was first performed in this format at the Science Museum in London, then based on the London performance we held an event in Oxford, UK in December 2015. The audience completed questionnaires as part of our analysis of the reception of the piece. This is an on-going process, with the next performance in Manchester in July 2016.

The performance was recorded and fed into the Ada Lovelace Symposium, alongside the Fibonacci piece (generated by simulation, but now recorded by a band) and also *Mesmerism*, a second work in Howard’s Lovelace Trilogy. The symposium was also recorded. The resulting materials were used for a presentation at the *Digital Music Research Network* event, and then the entire story was presented reflectively at the *Centre for Digital Scholarship* in the Bodleian Libraries. This is not the end, and nor is there a single archival end point to this process.

F. Discussion

It is clear from these exercises that a DMO must hold together diverse information that might not be routinely captured in individual file formats, including metadata for content and process. However, we also conclude that Digital Music Objects should be realizable through many existing standards; i.e. we do not propose to invent a universal DMO file format that would require pervasive and unmaintainable change.

Our aim then is to define the minimum set of attributes and behaviours that make a DMO, in the spirit of the “Minimum Information Model” for Research Objects [25] which in turn is based on the idea of minimum information checklists. We then need lightweight tooling to support compliant DMOs in some commonly adopted tools, such as Digital Audio Workstations.

Each of these examples has also given us insights into the lifecycle of DMOs. The complexity of reuse in Figure 6 is not just typical of music: in practice, the flow of research data in science looks similar. We also note that our experiments so far have not addressed discovery and curation, and these are also key points for future work.

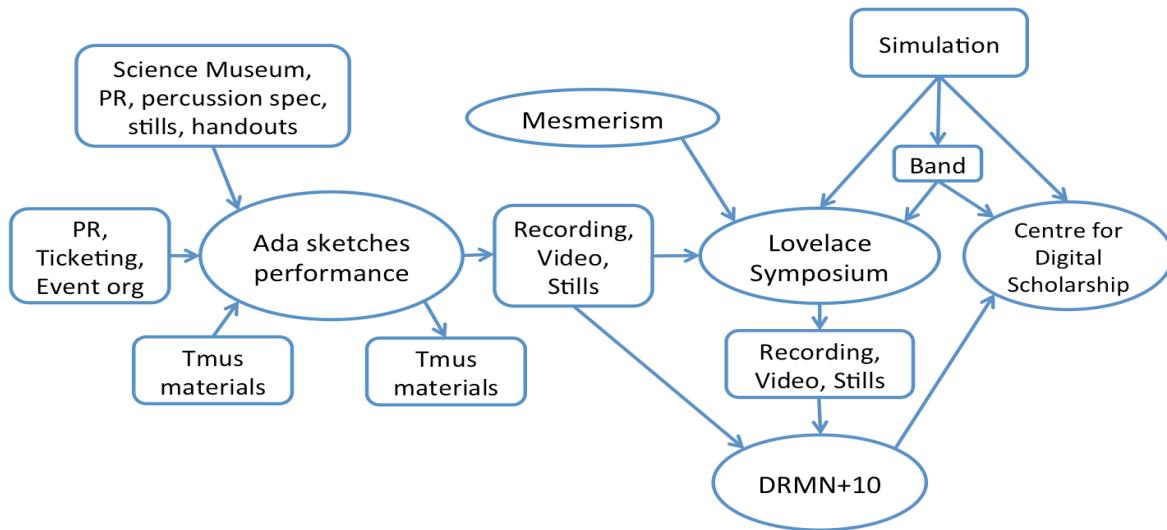


Fig. 6. The flow of physical and digital content in the performance and multiple replays of *Ada sketches* by Emily Howard

VII. CONCLUSION

In this paper we have explained the motivation for a comparison between the creation, production, distribution, consumption, and reuse pipelines of music and science. We have exercised a preliminary comparison framework, looking at the ‘megatrends’ afforded by digital content: duplication and distribution, democratization and disintermediation, and automation, as well as the customer activity cycle and the content lifecycle. On-going “in the wild” developments and capture scenarios have further informed the comparison.

The similarities between laboratory and studio are striking, and they appear to have common trajectories. Our music tooling may directly support or inform tomorrow’s laboratory practice, while reproducible research practice may lead to richer and more re-usable music content—especially in the face of automation.

The next phase of our work will define the minimum DMO concept and deploy tooling for further experimentation. Our key observations so far are:

a) Our work has emphasized the importance of describing process, both prospectively and retrospectively. Our early experiments with W3C PROV confirm its utility and potential. While it demonstrably works well when process descriptions are complete and precise, it does not, of itself, allow for multiple interpretations in multiple contexts. To explore this, our future work will also address provenance of provenance.

b) The digital-social intersections are key, but so too are the digital-physical intersections, as we interact with our (musical and scientific) instruments and materials in the studio, performance space, laboratory, and lecture room. Capturing a full range of contextual information is an essential part of the richness of the DMO.

c) The most compelling insight from our comparison so far is the science/music as performance perspective, with a common theme around the role of creativity. We have gained an appreciation of embodiment—what it means to be the human participant in creating and interpreting performance. This takes us to theories of cognition, and we are currently looking at the notion of *procedural blending* [26], utilising the Annalist tool.

We see that music is an embodied aspect of cognition. Drawing our parallels, how does this apply to science? The situated knowledge approach of Haraway has comment on this, striving for a faithful account of the real world while making explicit our perspective and positioning within the world [27]. All these examples remind us that the audience member (or observer, annotator, scientist) is engaged in a performance, and that annotations captured live are being made by that interpreter in that context and should be interpreted in that knowledge. This argues for more comprehensive consideration of our ‘interpreters’ in the creation and use of Research Objects and DMOs. It also raises questions about the reproducible record of research.

ACKNOWLEDGMENT

The authors are grateful to their colleagues for many discussions, including Steve Benford and Alan Chamberlain (University of Nottingham), Terhi Nurmiiko-Fuller, Carolin Rindfleisch and Ségolène Tarte (University of Oxford), Mark Sandler and Geraint Wiggins (Queen Mary University London), Emily Howard (Royal Northern College of Music), Iris Garrelfs (University of the Arts London), and Jeremy Frey (University of Southampton).

REFERENCES

- [1] A.J.G. Hey, S. Tansley, and K. Tolle (eds.) The Fourth Paradigm: Data-intensive scientific discovery. Microsoft Research, 2009.
- [2] J. Naughton. Lecture: Getting from here to there. *MedieKultur. Journal of media and communication research*, v. 30, n. 57, p. 14 p., nov. 2014. Available at: <http://ojs.statsbiblioteket.dk/index.php/mediekultur/article/view/18609>. Date accessed: 30 May 2016.
- [3] D. De Roure, C. Goble, S. Aleksejevs, S. Bechhofer, J. Bhagat, Cruickshank, D. et al. The evolution of myExperiment. Proceedings - 2010 6th IEEE International Conference on e-Science, eScience 2010, 153-160. doi: 10.1109/eScience.2010.59
- [4] J. Freire, N. Fuhr and A. Rauber. Reproducibility of Data-Oriented Experiments in e-Science (Dagstuhl Seminar 16041). *Dagstuhl Reports*, 6(1), 2016. doi: 10.4230/DagRep.6.1.108
- [5] J.G. Frey, D. De Roure and L. Carr. Publication at Source: Scientific Communication from a Publication Web to a Data Grid. In *Proc. EuroWeb*, (Oxford, UK, Dec 2002), British Computer Society.
- [6] G. Hughes, H. Mills, H., D. De Roure, D., J.G. Frey, L. Moreau, m.c. schraefel, G. Smith, and E. Zaluska. The semantic smart laboratory: a system for supporting the chemical eScientist. *Organic & Biomolecular Chemistry*, 2, (22), 3284-3293. 2004. doi: 10.1039/B410075A
- [7] J.S. Downie, A.F. Ehmann, M. Bay and M.C. Jones. The Music Information Retrieval Evaluation eXchange: Some Observations and Insights. *Advances in Music Information Retrieval* Vol. 274, pp. 93-115. 2010.
- [8] K.R. Page, B. Fields, B., D. De Roure, T. Crawford, J.S. Downie. Capturing the workflows of music information retrieval for repeatability and reuse. *J. Intell. Inf. Syst.* 41(3): 435-459. 2013.
- [9] Y.Gil and S. Miles (eds.) PROV Model Primer. W3C Working Group Note 30 April 2013. Available as <https://www.w3.org/TR/prov-primer/>
- [10] S. Vandermerwe. 2000. How increasing value to consumers improves business results. *Sloan Management Review*, 42, 3, 2000, 27-37.
- [11] T. Regner, J.A Barria, J.V. Pitt, and B. Nevill. An artist life cycle model for digital media content: Strategies for the Light Web and the Dark Web, *Electronic Commerce Research and Applications*, Volume 8, Issue 6, Nov-Dec, 2009. Pages 334-342, doi: 10.1016/j.elerap.2009.05.002.
- [12] A. Pras, C. Guastavino and M. Lavoie. The impact of technological advances on recording studio practices. *J. Am. Soc. Inf. Sci.*, 64: 612-626. 2013. doi:10.1002/asi.22840
- [13] C. Bird and J.G. Frey. Chemical information matters: an e-Research perspective on information and data sharing in the chemical sciences. *Chemical Society Reviews*, 42, (16), 6754-6775. 2013. doi: 10.1039/C3CS60050E.
- [14] C. Neylon. Science in Society. NESTA Crucible Workshop, Lancaster, 28 June, 2009. Available as http://commons.wikimedia.org/wiki/File:Research_cycle.png
- [15] S. Bechhofer, I. Buchan, D. De Roure, P. Missier, P. et al. 2013. Why Linked Data is Not Enough for Scientists, *Future Generation Computer Systems*, Vol. 29, No. 2. pp. 599-611.
- [16] D. De Roure, S. Bechhofer, C. Goble and D. Newman. Scientific Social Objects: The Social Objects and Multidimensional Network of the myExperiment Website, In *Proc. IEEE Third International Conference on Social Computing* (Boston, MA, USA, 9-11 Oct), SocialCom, pp.1398-1402, 2011. doi: 10.1109/PASSAT/SocialCom.2011.245
- [17] T. Nurmikko-Fuller, K.R. Page, P. Willcox, J. Jett, C. Maden, T. Cole, C. Fallaw, M. Senseney and J.S. Downie. Building Complex Research Collections in Digital Libraries: A Survey of Ontology Implications. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries* (Knoxville, USA, June 2015).
- [18] M. Yee-King, M. Krivenki, H. Brenton, A. Grimalt-Reynes, and M. d'Inverno. Designing educational social machines for effective feedback. In *Proc. 8th International Conference on e-learning*. (Lisbon, Portugal, 15-18 July). 2014.
- [19] G. Klyne, C. Willoughby, K.R. Page. Annalist: A practical tool for creating, managing and sharing evolving linked data. Linked Data on the Web Workshop 2016. To appear in CEUR Workshop Proceedings (CEUR-WS.org) Vol-1593.
- [20] S. Benford, A. Hazzard, and L. Xu. The Carolan guitar: a thing that tells its own life story. *ACM Interactions*, Volume 22 Issue 3, May - June 2015, Pages 64-66. doi: 10.1145/2745960
- [21] K.R. Page, T. Nurmikko-Fuller, C. Rindfleisch, R. Lewis, L. Dreyfus and D. De Roure. A toolkit for live annotation of opera performance: Experiences capturing Wagner's Ring Cycle. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR 2015)*. Málaga, Spain.
- [22] N. Cook. *Beyond the Score: Music as Performance*. Oxford University Press, Oxford. 2014.
- [23] Ada Lovelace Symposium 2015 - Celebrating 200 Years of a Computer Visionary, Oxford, UK, December 2015. ACM. 2015. ISBN 978-1-4503-4150-9.
- [24] A.A. Lovelace. Sketch of the analytical engine invented by Charles Babbage, with notes by the translator. In *Scientific Memoirs, Selected from the Transactions of Foreign Academies of Science and Learned Societies*, Vol. 3, 1843, pp. 666-731, volume 3. Richard and John E. Taylor, Red Lion Street, Fleet Street, London. Translation of *Notions sur la machine analytique de M. Charles Babbage* by Luigi Federico Menabrea, in *Bibliothèque universelle de Genève*. Nouvelle série 41, 352-76 (1842).
- [25] M. Gamble, C. Goble, G. Klyne and J. Zhao, MIM: A Minimum Information Model vocabulary and framework for Scientific Linked Data, *IEEE 8th International Conference on e-Science*, Chicago, IL, 2012, pp. 1-8. doi: 10.1109/eScience.2012.6404489
- [26] I. Garrelfs. From Conceptual Blending to Procedural Blending: Applying a Model of Cognition to Process in Sound Art Practice. In Denham, S. and Punt, M. eds., Plymouth: TT OA Papers pp. 71-88. 2016.
- [27] D. Haraway. Situated knowledges: The science question in feminism and the privilege of partial perspective. *Feminist Studies*, 14(3), 575-599.1988.

MOHA: Many-Task Computing meets the Big Data Platform

Jik-Soo Kim*, Cao Nguyen*†, Soonwook Hwang*†

* National Institute of Supercomputing and Networking at KISTI

Daejeon, Republic of Korea

Email: {jiksoo.kim, cao, hwang}@kisti.re.kr

† University of Science & Technology (UST)

Daejeon, Republic of Korea

Abstract—Many-Task Computing (MTC) has been a new computing paradigm that aims to bridge the gap between traditional High-Throughput Computing (HTC) and High-Performance Computing (HPC). MTC applications from various scientific domains such as pharmaceuticals, astronomy, physics often consist of a very large number (from thousands to even billions) of data-intensive (tens of MB of I/O per second) tasks with relatively short per task execution times (from seconds to minutes).

Each task in MTC applications may require relatively small amount of data processing especially compared to existing Big Data applications typically based on larger data block sizes (e.g. the default block size in Hadoop is 64MB). However, they can consist of much larger numbers of tasks where each task communicates through files instead of message passing interfaces such as MPI in HPC applications. Therefore, MTC can be another type of data-intensive workload where a large number of data processing tasks should be efficiently processed within a relatively short period of time.

In this paper, we present design and implementation of MOHA (Many-task computing On HAdoop) which can make an effective convergence of MTC technologies and the existing Big Data platform Hadoop. MOHA is developed as a Hadoop YARN application so that it can transparently co-host existing MTC applications with other Big Data processing frameworks such as MapReduce in a single Hadoop cluster. Our evaluation results based on microbenchmark show that MOHA can substantially reduce the overall execution time of many-task processing with minimal amount of resources compared to an existing Hadoop YARN application. In addition, MOHA can efficiently dispatch a large number of tasks which can be crucial to support challenging MTC applications. MOHA can bring many interesting research issues related to data grouping and declustering on Hadoop Distributed File System (HDFS), scalable job/metadata management, dynamic task load balancing which can ultimately contribute to a new data processing framework in the YARN based Hadoop 2.0 ecosystem.

I. INTRODUCTION

During the past decades, distributed/parallel computing systems have supported various types of challenging scientific/engineering applications. HTC (High-Throughput Computing) [1] mainly supported relatively long running applications consisting of loosely-coupled tasks, while HPC (High-Performance Computing) targets efficiently processing tightly-coupled parallel tasks by employing message passing interfaces such as MPI [2]. In addition, data-intensive computing paradigm mainly focuses on effectively leveraging distributed

storage systems and parallel processing frameworks [3], [4], [5], [6].

Recently, Many-Task Computing (MTC) [7] has been introduced as a new computing paradigm to address challenging applications that cannot be effectively supported through existing HTC or HPC frameworks. These applications from various scientific domains (e.g. pharmaceuticals, astronomy, physics, chemistry) often require a *very large number* of tasks (from thousands to millions of tasks), relatively *short per task execution times* (from seconds to minutes), and *data-intensive* tasks (tens of MB of I/O per CPU second). For example, a stacking service of astronomy images consists of many tasks ranging from 10,000 to millions and each task requires 100ms to seconds of computation with 100KB to MB of input/output data [7]. Similarly, a new drug discovery process (or virtual screening) may have more than one billion computations with a large variance of execution times and involve significant I/O for each computation as the compounds are typically stored in a database (from tens to hundreds of MB large) [8].

MTC emphasizes on exploiting much larger numbers of computing resources over relatively short periods of time to accomplish many computational tasks. Therefore, the primary metrics of target performance in MTC can be seconds which is similar to HPC (e.g. FLOPS, tasks/sec, MB/sec), while the number of operations per month is in HTC [1]. On the other hand, each task in MTC applications may require relatively small amount of data processing especially compared to existing Big Data applications typically based on larger data block sizes (e.g. the default block size in Hadoop [9] is 64MB). However, these applications can consist of much larger numbers of tasks where each task communicates through files instead of message passing interfaces such as MPI in HPC applications. Therefore, MTC can be another type of data-intensive workload where a very large number of data processing tasks should be efficiently processed.

On the other hand, Hadoop [9] has become the *de facto* “Big Data” store and processing infrastructure by leveraging a robust and scalable distributed file system (HDFS [6]) and an efficient parallel processing framework (MapReduce [3]). In the initial design of Hadoop 1.0, MapReduce programming model was tightly coupled with the resource management infrastructure (i.e. JobTracker and TaskTracker). This

architecture has resulted in inefficient abuse of a specific programming model and limitations in the scalability of the centralized job & resource scheduling mechanisms. With the advent of Apache Hadoop YARN [10], Hadoop 2.0 is now evolving into *multi-use data platform* that can harness various types of data processing workflows (batch, interactive, streaming, online, etc.) from a single-use batch processing system by *decoupling* application-level scheduling and overall resource management.

In this paper, we present our initial design and implementation of *MOHA* (Many-task computing On HAdoop) framework which can effectively combine Many-Task Computing technologies with the existing Big Data platform Hadoop. MOHA is developed as one of Hadoop YARN applications by utilizing YARN APIs [11] so that it can transparently co-host existing MTC applications with other Big Data processing frameworks such as MapReduce in a single Hadoop cluster. Users can utilize the MOHA Client (a plain Java object) to submit and monitor their MTC applications. Then MOHA runtime system automatically performs resource allocations through communicating with YARN ResourceManager and efficiently processes many tasks by employing multi-level scheduling mechanism [12], [13].

As a Proof-of-Concept (PoC), we have built a MOHA framework prototype that can execute shell command based many tasks across distributed computing resources in a Hadoop cluster and performed preliminary evaluation of our framework in terms of makespan and task dispatching performance based on microbenchmark [12]. Our evaluation results show that MOHA can substantially reduce the overall execution time of many-task processing with minimal amount of resources compared to an existing Hadoop YARN application (28.5 times faster turnaround time) which can improve the overall cluster utilization. In addition, MOHA can efficiently dispatch a large number of tasks (over 20,000 tasks/sec) which can be crucial to support challenging MTC applications by effectively exploiting well-known open-source distributed message queues [14], [15] and streamlined task dispatching mechanism [13]. MOHA can bring many interesting research issues related to data grouping & declustering on Hadoop Distributed File System (HDFS), scalable job/metadata management, dynamic load balancing of many tasks. By making an effective convergence of MTC and Hadoop technologies, we desire to contribute a new data processing framework to the YARN based Hadoop 2.0 ecosystem which can enable more rich data analytics workflows that can not be fully or efficiently supported only by using MapReduce.

The rest of this paper is structured as follows. Section II describes concepts and technologies of MOHA and we present experimental results in Section III. We discuss some of related research in MTC and Hadoop in Section IV, and conclude in Section V.

II. DESIGN AND IMPLEMENTATION OF MOHA

Our MOHA framework is developed as one of Hadoop YARN applications to transparently execute MTC applications

along with other data processing workflows in a single Hadoop cluster. In this section, we describe basic concepts and runtime model of YARN based Hadoop 2.0 platform, and present design and implementation details of MOHA framework.

A. Hadoop 2.0

Unlike the Hadoop 1.0 where application-level scheduling and resource management were tightly coupled, from Hadoop 2.0, YARN separates all of its functionality into two layers: a *platform layer* which is responsible for overall resource management, and a *framework layer* that coordinates application execution (as we can see from Figure 1) [11]. Basically, the fundamental concept of YARN is to split two major responsibilities of the JobTracker in Hadoop 1.0 (resource management and job scheduling) into separate daemons: a global and *per-cluster* ResourceManager (RM) and a *per-application* ApplicationMaster (AM).

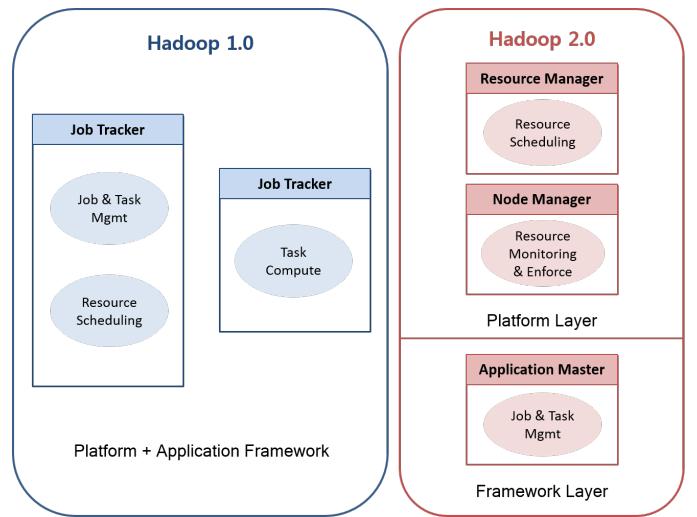


Fig. 1: From Hadoop 1.0 to Hadoop 2.0 [11]

Specifically, the RM performs *pure* resource scheduling including tracking resource usages, monitoring the health of nodes in the cluster, enforcing resource allocation invariants, and arbitrating any conflicts among users. By separating these multiple operations that were previously performed by a single JobTracker in Hadoop 1.0, RM simply allocates resources as form of *containers* based on an application's resource requirements (can be represented by CPU and/or memory) and ignores how the application actually makes use of those allocated resources (called *first-level scheduling*). Then, application specific responsibility is delegated to a per-application AM which coordinates the logical execution of a single application by requesting resources (containers) from RM, starting the application on allocated containers by communicating with the NodeManager (NM), and coordinating the execution plan (called *second-level scheduling*). The NM is a per-node slave which is responsible for launching multiple application containers, monitoring their resource usages, and reporting to RM.

Therefore in some sense, YARN is implementing a multi-level scheduling mechanism [16] where resource allocation (first-level) and application-level scheduling (second-level) are decoupled. Consequently, YARN RM can more focus on fair resource distributions among multiple Hadoop applications and application-specific operations are totally dependent on per-application AMs which can improve the scalability of the overall Hadoop cluster management. Conventional multi-level scheduling mechanisms in MTC community have been mainly adopted in order to *circumvent* the performance bottleneck of traditional local batch schedulers [12], [13], [17], [18]. Similar to the YARN RM, a first-level scheduling mainly reserves resources by submitting *pilot-jobs* [19], [20] to traditional local batch schedulers (e.g. PBS [21], LoadLeveler [22], gLite [23]) as batch jobs. Then, each pilot-job bypasses the local batch schedulers and pulls the tasks *directly* from the separate job queue which can implement a lightweight and fast task dispatching mechanism.

Based on the concept of MTC multi-level scheduling techniques, MOHA effectively exploits YARN as a first-level scheduler to acquire necessary resources (containers) in a Hadoop cluster and implements a fast and lightweight task dispatching mechanism. In this way, MOHA can overcome the performance bottleneck of YARN resource allocations & deallocations (as we will experimentally see from Section III-B). Since MOHA is introduced as a new framework into the existing Hadoop 2.0 platform, we have implemented it as an YARN application (“framework layer”) by leveraging YARN application APIs [11]. Specifically, we have developed **MOHA Client** which submits a MOHA application and performs data staging, **MOHA Manager** which is an ApplicationMaster of our MOHA framework mainly responsible for job & metadata management, and **MOHA TaskExecutor** which implements streamlined task pulling & processing mechanisms running on allocated containers.

B. MOHA System Architecture and Runtime Model

Figure 2 shows overall architecture and runtime model of MOHA framework which consists of MOHA Client, MOHA Manager, and MOHA TaskExecutor. Users can leverage the MOHA Client to submit and monitor their MTC applications and then RM allocates a container (resource) for the MOHA Manager based on parameters provided by the MOHA Client and launches it on top of the allocated resource. Once started, the MOHA Manager performs appropriate resource allocations by communicating with RM, and conducts creating a *MOHA Job Queue* (which contains many tasks) and launching MOHA TaskExecutors (which pull the tasks from the MOHA Job Queue and process them).

The MOHA Client is a plain Java object which utilizes YARN Client APIs and performs typical YARN application submission process as depicted in Figure 3a (from step 1 to step 3). Specifically, MOHA Client conducts following steps of operations:

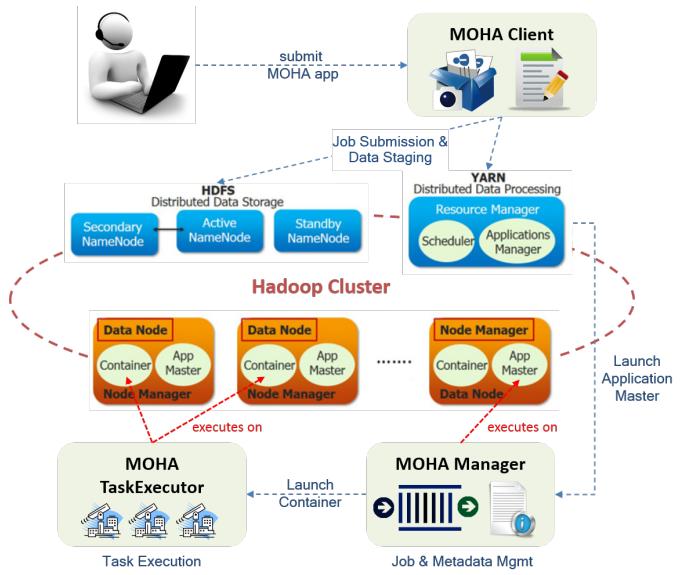


Fig. 2: MOHA System Architecture

- 1) Client Application Request: The MOHA Client creates a new MOHA application and informs RM that it will submit a new application.
- 2) Response with ApplicationID: The RM assigns an ApplicationID for this new MOHA application and prepares for accepting it.
- 3) Data Staging: The MOHA Client uploads required data files into the shared storage file system (HDFS) including application input data, executable, job description file, etc.
- 4) Application Submission Context: The MOHA Client constructs an *Application Submission Context* (including ApplicationID, user, queue, and *Container Launch Context* which consists of resource requirements, environment settings, security tokens) for this new MOHA application and submits it to the RM.

Basically, the MOHA Client submits a *MOHA Job* which represents an MTC application consisting of *a bag of many tasks* and provides runtime requirements of the MOHA Manager. To support the submission of a bag of tasks, MOHA Client currently provides a simple job description file which can describe multiple shell command based tasks. To effectively support various types of MTC applications performing parameter sweeps or N-body calculations, we will consider higher-level job description mechanisms such as OGF JSDF standard [24] as in our previous work [13].

The *Application Submission Context* contains the ApplicationID, user, queue, and other information required to start an AM (in our case, MOHA Manager). In addition, a *Container Launch Context* is included in the Application Submission Context which provides resource requirements (CPU/Memory), job related files (in our case, MTC application input file, executable, job description file), security tokens, and other information needed to launch an AM on

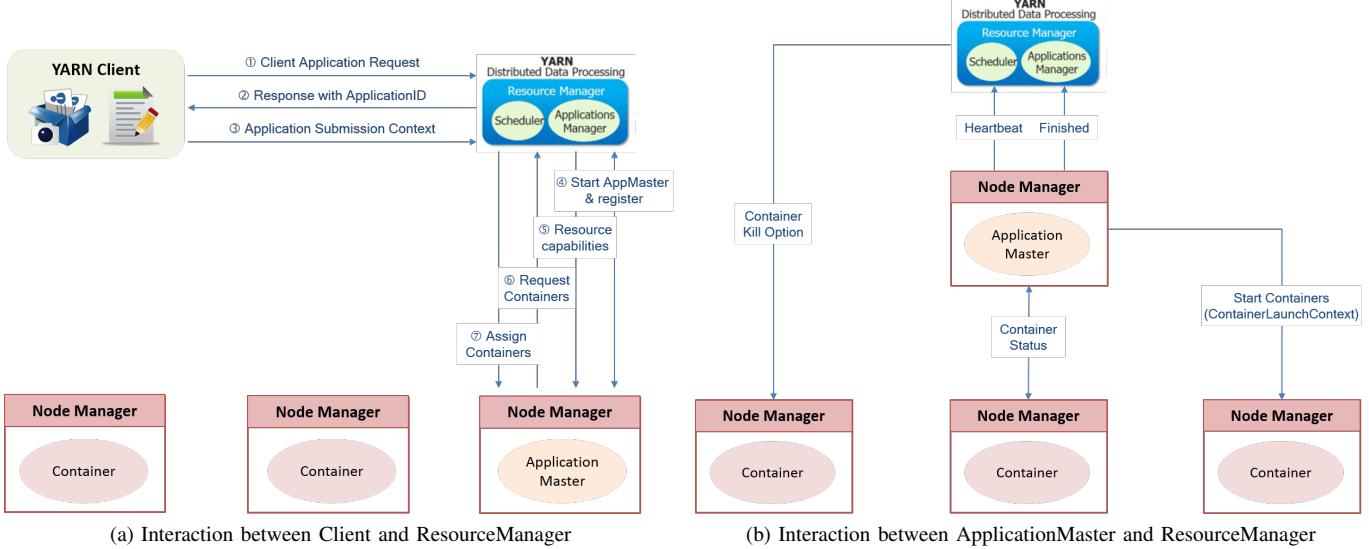


Fig. 3: Hadoop YARN Application Launch Process [11]

a node. Interestingly, YARN provides *resource localization* mechanism [11] for the job related files included in the Container Launch Context so that these files are *automatically* downloaded (typically from the HDFS) and prepared for use in the local working directories of containers by the NMs.

In our case, application input data and executable are necessary for running MOHA TaskExecutors. Once these files are uploaded into the HDFS by the MOHA Client, YARN NM automatically downloads them into the current working directory of an allocated container. Then it creates *symbolic links* of those files for easy access. Similarly, the job description file can be prepared for the MOHA Manager so that it can read in the file to split a MOHA Job into many tasks in its current working directory with the help of YARN resource localization service. After the submission of an application, the MOHA Client can also request RM to kill the application or provide status reports about the application.

Once a MOHA application is submitted to the RM by a MOHA Client, RM allocates a *container 0* (a special container only for the AM) for the MOHA Manager. Based on the Application Submission Context provided by the MOHA Client, RM launches the MOHA Manager on the allocated resource (step 4 in Figure 3a). Then, the started MOHA Manager communicates with RM to get required resources allocated and launch MOHA TaskExecutors on those resources as described in Figure 3a (from step 5 to step 7) and Figure 3b. Specifically, a MOHA Manager performs following steps of operations:

- 1) Registration: The launched MOHA Manager registers itself with RM.
- 2) Resource Capabilities: RM informs the MOHA Manager of available resource capabilities (in terms of CPU and Memory) in the cluster which can be utilized for various resource allocation policies.
- 3) Request Containers: The MOHA Manager requests con-

tainers for MOHA TaskExecutors to the RM based on the status of resource availability.

- 4) Assign Containers: RM informs a list of available resources (containers) and corresponding NMs.
- 5) Create and Launch a MOHA Job Queue: The MOHA Manager creates a MOHA Job Queue by leveraging distributed message queues [14], [15].
- 6) Insert Tasks: Based on the job description file provided by the MOHA Client, MOHA Manager splits a MOHA Job into multiple tasks and inserts them into the queue.
- 7) Start Containers: The MOHA Manager constructs the Container Launch Context for MOHA TaskExecutors and launch them on allocated resources through communicating with NMs.

Unlike conventional YARN AMs, the MOHA Manager should provide an effective mechanism to manage a potentially very large number of tasks from MTC applications. Therefore, a *scalable* and *robust* MOHA Job Queue becomes one of the most important components that can enable efficient and reliable task dispatching and processing. For this purpose, we are leveraging two well-known Apache open-source distributed message queues, *ActiveMQ* [14] and *Kafka* [15], [25]. Consequently, we perform a comparative analysis of two different MOHA frameworks, **MOHA-ActiveMQ** (ActiveMQ as a MOHA Job Queue) and **MOHA-Kafka** (a MOHA Job Queue based on Kafka). As we can see from some related research efforts [26], [27] in the MTC community that can improve the scalability of message queue systems, building a scalable and robust job queue can be a separate research topic that can be easily integrated in our MOHA framework.

ActiveMQ is a message broker in Java that supports Advanced Message Queuing Protocol (AMQP) [28] and provides a JMS client. ActiveMQ provides many configurations and features, and has been a mature message queue solution.

However it cannot scale very well in larger systems [25]. Kafka is an open source, distributed publish and consume service developed by LinkedIn. The design goal of Kafka is to provide a system that gathers logs from a large number of servers, and feeds it into HDFS which can be a natural fit into a YARN based Hadoop cluster. Kafka is fully distributed and provides high throughput so that it shows better scalability.

Finally, once a MOHA TaskExecutor starts on an allocated container, it starts pulling tasks directly from the MOHA Job Queue (either based on ActiveMQ or Kafka) and processes the tasks. MOHA TaskExecutor automatically exits if there are no more tasks left in the MOHA Job Queue. Since the resource allocation and task scheduling are decoupled in our MOHA framework (MOHA Manager simply assigns containers for MOHA TaskExecutors and actual tasks are efficiently processed by multiple MOHA TaskExecutors), it can implement an efficient task dispatching mechanism with minimal amount of resources. In addition, multiple MOHA TaskExecutors *proactively* pull the tasks from the MOHA Job Queue as long as there are remaining tasks in the queue which can show task dispatching bandwidth and scalability of the queue system. As we discussed in Section II-A, this type of multi-level scheduling mechanism (where a MOHA TaskExecutor corresponds to a pilot-job) can improve the overall resource utilization by minimizing resource usages and achieve higher performance of task dispatching and processing.

C. Discussion

As we mentioned in Section I, MTC applications typically require much larger numbers of tasks with relatively short task execution times, each of which performs substantial amount of data operations with potential interactions through files. In order to address these challenging issues from MTC applications, high-performance task dispatching mechanisms, effective dynamic load balancing techniques, and data-intensive workload support should be investigated and seamlessly integrated.

We argue that the existing Big Data platform Hadoop can be a viable choice for addressing these challenging applications so that technologies from MTC community should be effectively converged into the ecosystem. This is because there can be numerous data processing and analytics workflows that can not be effectively supported only by utilizing MapReduce [29]. For this reason, the current generation of Hadoop platform based on YARN is evolving into a multi-use data platform where various types of applications can coexist.

Section II-B presents our initial design and implementation of MOHA framework that can efficiently process shell command based many tasks in a Hadoop cluster by implementing multi-level scheduling and streamlined task dispatching as a Proof-of-Concept (PoC). However, to support real scientific or engineering applications based on many tasks, we need to further investigate various techniques that can improve overall system throughput and reduce user response time including but not limited to followings:

- *Scalable Job/Metadata Management:* As we experimentally see from Section III-B, current available open-

source distributed message queues can achieve high-performance task dispatching. However, as the overall size and complexity of MTC applications are growing, current message queue solutions might become another performance bottleneck. Therefore, improving scalability and performance of job queues for many tasks can be one of important research topics as we can see from some related work [26], [27] in MTC community based on Distributed Hash Table (DHT) and Cloud Computing technologies. Similarly, scalable metadata management framework (such as Li et al. [30]) that can monitor overall progress and status of multiple running MTC applications across Hadoop cluster nodes can play an important role to improve user convenience and productivity.

- *Dynamic Task Load Balancing:* Since MTC applications consist of much larger numbers of tasks compared to typically available computing resources in a cluster, a *task bundling* technique [12], [17] which aggregates a group of small tasks and uses it as a unit of dispatching and processing, is often adopted. For example, Falkon could achieve over 15,000 tasks/sec throughput by employing a bundling size of 100 (100 tasks are dispatched at once) and 128 processors [12]. However, this kind of task bundling technique can be only effective when the execution times of tasks in an MTC application are relatively *homogeneous* (as in the case of Falkon where “sleep 0” based microbenchmark was used). If there can be a large variance in task execution times, we need a weighted form of bundling technique by estimating the size of each task. Job profiling techniques such as Kim et al. [31] based on the Principal Component Analysis (PCA) [32] can be applied to estimate the running times of tasks, however, this problem can become more complicated as multiple users are submitting different types of MTC applications into a shared Hadoop cluster.
- *Data Grouping & Declustering:* One of unique characteristics of Hadoop data processing pattern is that it can exploit “data locality” to make computations *close* to the data. Unlike traditional Hadoop applications where a typical data block size is 64MB, MTC applications usually require small amount of input/output data (from hundreds of KBs to MBs) [7] but from many tasks. This can bring an interesting research issue about grouping and declustering a very large number of small data on existing HDFS since it is not optimized for these small files. Also, the size of a data bundle can be closely related to the unit of task bundling so that we can effectively decluster MTC data bundles across HDFS nodes and using the “data bundle locality”, MOHA Manager can schedule a group of tasks close to the data.

To summarize, MOHA framework can bring many interesting research issues that should combine relatively independent technologies from MTC and Hadoop communities. In addition, we also need to implement and evaluate several ancillary techniques to effectively support real MTC applications such

(a) hdp01 containing the HDFS NameNode
(b) hdp02 containing the YARN ResourceManager
(c) hdp03 mainly for workers

Fig. 4: MOHA Testbed Configurations including Masters (YARN ResourceManager, HDFS NameNode) and Slaves (YARN NodeManager, HDFS DataNode) with additional Hadoop service components

as a rich job description method that can support applications potentially with complex patterns of parameter sweeps or N-body calculations, and sand-boxing techniques to create a safe and appropriate working environment for those applications with complicated library dependencies.

III. EVALUATION

In this section, we present preliminary evaluation results of our MOHA framework in terms of overall application execution time and task processing rate (# of tasks processed / sec) based on microbenchmark. Specifically, we have performed a comparative analysis of three different task dispatching schemes, YARN Distributed-Shell [11], MOHA-ActiveMQ and MOHA-Kafka.

A. Experimental Setup

We have setup a small MOHA testbed consisting of three Dell PowerEdge R630 Rack Servers each of which has two Intel Xeon E5-2620v3 (2.4GHz, 15MB , QPI 8.0GT/sec, 85W, 6-Core) CPUs, 64GB (4 x 16GB RDIMM, 2133MT/s, Dual Rank, x8 Data Width) of main memory, and two 1TB 7.2K RPM SATA HDDs (one for the operating system and libraries, and the other for HDFS configuration). The software stack of MOHA testbed is based on CentOS 6.7 and Hortonworks Data Platform (HDP) [33] 2.3.2 automatically installed by Apache Ambari [34] (as we can see from Figure 4). The minimum and maximum container sizes allocatable by YARN in terms of Memory and VCores are set to 2GB/52GB and 1/20 respectively so that theoretically, up to 60 containers can co-run in the MOHA testbed.

As comparison models, we are using three different task processing frameworks: **YARN Distributed-Shell** which is a simple YARN application that can execute shell commands (scripts) on distributed containers in a Hadoop cluster, **MOHA-ActiveMQ** and **MOHA-Kafka** which exploit ActiveMQ and Kafka as MOHA Job Queues respectively (as

described in Section II-B). Although YARN Distributed-Shell is providing a relatively simple parallel execution environment in a Hadoop cluster, it can be also a useful baseline for exploring a new Hadoop YARN application. Since ActiveMQ is based on a centralized architecture, we are running it on a single node with New I/O (NIO) transport configuration [14]. For the Kafka, we deploy three Kafka *brokers* with minimum fetch size (64 bytes) for a fair comparison with ActiveMQ where a task bundling mechanism is not explicitly supported.

In Kafka, a *topic* is originally a category or feed name where messages are published and it corresponds to a MOHA Job Queue in our system. A topic can consist of multiple *partitions* which are ordered, immutable sequence of messages (in our case, tasks). These partitions can be declustered across multiple Kafka brokers (often called a Kafka *cluster*) for better scalability and reliability. Kafka introduces an interesting concept of *consumer group* where multiple consumers (in our case, MOHA TaskExecutors) can compose a separate group with the unique group name. Then each message published (inserted) to a topic (queue) is delivered to only one consumer in the consumer group. Therefore, if we want to implement a Kafka-based traditional queue system, we can simply put every consumer into the same consumer group. On the other hand, to enable publish-subscribe models, every consumer should have different consumer group names each other.

Since we are exploiting the Kafka to maintain many tasks from MTC applications, each task should not be executed more than once unless we explicitly require it for reliability purpose. Therefore, to ensure maximum parallelism and “one-task-per-executor” property (i.e. each task is only consumed by a single MOHA TaskExecutor), the number of partitions in a topic is at least the number of MOHA TaskExecutors in a Hadoop cluster and all of the MOHA TaskExecutors are belonging to the *same* consumer group (further details of Kafka concepts and configurations can be found in [15], [25]).

Similar to existing MTC research work [12], we have used

microbenchmark based on “sleep 0” tasks which mainly focus on evaluating task dispatching performance. As performance metrics, we are using *elapsed time* and *task processing rate*. The task processing rate is denoted by “# of tasks/sec” processed by each MOHA TaskExecutor. For the elapsed time metric, we further break it down into multiple steps of a MOHA application execution as followings:

- *Total Elapsed Time*: The makespan of a MOHA application (time to complete all tasks).
- *Init Time*: The elapsed time for the MOHA Manager to register itself with RM, create a MOHA Job Queue (based on either ActiveMQ or Kafka), and insert a bag of tasks into the queue.
- *Resource Alloc Time*: The elapsed time to complete all of container allocations requested by the MOHA Manager through RM.
- *Task Processing Time*: The elapsed time to process tasks by pulling them from the MOHA Job Queue in a MOHA TaskExecutor.
- *Overhead*: The other portion of time to complete a MOHA application execution

The main objective of this breakdown in elapsed times is to see various factors such as MOHA Job Queue launching overhead, YARN resource allocation time, and container deallocation overhead that can affect the overall user response times of MOHA applications.

B. Experimental Results

Figure 5 and Figure 6 show comparative performance evaluation results in terms of elapsed times in YARN Distributed-Shell, MOHA-ActiveMQ and MOHA-Kafka.

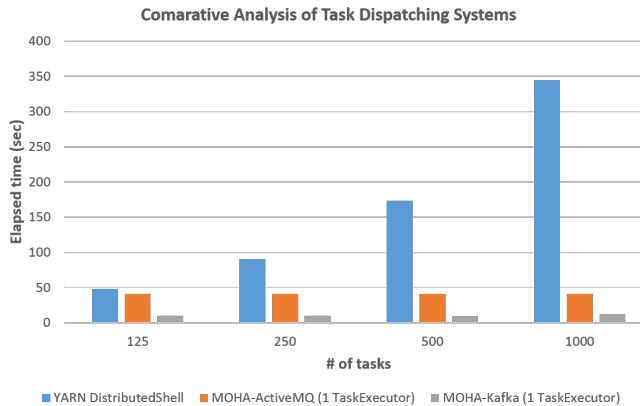


Fig. 5: Performance Comparison of YARN Distributed-Shell and MOHA Frameworks (Total Elapsed Time)

As we can see from Figure 5, MOHA frameworks (MOHA-ActiveMQ and MOHA-Kafka) clearly outperform YARN Distributed-Shell as the number of tasks increases in terms of total elapsed times. Specifically, compared to YARN Distributed-Shell, when the number of tasks reaches 1,000, MOHA-ActiveMQ shows 8.4 times faster turnaround time and MOHA-Kafka achieves even better performance (28.5 times

less makespan). Furthermore, in this experiment, MOHA-ActiveMQ and MOHA-Kafka only utilized a *single* container for the MOHA TaskExecutor to process tasks. This means that a single MOHA TaskExecutor could process 1,000 tasks less than 50 seconds while YARN Distributed-Shell took about 340 seconds to complete all the tasks.

This is mainly due to substantial overhead of multiple resource allocations & deallocations in YARN, as the job scheduling and container allocation are tightly coupled in YARN Distributed-Shell. For example, if an MTC application consists of 1,000 tasks, YARN Distributed-Shell has to launch and execute “1,000” containers (and deallocate them also). However, MOHA frameworks can allocate minimum number of containers as long as MOHA TaskExecutors can achieve enough throughput of task dispatching and processing. As we can see from Figure 6a and 6b, the resource allocation time of a single container can take a couple of seconds. Although the resource allocation overhead may not increase linearly as the number of containers increases, it can clearly show the overhead of multiple container allocations & deallocations. We speculate that this performance gap between YARN Distributed-Shell and MOHA frameworks can rapidly increase as we submit much larger number of tasks, however in our small testbed, YARN Distributed-Shell could not properly handle thousands of tasks.

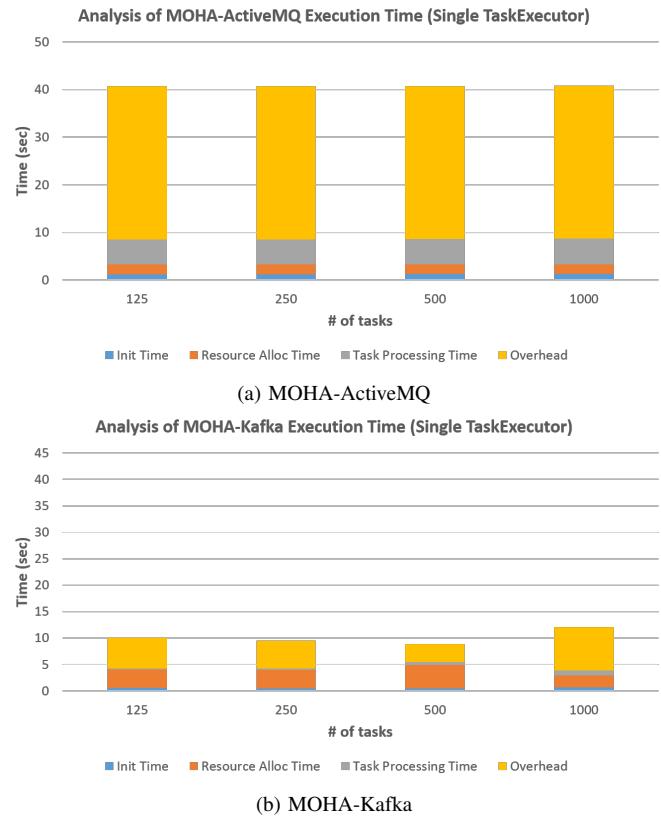


Fig. 6: Execution Time Breakdowns of MOHA Frameworks

By employing multi-level scheduling and streamlined task dispatching mechanisms, our MOHA frameworks can sub-

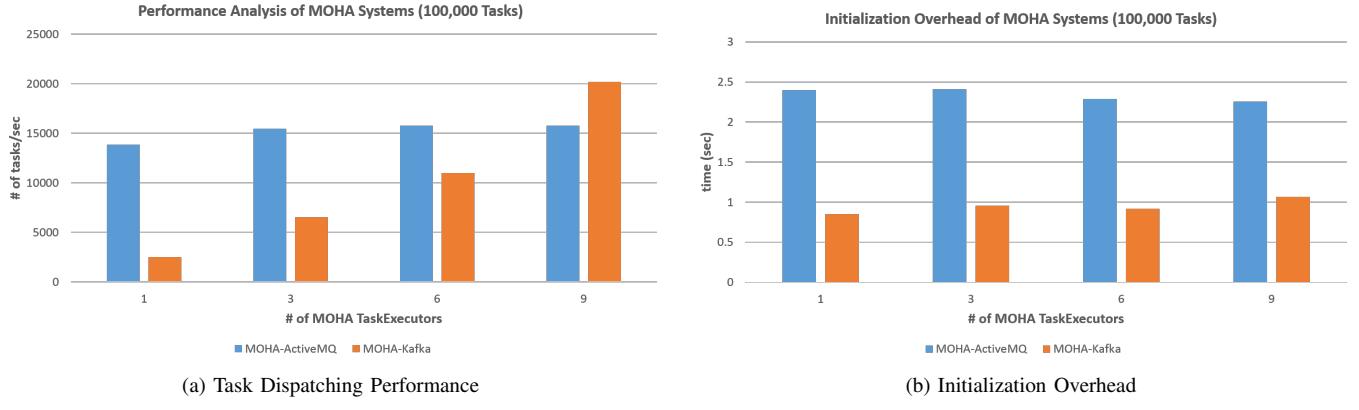


Fig. 7: Task Dispatching Performance and Initialization Overhead of MOHA Frameworks

stantially reduce the cost of executing many tasks. If we have a relatively small number of tasks that are usually long running (as in traditional HTC applications), the basic YARN scheduling mechanism might be sufficient to support. However, results from Figure 5 show that to effectively support MTC applications, we need to introduce another framework that can efficiently process many tasks with minimal resource usages that can eventually improve the overall Hadoop cluster utilization.

Figure 6a and 6b show breakdowns of elapsed times in MOHA-ActiveMQ and MOHA-Kafka as we described in Section III-A. Interestingly, MOHA-Kafka clearly outperforms MOHA-ActiveMQ in terms of total elapsed times and the majority of performance improvements are coming from “Task Processing Time” and “Overhead”. As we will see from the evaluation results of task dispatching performance in Figure 7, the distributed and lightweight Kafka message dispatching mechanism [25] enables MOHA-Kafka to achieve better task processing rate compared to the MOHA-ActiveMQ. Therefore, performance gain from the Task Processing Time is mainly due to architectural and technical differences between ActiveMQ and Kafka. However, the Overheads of MOHA-ActiveMQ in Figure 6a seem to be much larger than those of MOHA-Kafka (as seen from Figure 6b) and they are consistent even if we increase the number of tasks. To investigate this problem, we analyzed the logs from the NodeManager that was maintaining a container allocated for the MOHA TaskExecutor. It turns out that due to higher memory usages in the MOHA-ActiveMQ’s TaskExecutor, the NodeManager resource monitoring time until it returns the allocated container to RM (deallocation) took longer in MOHA-ActiveMQ than MOHA-Kafka. This is possibly due to relatively heavyweight ActiveMQ consumer libraries (compared to Kafka) utilized in the MOHA-ActiveMQ’s TaskExecutor. Therefore, it can show another benefit of MOHA-Kafka that can reduce the total elapsed time by minimizing the resource usages of allocated containers.

Figure 7a and 7b show task processing rate and initialization overhead of MOHA-ActiveMQ and MOHA-Kafka respec-

tively with total 100,000 sleep tasks. As we can see from Figure 7a, MOHA-Kafka outperforms MOHA-ActiveMQ as the number of MOHA TaskExecutors increases. Specifically, when the number of MOHA-TaskExecutors reaches 9, MOHA-ActiveMQ shows over 15,000 tasks/sec and MOHA-Kafka achieves more than 20,000 tasks/sec throughputs which can outperform Falkon’s task dispatching performance with 128 processors [12]. Note that we have not fully utilized Kafka’s task bundling functionality by increasing the fetch size yet, which will show an order of magnitude higher task dispatching performance. Again, these experimental results can show effectiveness of MOHA frameworks that can achieve powerful task processing performance with a small number of containers to support MTC applications. Interestingly, results in Figure 7a can show the characteristics of ActiveMQ and Kafka architectures. The task dispatching performance remains relatively stable regardless of the number of leveraged resources in MOHA-ActiveMQ. However, as we increase the number of TaskExecutors, MOHA-Kafka clearly shows better scalability.

The initialization overhead presented in Figure 7b consists of MOHA Manager’s registration with RM, creating a MOHA Job Queue, reading in the job description file, splitting a MOHA Job into many tasks, and inserting them into the queue. The majority of this initialization overhead comes from MOHA Job Queue creation and insertion of many tasks so that it can reflect the queuing time of ActiveMQ and Kafka. MOHA-ActiveMQ can create a job queue and insert 100,000 tasks within a couple of seconds and MOHA-Kafka can complete this process within 1 second which will be sufficient to insert a large number of tasks. We speculate that with the help of task bundling and batching of task insertions, MOHA-Kafka will show enough performance to dispatch and process billions of many tasks.

IV. RELATED WORK

In the MTC community, there have been numerous research efforts to address challenging problems including high-performance task dispatching mechanisms and data-intensive MTC workload support. Middleware systems such

as Falkon [12], [17], [18], [35] and MyCluster [36] exploit multi-level scheduling mechanisms to achieve higher task dispatching performance than traditional local batch schedulers. The dispatcher in the Falkon efficiently distributes tasks to the executors on the computing resources which can be dynamically acquired by the provisioner. Our MOHA Manager is implementing similar functionality of dispatcher and provisioner, and MOHA TaskExecutors correspond to executors in the Falkon. Falkon also implements data diffusion approach [35] by implementing a data-aware scheduler in the dispatcher component. This can enable each executor to utilize the data cached in nearby neighbors to support data-intensive MTC applications. This is an opposite approach to leverage close cached data to the computations compared to Hadoop's philosophy of leveraging data locality (where computations are sent close to the data). MOHA tries to combine advantages of MTC technologies and Hadoop to more effectively support data-intensive MTC applications.

We have also supported various many-task applications (e.g. high-energy physics, nuclear physics, drug repositioning, virtual screening) through our own middleware system called *HTCaaS* which can leverage distributed computing infrastructures in Korea [8], [13], [37], [38], [39], [40]. Therefore, MOHA is a spin-off project that can inherit core functionality of the HTCaaS system and apply them to the YARN based Hadoop 2.0 platform.

On the other hand, there have been some research efforts that converge existing distributed computing technologies with Hadoop platform. Research such as [29], [41], [42], [43], [44] leverage MPI to support compute-intensive steps (or applications) on top of Hadoop. Especially, GERBIL [29] implements a new YARN application that can perform resource allocation and scheduling of MPI applications. Therefore, similar to our MOHA, GERBIL tries to combine HPC and Hadoop technologies to enrich data analytics in the Hadoop 2.0 ecosystem. In addition, HOG (Hadoop on the Grid) [45] builds a MapReduce framework that can be executed on the Open Science Grid consisting of multiple institutions geographically distributed across the United States. Some of recent research from MTC community try to improve the Hadoop YARN scalability [46] and performance of task dispatching mechanisms [26], [27]. These work can be complementary with our MOHA framework since our main objective is to apply recent techniques from Many-Task Computing to YARN-based Hadoop platform.

V. CONCLUSION AND FUTURE WORK

In this paper, we presented our design and implementation of MOHA (Many-task computing On HAoop) framework which can effectively combine MTC technologies with the existing Big Data platform Hadoop. MOHA is developed as one of Hadoop YARN applications so that it can transparently co-host existing MTC applications with other Big Data processing frameworks such as MapReduce in a single Hadoop cluster.

As a Proof-of-Concept (PoC), we have built a MOHA framework prototype that can execute shell command based many tasks across distributed computing resources and performed preliminary evaluation of our framework in terms of makespan and task dispatching performance based on microbenchmark. Evaluation results presented in Section III show that MOHA can substantially reduce the overall execution time of many-task processing with minimal amount of resources compared to the existing YARN Distributed-Shell, and efficiently dispatch a large number of tasks which can be crucial to support challenging MTC applications by exploiting multi-level scheduling and streamlined task dispatching.

As we discussed in Section II-C, MOHA can bring many interesting research issues related to data grouping & declustering on HDFS, scalable job/metadata management, dynamic load balancing of many tasks. Also, we can consider applying a new type of high-performance storage system in HPC area such as Lustre [47] on top of Hadoop [48] to handle relatively small data files from MTC applications by replacing conventional HDFS. All of these research efforts can ultimately contribute to a new data processing framework for MTC applications in the YARN based Hadoop 2.0 ecosystem. Based on our years of experience to support real scientific applications in MTC area, we plan to apply these applications on our new MOHA framework and report its behaviors under various workloads.

ACKNOWLEDGMENT

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No.R0190-16-2012, High Performance Big Data Analytics Platform Performance Acceleration Technologies Development)

REFERENCES

- [1] M. Livny, J. Basney, R. Raman, and T. Tannenbaum, "Mechanisms for High Throughput Computing," *SPEEDUP Journal*, vol. 11, no. 1, 1997.
- [2] J. J. Dongarra, S. W. Otto, M. Snir, and D. Walker, "A message passing standard for MPP and workstations," *Communications of the ACM*, vol. 39, no. 7, pp. 84–90, 1996.
- [3] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *COMMUNICATIONS OF THE ACM*, vol. 51, no. 1, 2008.
- [4] Y. Gu and R. Grossman, "Toward Efficient and Simplified Distributed Data Intensive Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 6, pp. 974–984, 2011.
- [5] H. G. Sanjay Ghemawat and S.-T. Leung, "The Google File System," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP'03)*, Oct. 2003.
- [6] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," in *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST'10)*, May 2010.
- [7] I. Raicu, I. Foster, and Y. Zhao, "Many-Task Computing for Grids and Supercomputers," in *Proceedings of the Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS'08)*, Nov. 2008.
- [8] B. T. Quang, J.-S. Kim, S. Rho, S. Kim, S. Kim, S. Hwang, E. Medernach, and V. Breton, "A Comparative Analysis of Scheduling Mechanisms for Virtual Screening Workflow in a Shared Resource Environment," in *Workshop on Clusters, Clouds and Grids for Life Sciences (CCGrid-Life 2015) in conjunction with CCGrid 2015*, May 2015.
- [9] The Apache Hadoop project: open-source software for reliable, scalable, distributed computing, "Available at <http://hadoop.apache.org/>."

- [10] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O’Malley, S. Radia, B. Reed, and E. Baldeschwieler, “Apache Hadoop YARN: Yet Another Resource Negotiator,” in *Proceedings of the 4th Annual Symposium on Cloud Computing (SoCC’13)*, Oct. 2013.
- [11] A. Murthy, V. Vavilapalli, D. Eadline, J. Niemiec, and J. Markham, *Apache Hadoop YARN: Moving beyond MapReduce and Batch Processing with Apache Hadoop 2*. Addison-Wesley Data & Analytics, 2014.
- [12] I. Raicu, I. Foster, M. Wilde, Z. Zhang, K. Iskra, P. Beckman, Y. Zhao, A. Szalay, A. Choudhary, P. Little, C. Moretti, A. Chaudhary, and D. Thain, “Middleware support for many-task computing,” *Cluster Computing*, vol. 13, no. 3, pp. 291–314, 2010.
- [13] J.-S. Kim, S. Rho, S. Kim, S. Kim, S. Kim, and S. Hwang, “HTCaaS: Leveraging Distributed Supercomputing Infrastructures for Large-Scale Scientific Computing,” in *Proceedings of the 6th ACM Workshop on Many-Task Computing on Clouds, Grids, and Supercomputers (MTAGS’13) held with SC13*, Nov. 2013.
- [14] Apache ActiveMQ: the most popular and powerful open source messaging and Integration Patterns server, “Available at <http://activemq.apache.org/>”
- [15] Apache Kafka: A high-throughput distributed messaging system, “Available at <http://kafka.apache.org/>”
- [16] G. Banga, P. Druschel, and J. C. Mogul, “Resource containers: a new facility for resource management in server systems,” in *Proceedings of the third symposium on Operating systems design and implementation (OSDI’99)*, Feb. 1999.
- [17] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, and M. Wilde, “Falkon: a Fast and Light-weight tasK executiON framework,” in *Proceedings of the 2007 ACM/IEEE conference on Supercomputing (SC’07)*, Nov. 2007.
- [18] I. Raicu, Z. Zhang, M. Wilde, I. Foster, P. Beckman, K. Iskra, and B. Clifford, “Towards Loosely-Coupled Programming on Petascale Systems,” in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing (SC’08)*, Nov. 2008.
- [19] A. Luckow, M. Santcroos, O. Weidner, A. Merzky, P. Mantha, and S. Jha, “P*: A Model of Pilot-Abstractions,” in *Proceedings of the 8th IEEE International Conference on eScience (eScience 2012)*, Oct. 2012.
- [20] A. Luckow, L. Lacinski, and S. Jha, “SAGA BigJob: An Extensible and Interoperable Pilot-Job Abstraction for Distributed Applications and Systems,” in *Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid 2010)*, May 2010.
- [21] B. Bode, D. M. Halstead, R. Kendall, Z. Lei, and D. Jackson, “The Portable Batch Scheduler and the Maui Scheduler on Linux Clusters,” in *Proceedings of the Usenix, Proceedings of the 4th Annual Linux Showcase & Conference*, Nov. 2000.
- [22] IBM Tivoli Workload Scheduler LoadLeveler, “Available at <http://www-03.ibm.com/systems/software/loadleveler/>”
- [23] gLite - Lightweight Middleware for Grid Computing, “Available at <http://glite.cern.ch/>”
- [24] Open Grid Forum Job Submission Description Language, “Available at <http://www.gridforum.org/documents/GFD.56.pdf>”
- [25] J. Kreps, N. Narkhede, and J. Rao, “Kafka: a Distributed Messaging System for Log Processing,” in *Proceedings of the 6th International Workshop on Networking Meets Databases (NetDB’11)*, Jun. 2011.
- [26] I. Sadooghi, S. Palur, A. Anthony, I. Kapur, K. Belagodu, P. Purandare, K. Ramamurtty, K. Wang, and I. Raicu, “Achieving Efficient Distributed Scheduling with Message Queues in the Cloud for Many-Task Computing and High-Performance Computing,” in *Proceedings of the 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2014)*, May 2014.
- [27] I. Sadooghi, K. Wang, D. Patel, D. Zhao, T. Li, S. Srivastava, and I. Raicu, “FaBRiQ: Leveraging Distributed Hash Tables towards Distributed Publish-Subscribe Message Queues,” in *Proceedings of the 2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC)*, Dec. 2015.
- [28] The Advanced Message Queuing Protocol (AMQP), an open standard for passing business messages between applications or organizations, “Available at <https://www.amqp.org/>”
- [29] L. Xu, M. Li, and A. R. Butt, “GERBIL: MPI+YARN,” in *Proceedings of the 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, May 2015.
- [30] T. Li, X. Zhou, K. Br, D. Zhao, K. Wang, A. Rajendran, Z. Zhang, and I. Raicu, “ZHT: A light-weight reliable persistent dynamic scalable zero-hop distributed hash table,” in *In Proceedings of the 27th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2013.
- [31] S. Kim, J.-S. Kim, S. Hwang, and Y. Kim, “Towards effective science cloud provisioning for a large-scale high-throughput computing,” *Cluster Computing*, vol. 17, no. 4, pp. 1157–1169, 2014.
- [32] I. Jolliffe, “Principal Component Analysis (PCA),” *Springer Verlag*, 2002.
- [33] Hortonworks Data Platform: the industry’s only true secure, enterprise-ready open source Apache Hadoop distribution based on a centralized architecture (YARN), “Available at <http://hortonworks.com/products/hdp/>”
- [34] Apache Ambari project: making Hadoop management simpler by developing software for provisioning, managing, and monitoring Apache Hadoop clusters, “Available at <https://ambari.apache.org/>”
- [35] I. F. Ioan Raicu, Yong Zhao and A. Szalay, “Accelerating Large-Scale Data Exploration through Data Diffusion,” in *Proceedings of the 2008 ACM International workshop on Data-aware distributed computing (DADC’08)*, Jun. 2008.
- [36] E. Walker, J. P. Gardner, V. Litvin, and E. L. Turner, “Creating Personal Adaptive Clusters for Managing Scientific Jobs in a Distributed Computing Environment,” in *Proceedings of the Challenges of Large Applications in Distributed Environments (CLADE’06)*, Jun. 2006.
- [37] S. Rho, S. Kim, S. Kim, S. Kim, J.-S. Kim, and S. Hwang, “HTCaaS: A Large-Scale High-Throughput Computing by Leveraging Grids, Supercomputers and Cloud,” in *Research Poster at IEEE/ACM International Conference for High Performance Computing, Networking, Storage and Analysis (SC12)*, Nov. 2012.
- [38] S. Kim, E. Hwang, T. kyung Yoo, J.-S. Kim, S. Hwang, and Y. ri Choi, “Platform and Co-Runner Affinities for Many-Task Applications in Distributed Computing Platforms,” in *Proceedings of the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CC-Grid 2015)*, May 2015.
- [39] J.-S. Kim, S. Rho, S. Kim, S. Kim, S.-K. Kim, and S. Hwang, “Large-Scale Scientific Simulations throughout HTCaaS: Technologies, Practice and Applications,” in *International Symposium on Grids and Clouds (ISGC) 2014*, Mar. 2014.
- [40] E. Hwang, S. Kim, T. kyung Yoo, J.-S. Kim, S. Hwang, and Y. ri Choi, “Resource Allocation Policies for Loosely Coupled Applications in Heterogeneous Computing Systems,” *IEEE Transactions on Parallel and Distributed Systems*, 2015.
- [41] J. Ye, J.-H. Chow, J. Chen, and Z. Zheng, “Stochastic gradient boosted distributed decision trees,” in *Proceedings of the 18th ACM conference on Information and knowledge management (CIKM’09)*, Nov. 2009.
- [42] MapReduce-MPI (MR-MPI) library: an open-source implementation of MapReduce written for distributed-memory parallel machines on top of standard MPI message passing, “Available at <http://mapreduce.sandia.gov/>”
- [43] X. Lu, F. Liang, B. Wang, L. Zha, and Z. Xu, “DataMPI: Extending MPI to Hadoop-Like Big Data Computing,” in *Proceedings of the 28th IEEE International Parallel and Distributed Processing Symposium (IPDPS ’14)*, May 2014.
- [44] MPICH-yarn: an application running on Hadoop YARN that enables MPI programs running on Hadoop YARN clusters., “Available at <https://github.com/alibaba/mpich2-yarn>”
- [45] C. He, D. Weitzel, D. Swanson, and Y. Lu, “HOG: Distributed Hadoop MapReduce on the Grid,” in *Proceedings of the 5th Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS) 2012 in conjunction with SC12*, Nov. 2012.
- [46] K. Wang, N. Liu, I. Sadooghi, X. Yang, X. Zhou, M. Lang, X.-H. Sun, and I. Raicu, “Overcoming Hadoop Scaling Limitations through Distributed Task Execution,” in *Proceedings of the IEEE International Conference on Cluster Computing (Cluster 2015)*, Sep. 2015.
- [47] P. Schwan, “Lustre: Building a File System for 1,000-node Clusters,” in *Proceedings of the Linux Symposium*, Jul. 2003.
- [48] LUSTRE HADOOP PLUGIN: A Hadoop plugin that enables use of the Lustre Parallel File System, “Available at <https://github.com/Seagate/lustrefs>”

Globus Auth: A Research Identity and Access Management Platform

Steven Tuecke, Rachana Ananthakrishnan, Kyle Chard, Mattias Lidman,
Brendan McCollam, Stephen Rosen, and Ian Foster

Computation Institute

The University of Chicago and Argonne National Laboratory, Chicago, IL 60637, USA

Abstract—Globus Auth is a foundational identity and access management platform service designed to address unique needs of the science and engineering community. It serves to broker authentication and authorization interactions between end-users, identity providers, resource servers (services), and clients (including web, mobile, desktop, and command line applications, and other services). Globus Auth thus makes it easy, for example, for a researcher to authenticate with one credential, connect to a specific remote storage resource with another identity, and share data with colleagues based on another identity. By eliminating friction associated with the frequent need for multiple accounts, identities, credentials, and groups when using distributed cyber-infrastructure, Globus Auth streamlines the creation, integration, and use of advanced research applications and services. Globus Auth builds upon the OAuth 2 and OpenID Connect specifications to enable standards-compliant integration using existing client libraries. It supports identity federation models that enable diverse identities to be linked together, while also providing delegated access tokens via which client services can obtain short term delegated tokens to access other services. We describe the design and implementation of Globus Auth, and report on experiences integrating it with a range of research resources and services, including the JetStream cloud, XSEDE, NCAR’s Research Data Archive, and FaceBase.

I. INTRODUCTION

Developers of research applications and services face two major infrastructure challenges: (1) providing secure and reliable identity and access management (IAM) functionality; and (2) integrating, in a manner convenient for users, with other services that have been developed by independent parties. The difficulties of addressing these challenges has resulted in a fragmented ecosystem of research services. For example, few scientific web applications and science gateways leverage federated identity systems such as InCommon [1]. Instead, each service provider commonly cobbles together its own identity management solution. The result is often applications with limited functionality (due to the high cost and expertise of implementing sophisticated IAM functionality), little integration (due to different IAM approaches), high development and maintenance costs (due to each group creating their own partial solutions), poor user experience (due to inconsistent and incompatible IAM functionality), and even poor security (due to buggy implementations).

Globus Auth is platform-as-a-service (PaaS) that addresses these challenges, with the goal of streamlining the creation, integration, and use of advanced research services [2]. In

brief, it allows research service providers to outsource identity, authentication, credential, and authorization management functions to a cloud-hosted, professionally managed service. In so doing, providers gain five major benefits. First, they gain access to sophisticated IAM functionality that would be difficult for them to implement themselves. Second, they gain integration with other system and services, based on standards such as OAuth 2 [3], OpenID Connect [4], SAML [5], and X.509. Third, they reduce implementation and operation costs: complex in-house code can be replaced with simple REST API calls to a professionally operated service. Fourth, they improve user experience by delivering high-quality, consistently presented IAM capabilities and interfaces. And fifth, they improve the security of their system by using IAM functionality implemented and operated by dedicated security professionals.

The rest of this paper is as follows. We describe the use cases that motivate the Globus Auth design in Section II. In Section III we present the Globus Auth model and its use of authentication and authorization standards. In Section IV we review how Globus Auth is implemented, deployed, and operated. In Section V we present five research services that build upon Globus Auth. Finally, in Section VI we describe related work before summarizing our contributions in Section VII.

II. MOTIVATING USE CASES

Globus Auth addresses important use cases that arise in scientific settings. Specifically, it brokers authentication and authorization interactions, enables identity linking, supports single sign on across scientific services, and enables delegated access to external services. We focus on the advantages that Globus Auth brings to developers and users of scientific services [6]—an increasingly common model for delivering scientific capabilities to a broad community via well defined, Internet accessible services.

Identity broker: Developers of scientific services want to allow users to authenticate using existing identities (e.g., campus accounts). However, implementing such support requires registering clients with a multitude of identity providers, developing support for a number of different authentication protocols (e.g., SAML, OAuth 2, OpenID), addressing subtle differences that exist between implementations due to non-prescriptive specifications, and supporting these implementations over time as changes are inevitably made. Ideally,

developers would like to leverage a reliable and secure service that can broker different identity providers while offering a single stable API from which these identities can be used.

Identity federation: Researchers are accustomed to requiring an ever-growing number of identities to perform their daily tasks. For example, using institution identities to access local storage resources, Google accounts to access Google Docs, and national cyberinfrastructure credentials to access HPC resources. Instead, users want to be able to link their identities, proving ownership once (or infrequently), and then being authorized to perform actions based on this set of identities. For example, consider a common authorization model that uses group membership to manage access to resources. A user should, irrespective of which identity they have used for authentication, be able to perform the roles granted to any identity in that set.

Single sign on: Researchers now have access to many domain-specific and general scientific services. However, these services typically operate in silos in which identities, groups, data, analyses, and other state are not shared between services. Thus, artificial barriers exist between services. Instead we envision a global scientific ecosystem in which many different scientific services build upon a common platform [7]. As a first step towards this goal, methods are required that enable state to be easily shared between services, so that users can use a single identity across services (i.e., single sign on), and service developers can unambiguously refer to users.

Delegated access: Scientific services are increasingly built upon a suite of external services: i.e., as “mash ups.” For example, a climate modeling service may outsource to external services tasks such as managing users, groups, data storage, and computation; the climate modeling service implementation then needs to address only domain-specific issues. This approach has the benefit of allowing developers to deliver advanced capabilities at a fraction of the cost of developing them entirely from scratch. However, for such applications to work, they must be able to make requests to external services on behalf of users, with user information passed down so that user authorizations can be enforced at remote services. We thus require sophisticated authorization models via which users can allow services to transfer to other services the authority to access their state or to perform actions on their behalf. That is, they require secure authorization delegation mechanisms.

III. GLOBUS AUTH

Globus Auth provides a set of features that enable identities to be asserted from a range of identity providers, offers various standard interfaces for integration in third-party applications and services, and enables linking of identities to facilitate federated login. Here, we describe the unique features of Globus Auth.

A. General model

Globus Auth builds upon the OAuth 2 and OpenID Connect specifications to deliver identity and access management as a platform. The high level model is depicted in Fig. 1. Here

we outline the general Globus Auth model and describe how it relates to the OAuth 2 and OpenID Connect specifications. Terms defined in these specifications are denoted with italics.

A *protected resource* (or simply *resource* in this paper) is something that can be addressed via a URL, and is accessible to authorized clients via HTTPS methods (e.g., a REST API).

Globus Auth is an *authorization server*. It issues an *access token* to a *client* after successfully authenticating the *resource owner* and obtaining authorization for the client to access *resources* provided by a *resource server*. The resource owner is typically an end-user, who authenticates to a Globus Auth-managed Globus account using an identity issued by one of an extensible set of (federated) identity providers supported by Globus Auth. The resource owner authorizes (i.e., *consents*) that the client can request access to the resource server on the resource owner’s behalf within a limited *scope*. The client might be an application (e.g., web, mobile, desktop, command line), or it may be another service, as described below.

When a client makes a request to a resource server, it presents the access token as part of the request (in the HTTP Authorization header), to demonstrate that it is authorized to make the request.

Globus Auth can act as the authorization server to an extensible set of resource servers. All Globus [8] services, such as the Globus data management [9] and groups services [10], are resource servers that use the Globus Auth authorization server. Third parties can also create their own resource servers that rely on the Globus Auth authorization server in exactly the same way as Globus services. This broad applicability is why we call Globus Auth a foundational service: it provides a platform for an extensible, integrated ecosystem of resource servers and their clients.

The OAuth 2 specification states that “[t]he interaction between the authorization server and resource server is beyond the scope of [the OAuth 2] specification.” Globus Auth fills this gap by defining a REST API that allows a resource server, upon receiving a request with an access token from a client, to verify that the access token is valid and intended for use with this resource server, and to query for additional information related to that access token such as the client identity, the scope, the resource owner’s identity, and other identities linked to that resource owner’s identity, which the resource server can use to make authorization decisions for the request. Globus Auth leverages the OAuth 2.0 Token Introspection specification (RFC 7662) [11] for this interaction.

Globus Auth also plays the role of an OAuth 2 resource server, allowing clients to access Globus Auth-managed resources, such as identities, and access tokens. For example, clients acting on behalf of end-user resource owners, can be clients to the Globus Auth resource server, to access and manage the end-user’s Globus account-related information. When a resource server receives a request from a client with an access token, that resource server assumes the role of a (different) client to the Globus Auth resource server, in order to validate the access token. In this situation, the resource server uses its own client id and client secret when making requests

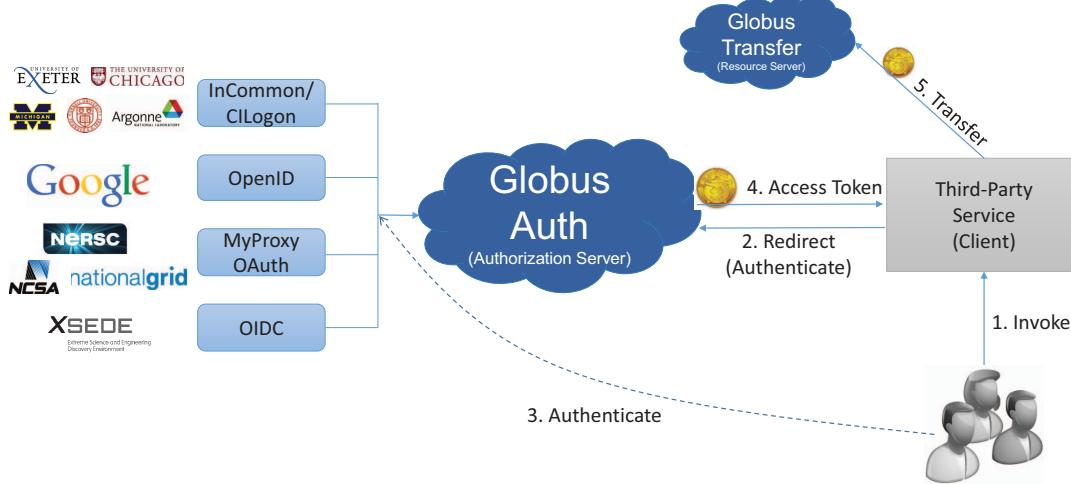


Fig. 1: The Globus Auth model. A user authenticates with a third-party service using one of the supported Globus Auth IDPs. Globus Auth presents a page allowing the user to consent the third-party service to access other resources. Globus Auth presents the third-party service a delegated access token that can be used to both authenticate the user and also access other approved services as that user.

to the Globus Auth resource server. An identity provider can act as a client to the Globus Auth resource server to provision and manage identities within the identity provider's namespace. In this situation, the identity provider uses its own client id and client secret (established previously through a registration process with Globus Auth) when making requests to the Globus Auth resource server.

B. Identities

A Globus Auth identity is a unique name (e.g., `user@example.org`), issued by an identity provider (e.g., an institution or Google), for which a user or client can prove possession via an authentication process (e.g., presenting a password to the identity provider). Globus Auth manages the use of identities (e.g., to login to clients and services), their properties (e.g., associated contact information), and relationships between identities (e.g., allowing login to an identity by using another linked, “federated” identity).

Globus Auth neither defines its own identity usernames nor verifies authentication (e.g., via passwords) with identities. Rather, it acts as an intermediary between external identity providers and clients and services that want to leverage identities issued by those providers.

Globus Auth assigns each identity that it encounters an identifier (`id`): a universally unique identifier (UUID) that is guaranteed to be unique among all Globus Auth identities, and that will never be reused. This identifier is what resource servers and clients should use as the canonical identifier for a Globus Auth identity. Associated with each `id` is an identity provider (`identity_provider`), a name given to the identity by the provider, and other provider-supplied information such as display name and contact email address (see Listing 1).

The identity `username` is a (somewhat) human friendly string, such as an email address or InCommon

Listing 1: Globus Auth Identity

```
username: user@institution.edu
id: ab2312e23-8a34-bd230bca023120ab
identity_provider: institution.edu
display_name: User Name
email: user@institution.edu
```

`eduPersonPrincipleName`, which is guaranteed by Globus Auth to be unique at any point in time. However, because some identity providers (e.g., InCommon) reuse identity usernames (typically with a hiatus between uses), a given identity `username` may map to different identity `ids` over time. In such cases, Globus Auth uses a unique identifier (`provider_specific_id`) provided by the identity provider (e.g., InCommon `eduPersonTargetedID`) to disambiguate, and ensure that at any given time there is a one-to-one mapping between an identity `username` and an identity `id`.

If Globus Auth encounters an identity `username` that has been reused (i.e., same identity `username`, different `provider_specific_id`), it will invalidate the old identity and create a new Globus Auth identity uniquely associated with that identity `username`. Conversely, if Globus Auth encounters an existing identity where the identity `username` has changed for a given `provider_specific_id` (e.g., the user changes their name), it will update the identity `username` while retaining the same Globus identity `id`. Thus, at any point in time, the relationship between identity `username` and Globus Auth identity is unique, and a Globus Auth identity `id` can be relied on to always refer to the same identity.

C. Identity providers

Globus Auth supports an extensible set of identity providers that may employ a variety of identity naming and authentication approaches.

1) Registration with Globus Auth: Each identity provider supported by Globus Auth must register with Globus Auth. (Currently this registration is an out-of-band process, but in the future it can be automated via the Globus Auth API.) At time of registration, Globus Auth establishes a client id and client secret for the identity provider, to be used to allow the identity provider to authenticate as a client to Globus Auth. Identity provider clients can use certain Globus Auth interfaces, such as identity provisioning.

Each identity provider must register either a web browser based authentication protocol (e.g., OpenID Connect, SAML), or a non-browser based protocol (e.g., LDAP, Kerberos, SAML ECP), or both. If an identity provider registers only a non-browser based protocol, Globus Auth will provide a browser based interface for this identity provider. If an identity provider registers only a browser based protocol, some Globus Auth OAuth 2 grant types will not be possible with this identity provider (e.g., Resource Owner Password Credentials Grant), limiting the use of this provider's identities to only browser-based applications.

2) Identity provider namespaces: Each identity provider has one or more namespaces in which it can exclusively issue identity usernames.

An identity provider that issues “user@provider” names is constrained to issuing identities with one or more specific domain names. For example, The University of Chicago’s identity provider, is the only provider that can issue identity usernames with a provider domain of “@uchicago.edu” (e.g., “johndoe@uchicago.edu”). Note that subdomains are distinct namespaces from their parent domain. For example, “@uchicago.edu” and “@ci.uchicago.edu” are distinct namespaces, from potentially different providers.

Some identity providers use email addresses as their user- names. For example, an identity provider restricted to issuing identities with names of “*@provider.org” may issue an identity with the name “johndoe@uchicago.edu@provider.org,” but not “johndoe@uchicago.edu.”

3) Identity and account provisioning: An identity provider, acting as a client to Globus Auth, may explicitly provision its own identities into Globus Auth through the API. The identity must include an identity username, and may include various other fields such as email address, display name, etc.

When a user logs into Globus Auth using an identity that is not associated with a Globus account (i.e., it is not a primary identity or linked identity of any account), either a Globus account must be created with this identity as the account’s primary identity, or this identity must be linked to an existing account’s primary identity. For some identity providers, when an unlinked identity authenticates to Globus Auth, an account will automatically be created with this identity as the primary. For other identity providers, Globus Auth will prompt the user to create an account or link the identity with another account.

4) Supported identity providers: Globus Auth currently supports a range of identity providers including GlobusID, OpenID Connect and Google identity providers.

GlobusID (Globus legacy usernames): Globus previously required that users create an explicit Globus account with a unique Globus username and password. This is no longer required with Globus Auth. Rather, Globus usernames are now managed and issued by a separate service: the GlobusID identity provider. GlobusID identities are issued under the identity provider domain namespace of “@globusid.org.” This identity provider has no special status with Globus Auth: it is just another identity provider.

OpenID Connect: Globus Auth can act as a client to any standard OpenID Connect identity provider. The Globus Auth identity username for OpenID Connect identities will be the `sub` claim from the ID token issued by that identity provider, suffixed with DNS name of the OpenID Connect server as the provider domain. For example, if an OpenID Connect server running at “example.org” issues an ID token with a `sub` claim of “joeuser,” the Globus Auth identity username is “joeuser@example.org.” OpenID Connect identity providers can optionally register to follow Google’s conventions on use of the `sub`, `email`, and `email_verified` claims, which Globus Auth uses as described below.

Google: While Google uses OpenID Connect (with some extensions), it is handled as a special case by Globus Auth. The Google identity provider can issue identities for any email address, and by default, such identities will have a Globus Auth identity username of the email address (i.e., the value of the Google-issued OpenID Connect ID token `email` claim), with a “@accounts.google.com” provider domain: for example, “user@uchicago.edu@accounts.google.com.” Globus Auth only accepts Google-issued identities for email addresses that it has verified (i.e., Google-issued ID token has an `email_verified` claim with the value “true”). Globus Auth uses the value of the Google-issued ID token `sub` claim, as a provider-specific unique identifier for the identity. However, Google is also the exclusive issuer of identities for certain domains, such as “@gmail.com” and certain app domains registered with Globus Auth. For these pre-defined domains, Globus Auth does not add “@accounts.google.com” to the identity username.

Email addresses: Globus Auth treats email addresses as a special type of identity, where the identity’s username is the email address (without an additional provider domain), and authentication of that username is done using the common email verification technique of sending an email to the address containing a secret that the user must validate via a web authentication/verification form. Note that due to identity provider namespacing, as described above, Globus Auth will never allow an email address identity with a domain name issued by a registered identity provider. For example, if the University of Chicago identity provider owns the “@uchicago.edu” namespace, “user@uchicago.edu” must be authenticated using the University of Chicago identity provider, and not simply via email address verification. If a

The screenshot shows the 'Account' section of the Globus Auth interface. At the top, there are links for 'Manage Data', 'Publish', 'Groups', 'Support', and 'Account'. Below that, under 'Identities', there are two entries. The first entry, 'Kyle Chard via Globus ID', lists the name 'Kyle Chard', username 'kyle@globusid.org', and email 'kyle@uchicago.edu'. It also notes that this identity is managed by Globus ID. The second entry, 'Organization: Computation Institute', is labeled as the 'primary identity' and 'linked identity'. It lists the same information: 'Name: Kyle Chard', 'Username: chard@uchicago.edu', and 'E-mail: chard@uchicago.edu'. There are buttons for 'Manage Your Consents', 'Logout', and 'Add Linked Identity'.

Fig. 2: Linked Globus Auth identities.

new identity provider is registered with an exclusive provider domain for which email address identities were previously issued, then Globus Auth will automatically change the provider of such identities to the new identity provider.

D. Linking identities

Globus Auth enables the creation of a “Globus account” using any identity. A Globus account is not an identity, rather it represents a set of identities that belong to the same individual. Fig. 2 shows an example of a Globus account with several linked identities.

A Globus account is a set of identities comprising a primary identity and a number of other identities linked to that primary identity. An identity can be the primary identity of at most one Globus account. Identity linking allows for authentication via one identity to imply login to a Globus account with a different primary identity (i.e., federated identity login). Note that a Globus account is not an identity itself. An account does not have its own name. Rather, a Globus account is identified by its primary identity. Similarly, profile information and other metadata is tied to identities, not to accounts.

Clients and services should grant access to resources on the basis of identities (specifically, identity ids) and their associated attributes (e.g., group memberships, organization affiliations), not accounts. Login to a Globus account, via its primary identity or one of its linked identities, implies login to the account’s primary identity and all identities linked to that account’s primary identity. In other words, login to a Globus account potentially grants access to all resources accessible via all identities linked to that Globus account’s primary identity. In future work, Globus Auth will support “level of assurance” policies to further constrain the access(es) that are allowed by the set of linked identities.

E. Resource server registration

A resource server must register with Globus Auth before it can use Globus Auth as an authorization server. During registration, Globus Auth establishes a client identifier and client secret for the resource server. These credentials allow the resource server to authenticate to Globus Auth in order to obtain and validate access tokens.

A resource server, during registration, can set a number of configurable properties such as a unique name, a restricted

set of allowable identity providers, its scopes, and scopes it uses from other resource servers. The resource server must register a unique resource server name, a DNS name that is uniquely identifiable. The resource server name is used as part of the scope URNs for its resources. Resource servers may optionally restrict the set of permitted identity providers. In this case, users must have an identity issued by the selected identity providers, to access the resource server. During the authentication process, the resource server will request a specific *effective identity* associated with the access token. It will then be given the user’s identity from this provider, even if the user has a different primary identity.

A resource server defines a set of scopes for itself, each corresponding to a subset of that resource server’s resources or functionality. For example, a service could offer separate scopes for *starting*, *viewing*, and *managing* tasks. Each scope has a Globus Auth-issued URN that is unique across all scopes on all resource servers, and is never reused. For example:

```
urn:globus:auth:scope:example.com:tasks:start
urn:globus:auth:scope:example.com:tasks:view
urn:globus:auth:scope:example.com:tasks:manage
```

Clients request an access token that authorizes use of a specific set of scopes (and thus resource servers). A resource server may choose to offer just a single scope that grants full access; more limited scopes allow the resource server to protect resources better by offering more limited rights.

Finally, as described in the following section, a resource server may also define a set of scopes that it will use as a client to other resource servers.

F. Access delegation

The OAuth 2 specification defines how to obtain and use access tokens for interactions between a client and a resource server, within a specified scope. However, it does not prescribe an approach to allow delegated token usage. For example, consider a resource server (RS1) that receives a request from a client (C1) using a request access token (AT1), and that then wants to act as a client (C2) to another resource server (RS2), in order to help fulfill the request. The OAuth 2 specification does not specify what access token should be used in the request from C2 to RS2.

This scenario arises frequently within research IT scenarios. For example, a user of a web application client wants to submit a request to a workflow management service to run a workflow. The workflow resource server, in turn, wants to submit a request to the Globus data sharing service [12] to access data from a shared endpoint for use in the workflow. In order to satisfy the request, the Globus data sharing resource server must, in turn, make a request to the Globus groups service to determine the groups of which the user is a member, based on that user’s linked identities, in order to determine the user’s permissions on that shared endpoint. In this scenario, we call the Globus groups service a *dependent resource server* to the Globus data sharing resource server, and the Globus data

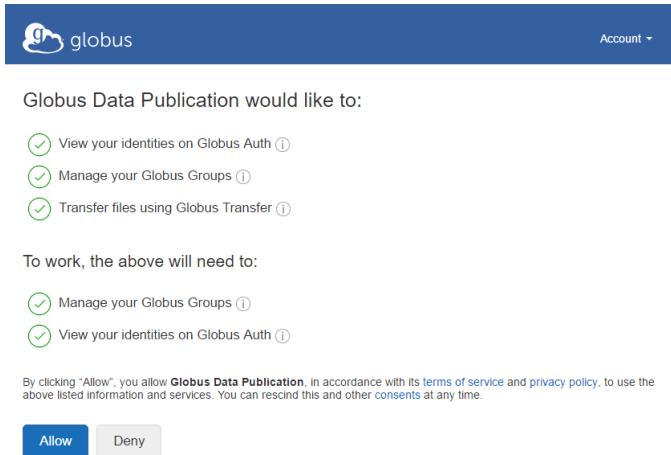


Fig. 3: The Globus Auth consent web interface. In this case the client (Globus Data Publication) is requesting delegated access tokens to retrieve identities managed by Globus Auth, manage groups using Globus groups, and perform transfers using Globus Transfer.

sharing resource server is a *dependent resource server* to the workflow service.

The Globus Auth authorization server provides an API for its resource servers. This API allows a resource server to request new *dependent access tokens* based on the access token that it received from its client. These dependent access tokens can be used to access downstream dependent resource server scopes. The Dependent Token Grant API supports access token delegation for such service invocation chains. When first accessing the client via the Globus Auth authentication workflow, the user is asked to consent the sharing of tokens with the requesting client, as shown in Fig. 3.

G. Groups

Groups are an important component of any authorization model and are therefore used in conjunction with Globus Auth by several external services. While groups management is beyond the scope of Globus Auth, we have extended Globus groups [10] to support linked Globus Auth identities. In this case, linked identities enable a user to perform actions permitted by the union of memberships that their set of identities dictate. Applications can leverage Globus groups to define course-grained authorizations by assessing users' memberships. Globus groups is offered like any other Globus Auth resource server and delegated access to a user's groups can be obtained via a requested scope.

IV. IMPLEMENTATION AND OPERATIONS

The Globus Auth service is comprised of standards compliant OAuth 2 endpoints including custom extensions, API endpoints for querying identity and identity set information, and a user-facing web interface for managing clients, consents, and identity linkages.

The service is written in Python using the Pyramid web framework. It uses an Amazon Web Services (AWS) Reliable Database Service (RDS) PostgreSQL database for storing

state. RDS was chosen for its proven performance, scalability and reliability. Globus Auth has been used in production since February 2016, servicing more than 40,000 Globus users with no downtime to date.

Globus Auth has extremely high requirements on security, availability and data integrity. This section describes important practices and patterns used during development and operations to ensure those requirements are met.

A. Implementation

The Globus Auth implementation comprises three major components: a web interface and API, the application, and a stateful database.

The Globus Auth web interface provides the ability for users to authenticate (by selecting an identity provider), consent to client access, manage an account, and add and remove linked identities. A subset of these capabilities are also available via REST APIs for integration in third-party applications.

The application implements client functionality to integrate with various identity providers, for example acting as an OpenID Connect client to Google. It provides the logic to securely register and manage identity providers and a modular framework via which new identity provider types can be added. The application manages all resources in the system such as identities, accounts, and clients. It provides support to register and configure clients including defining scopes, identity restrictions, and customizing the authentication workflows. Finally, the application also implements server OAuth 2 and OpenID Connect APIs for integration in third-party applications.

The database stores all Globus Auth state including identities, accounts, clients, consents, and identity providers. Database constraints and triggers are used extensively in the data model. For example, a database trigger ensures that if an identity is unlinked, consents that depend on that identity are automatically revoked. This mechanism reduces the risk of application-level bugs and also eliminates the risk that administrator tools that bypass the user application interfaces neglect to implement important side effects.

B. Standards compliance

We have worked hard to ensure that Globus Auth is compliant with the OAuth 2 standard and related specifications. This compliance ensures that external developers can use any one of a number of off-the-shelf libraries to integrate their services with Globus Auth. Custom extensions in Globus Auth, such as returning multiple tokens in a single response, have been designed so that they do not violate existing standards.

A challenge when working with OAuth 2 is that the specification is, in places, vague, and that existing implementations either do not support every aspect of the specification or implement some aspects incorrectly. Thus we have submitted patches to OAuth 2 libraries to implement specification behaviors on which we rely. In addition, we have adapted Globus Auth to support commonly used but non-standard behavior: for example, we support both scope lists that are space-separated

(as mandated by the specification) and comma-separated (the behavior of several widely used implementations).

C. Security

All Globus Auth API endpoints follow a pattern whereby anything that is not explicitly allowed is forbidden. The first part of this model is *ingress security*. Every incoming request must be authenticated with valid credentials before being allowed into a lower level of the application. Business logic permission checks are then performed. Finally, before an object (e.g., an identity) is rendered into a response, an *egress security check* is performed to ensure that the object has been explicitly marked as viewable by the authenticated party.

We have invested considerable effort into ensuring that Globus Auth is protected from both general and targeted attacks. We take care to manage private data and limit both internal and external access to Globus Auth resources. All credentials that Globus Auth is responsible for validating, such as access tokens and client secrets, are only ever stored in a hashed state. Further, any potentially sensitive data stored in the database are encrypted with a secret shared amongst the application servers. Where possible, Globus Auth stores signatures and message digests for the credentials that it issues, rather than the credentials themselves. Secrets are generated using entropy sources suitable for cryptographic use, and signatures are generated using SHA256 or SHA512. Finally, all sensitive data used in deployment are encrypted in our configuration management system with private keys belonging to the application servers.

Globus Auth access tokens are HMAC-signed and contain an expiration timestamp encoded within the token itself. Thus, Globus Auth can reject invalid or expired access tokens without querying the database. Globus Auth also supports an extensible set of token versions, allowing seamless key rotation and token format changes.

D. Testing and development model

Globus Auth features an extensive automated test suite, covering not only expected behavior but also a wide number of error cases such as misbehaving client applications and reliant services being unavailable. The full test suite is run on every single build of the system, and no build can be promoted beyond a “sandbox” environment without every test passing.

Globus Auth follows a rigorous development and testing model, where code changes progress through various environments before being deployed in production. Every code change is first reviewed by another member of the team. Next it is deployed in a sandbox environment where the development team can perform integration testing. Releases are then deployed on a staging environment in which a team of test engineers conduct a range of manual test cases. Finally, the fully tested code is deployed to production servers.

E. Migration and legacy support

Globus Nexus [10] has long provided identity management capabilities for Globus services. With the creation of Globus

Auth it was necessary that we continue to support existing Globus identities. To do so, we registered a new identity provider in Globus Auth, called GlobusID. The transition to GlobusID was further complicated by the fact that Globus Nexus already supported an identity linking model. To retain these linkages we migrated identities from Globus Nexus to Globus Auth. We used a script-based approach to migrate existing identities, along with the identity provider that issued the linked identity, to Globus Auth. We then assigned each identity a unique Globus Auth identifier (i.e., UUIDs) and linked the identities together into a Globus account.

Globus Nexus has offered an OAuth 2 interface (called GOAuth) for several years. While this interface enables integration with third party applications, it neither provides the flexibility of Globus Auth nor supports integration with standard client libraries. Thus, we have deprecated this interface in favor of Globus Auth. To avoid the many applications and services that previously used GOAuth having to migrate to Globus Auth at the time of release, we developed a proxy model via which GOAuth requests can be mapped to Globus Auth requests. We created Globus Auth clients for all existing applications. Requests to the GOAuth authorization endpoint are redirected to Globus Auth, where a Python middleware layer translates the scope and client_id parameters into forms accepted by Globus Auth. Globus Auth handles the authorization request normally, prompting for user consent if needed and returning an authorization code. The legacy client POSTs the authorization code to the GOAuth token endpoint, which acts as a proxy, passing the code to Globus Auth for validation, and returning the access token response issued by Globus Auth in place of its native GOAuth token response.

F. Operations

The Globus Auth service implementation comprises two primary components: the Database and the Application. The Database component is a replicated PostgreSQL database hosted on RDS. The Application servers are a cluster of stateless web servers which use the database for all data persistence requirements. This lets us scale the main load-sensitive component—the web servers—trivially without any impact on service availability. Additionally, we autoscale web worker processes with load, allowing for resilience against failure and spikes in the rate of requests. In addition to the core infrastructure we operate a number of additional services to monitor the system, record and aggregate logs, and detect intrusions. These services enable our operations team to be alerted to service degradation immediately and to review the events leading up to an error or failure.

Deployments and software updates are handled largely by the Chef configuration management system, allowing us to safely assume that all application servers are identical and interchangeable. Chef allows us to develop automated and reproducible scripts for completely configuring distributed infrastructure including all dependencies, application software, and configurations. Using a reproducible and automated model enables us to maintain high availability and reduce errors,

allows versioned infrastructure configurations, supports reproducible deployments, enables deployment of production-like test and staging environments, and facilitates rapid deployment required for updates, infrastructure scaling, and failure. To eliminate downtime caused by updates, all application updates and deployments are conducted as rolling upgrades on the application servers. These are the targets of a continuous integration pipeline that additionally lets us expand or replace the set of application servers at any time.

V. THIRD-PARTY INTEGRATION

External services and applications can use Globus Auth IAM capabilities directly, via the Globus Auth API. In addition, the Globus Auth delegated authorization model allows integrated services to access any other Globus Auth-compliant service using delegated access tokens. We describe here how five services built upon Globus Auth.

A. Globus Services

Globus provides a suite of capabilities for managing research data. Globus Transfer [9] provides for reliable, high-performance, third-party, unattended file transfers between Globus-accessible storage servers. Globus Data Publication [13] supports self-service publication of research data with user-configured submission and curation workflows, metadata association, persistent identifier creation, and flexible search mechanisms. Globus groups [10] provides a user-oriented groups model with web and email-based workflows, role-based management, and configurable visibility and membership policies.

All Globus services use Globus Auth for identity management, authentication, and authorization. Each service has a registered Globus Auth client, implements standard OAuth 2 authentication workflows, and uses Globus Auth APIs to retrieve identity information. Each service is *linked identity aware*, meaning that it derives user state from the collection of linked identities. Thus, a Globus Transfer user can access endpoints shared with any of their linked identities; a Globus Data Publication user can perform the superset of roles assigned to their linked identities; and Globus group membership is derived from the user's linked identities.

As an example of the enhanced capabilities obtained from using Globus Auth, we have recently developed a secure Globus HTTPS endpoint server [14] that enables users to access Globus endpoint-accessible data via HTTPS. This server is an extension of the Globus Connect Server installed on a Globus endpoint, and acts as an OAuth2 resource server for itself and all shared endpoints that it hosts. A user's web client is a "client" to the HTTPS resource server. The HTTPS resource server is a client to Globus Auth and Globus Transfer resource servers. Globus Auth serves as the authorization server. When a user attempts to access a file on a Globus endpoint via HTTPS, the client is redirected to Globus Auth to authenticate using a supported identity provider. The Globus Auth API creates access tokens that match the approved consents (transfer and auth) for this user, and presents these

tokens to the HTTPS resource server. The HTTPS resource server can then validate the token and use the delegated transfer access token to validate the user's access permissions on that endpoint.

B. Research Data Archive

The National Center for Atmospheric Research (NCAR) maintains the Web-based Research Data Archive (RDA), which contains more than 600 data collections. These collections, which range in size from gigabytes to tens of terabytes, include meteorological and oceanographic observations, operational and reanalysis model outputs, and remote sensing datasets to support atmospheric and geosciences research, along with ancillary datasets, such as topography/bathymetry, vegetation, and land use datasets. RDA users are primarily researchers at federal and academic research laboratories. In 2014 alone, more than 11,000 people downloaded more than 1.1 petabytes. Until recently, all downloads were over HTTP, either via Web browser, or via scripts that use wget or cURL.

In order to provide its users with an easy to use, reliable, high performance delivery service, NCAR recently added the ability to download data via Globus. Globus provides simple web interfaces for setting up and monitoring downloads, and implements the downloads themselves by specialized software and protocols that usually outperform HTTP and that can resume a download even if the system being downloaded to (or from) is temporarily turned off or temporarily loses its network connection. The Globus Transfer service thus ensures that downloads complete, regardless of how many times they are interrupted along the way.

When NCAR added Globus data services to RDA, they also integrated support for Globus identities and authentication. The original integration used Globus Nexus and thus required that each RDA user have a Globus username and password (i.e., a GlobusID identity). NCAR is now migrating RDA to use Globus Auth, which means that RDA users will no longer need to have a GlobusID identity. Instead, users can leverage their existing RDA identity either individually or linked with other identities in a Globus account. This enhancement significantly improves user experience and decreases the complexity of the RDA-Globus integration.

C. XSEDE

The Extreme Science and Engineering Discovery Environment (XSEDE) [15] is the national scientific cyberinfrastructure federation in the US. XSEDE supports 16 supercomputers and high-end visualization and data analysis resources across the US. The XSEDE ecosystem also includes administration and scientific services such as the XSEDE User Portal (XUP) and XSEDE Resource Allocation Service (XRAS). These services allow users to manage accounts, publications, and allocations associated with their use of XSEDE resources.

XSEDE is currently integrating Globus Auth into their user-facing services and APIs. XSEDE service APIs are being updated to support Globus Auth access tokens and web interfaces are being extended to operate with Globus Auth supported

identities. To translate XSEDE identities into a supported Globus Auth protocol we have developed an OpenID server that translates XSEDE identities obtained using Kerberos into an OpenID Connect interface. All XSEDE clients (e.g., XUP) use Globus Auth clients that are set with an “effective identity” policy that requires that all users have an XSEDE identity linked to their Globus account.

D. Jetstream

Jetstream [16] is an NSF-funded cloud resource designed to support general science and engineering research. Jetstream offers on-demand access to virtualized resources and services. Its implementation is based on OpenStack and uses the Atmosphere cloud computing environment to expose an interface to users. Atmosphere offers both web and REST interfaces that allow users to instantiate and manage virtual machines, including all of the complexities involved with storage, network, and security configurations. It provides a repository of community virtual machines which are prepopulated with standard software, this allows users to quickly stand up environments to conduct their research.

The Jetstream team have integrated Atmosphere with Globus Auth to provide seamless authentication and authorization experience. Atmosphere acts as a resource server enabling user access to its resources. To support a delegated access model in which other services may use Atmosphere functionality we have registered a new Globus Auth scope for accessing Atmosphere resources. As Jetstream is an XSEDE resource, Atmosphere also requires that each authenticated user have a linked XSEDE identity. The authentication flow redirects users to Globus Auth, which in turn allows the user to authenticate using a supported identity. The resulting access token is returned to Atmosphere and validated via the Globus Auth APIs. The user is then either granted or denied access based on inspection of the token and linked identities.

E. FaceBase

The FaceBase consortium generates data and develops tools to support research into craniofacial development and malformation. Ten *spoke* projects are tasked with generating data and tools, and a single *hub* is responsible for aggregating these data and tools and making them available to the craniofacial research community. FaceBase includes genetic, molecular, biological, imaging, and other data for zebrafish, mouse, human, and other organisms.

The FaceBase architecture is multi-faceted. Data is stored in a proprietary object store and metadata is stored in a relational entity management system [17]. A flexible data search and browsing interface is provided by a dynamic web application. Rather than manage identities and groups (for access control), FaceBase instead uses Globus Auth and Globus groups. The FaceBase team have developed a modular authentication and authorization plugin to their services. This plugin uses a standard OpenID Connect client library to implement the OpenID Connect workflow provided by Globus Auth. To abstract the complexities involved with using multiple services, FaceBase

relies on a branded Globus web instance to enable identity and group management for their users. In this way, FaceBase users are able to authenticate using any of the Globus Auth supported identity providers, manage their Globus account (e.g., linking various identities) and manage groups.

VI. RELATED WORK

Social and commercial applications have long foregone built-in identity management for the use of external identity management solutions. For example, many websites and mobile applications use social network identities, such as Google and Facebook, for authentication. In each case, the third-party application or service implements a standard OAuth 2 client to the Google or Facebook authorization server. However, unlike Globus Auth, these systems do not provide identity brokering capabilities and instead support only a single identity provider.

InCommon is a framework that provides trusted access to online resources and identity management federation across US academic institutions. Global Authentication INfrastructure for education (eduGAIN) [18] extends this notion through a global confederation that connects regional federations, including InCommon. Both InCommon and eduGAIN are built upon SAML and allow service providers to enable user authentication using federated identities. Neither is capable of supporting various IDP protocols, linking identities, or providing scoped or delegated tokens. The Agave [19] and Apache Airavata [20] platforms provide user management, authentication and authorization, job submission, and data management capabilities via REST APIs. Agave supports OAuth 2-based authentication workflows that allow third-party applications to leverage its capabilities. However, unlike Globus Auth, it provides no brokering, account linking, or delegation capabilities. Apache Airavata does not support identity and group management.

Several commercial entities provide identity brokering services. For example, the Google identity platform supports identities from SAML and OpenID Connect identity providers, while also offering OAuth 2 and OpenID Connect interfaces for application integration. Atlassian Crowd [21] is a service-based identity management service for web applications. It enables user identities to be sourced from external directories and exposes different authentication interfaces, such as OpenID, that can be embedded in external applications.

Amazon Web Services Identity and Access Management (IAM) [22] provides identity federation support enabling users to authenticate using their local identity provider. This integration is built on SAML. IAM also provides Web identity federation, a mechanism that allows developers to use different identity providers and to trade an authentication token from these providers for temporary AWS credentials. It supports Amazon, Facebook, and Google identity providers. The related Amazon Cognito [23] associates a unique identifier with each identity that can be referenced across devices and applications. It supports the creation of temporary, limited-privilege credentials such that a third-party application can access AWS resources. Its primary purpose is simplified synchronization of

application state across devices and thus it lacks capabilities such as identity linking and delegated access tokens.

Another commercial service, Auth0 [24], addresses the challenge of mapping from many identities to many applications. It supports a large number of identity providers, including Facebook, Google, LinkedIn, Github, and Amazon. Like Globus Auth, it provides APIs for accessing and managing profiles, and allows users to link identities, use linked identities in federated login scenarios, restrict the range of identity providers that can be used to authenticate with a given client, and use OAuth 2 and OpenID Connect interfaces for integration into third-party applications.

Despite these similarities, Auth0's focus on identity mapping leads to important differences. For example, Auth0 provides little support for enforcing limited access to managed resources. At present, it only provides limited control over who can access the profile information that it manages, offering just three pre-defined scopes for profile information access. In contrast, Globus Auth implements a flexible and extensible consent-based scope model in which many different scopes can be defined for each resource server. Similarly, the Auth0 delegation model is limited: clients must be preconfigured with a group of add-on (or external) services for which delegated tokens can then be obtained. In contrast, Globus Auth provides a rich delegation model in which tokens with different scopes can be obtained by a given client to access other services on behalf of the user. Globus Auth is also differentiated by its support for primarily research identity providers.

VII. SUMMARY

Globus Auth provides a flexible identity and access management platform for the research community. Its unique characteristics, including identity brokering, identity linking, and delegated access model, directly address many frictions associated with creating and operating research services. Globus Auth already supports a range of research identity providers and can be integrated with external research services via standard OAuth 2 interfaces. Globus Auth can thus form the basis for a new generation research platform on which researchers can rapidly develop new services that leverage other research services. Moreover, Globus Auth provides a fabric for integrating the currently fragmented and siloed ecosystem of research services. In the three months since deployment Globus Auth has been adopted by a number of large research projects. This encouraging uptake highlights the potentially transformative effect that Globus Auth can have on the research service landscape.

ACKNOWLEDGMENTS

We thank Globus subscribers for supporting the operation and development of Globus. We also thank users of Globus services for their continued support. This research was supported in part by NSF grant ACI-1053575 (XSEDE) and US Department of Energy contract DE-AC02-06CH11357.

REFERENCES

- [1] V. Welch, A. Walsh, W. Barnett *et al.*, "A roadmap for using NSF cyberinfrastructure with InCommon," in *TeraGrid Conference: Extreme Digital Discovery (TG)*, 2011, pp. 28:1–28:2. [Online]. Available: <http://doi.acm.org/10.1145/2016741.2016771>
- [2] R. Ananthakrishnan, K. Chard, I. Foster *et al.*, "Globus platform-as-a-service for collaborative science applications," *Concurrency - Practice and Experience*, vol. 27, pp. 290–305, 2014.
- [3] D. Hardt, *The OAuth 2.0 Authorization Framework*, <http://www.rfc-editor.org/info/rfc6749> [accessed May 2016], RFC 6749 Std., October 2012.
- [4] N. Sakimura, J. Bradley, M. Jones *et al.*, "OpenID connect core 1.0," http://openid.net/specs/openid-connect-core-1_0.html [accessed May 2016], OpenID Foundation, 2014.
- [5] S. Cantor, J. Kemp, R. Philpott *et al.*, "Security assertion markup language (SAML) v2.0," <https://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf> [accessed May 2016], OASIS, 2014.
- [6] I. Foster, "Service-oriented science," *Science*, vol. 308, no. 5723, pp. 814–817, 2005. [Online]. Available: <http://science.sciencemag.org/content/308/5723/814>
- [7] I. Foster, K. Chard, and S. Tuecke, "The discovery cloud: Accelerating and democratizing research on a global scale," in *IEEE International Conference on Cloud Engineering (IC2E)*, April 2016, pp. 68–77.
- [8] I. Foster, "Globus Online: Accelerating and democratizing science through cloud-based services," *IEEE Internet Computing*, vol. 15, no. 3, pp. 70–73, May 2011.
- [9] B. Allen, J. Bresnahan, L. Childers *et al.*, "Software as a service for data scientists," *Communications of the ACM*, vol. 55, no. 2, pp. 81–88, Feb. 2012. [Online]. Available: <http://doi.acm.org/10.1145/2076450.2076468>
- [10] K. Chard, M. Lidman, B. McCollam *et al.*, "Globus Nexus: A platform-as-a-service provider of research identity, profile, and group management," *Future Generation Computer Systems*, vol. 56, pp. 571–583, 2016. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0167739X1500285X>
- [11] J. Richer, "OAuth 2.0 token introspection," <https://tools.ietf.org/html/rfc7662> [accessed May 2016], Internet Engineering Task Force (IETF), 2014.
- [12] K. Chard, S. Tuecke, and I. Foster, "Efficient and secure transfer, synchronization, and sharing of big data," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 46–55, Sept 2014.
- [13] K. Chard, J. Pruyne, B. Blaiszik *et al.*, "Globus data publication as a service: Lowering barriers to reproducible science," in *11th IEEE International Conference on e-Science*, Aug 2015, pp. 401–410.
- [14] K. Chard, S. Tuecke, I. Foster *et al.*, "Globus: Recent enhancements and future plans," in *the Annual Annual Conference on Extreme Science and Engineering Discovery Environment (XSEDE)*, 2016.
- [15] J. Towns, T. Cockerill, M. Dahan *et al.*, "XSEDE: Accelerating scientific discovery," *Computing in Science Engineering*, vol. 16, no. 5, pp. 62–74, Sept 2014.
- [16] C. A. Stewart, T. M. Cockerill, I. Foster *et al.*, "Jetstream: A self-provisioned, scalable science and engineering cloud environment," in *XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure*, ser. XSEDE '15, 2015, pp. 29:1–29:8. [Online]. Available: <http://doi.acm.org/10.1145/2792745.2792774>
- [17] R. Schuler, C. Kesselman, and K. Czajkowski, "Digital asset management for heterogeneous biomedical data in an era of data-intensive science," in *IEEE International Conference on Bioinformatics and Biomedicine*, Nov 2014, pp. 588–592.
- [18] "eduGAIN," <http://services.géant.net/edugain/> [accessed May 2016].
- [19] R. Dooley, M. Vaughn, D. Stanzione *et al.*, "Software-as-a-service: The iPlant Foundation API," in *5th IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS)*, 2012.
- [20] M. Pierce, S. Marru, L. Gunathilake *et al.*, "Apache Airavata: Design and directions of a science gateway framework," in *6th International Workshop on Science Gateways*, June 2014, pp. 48–54.
- [21] "Atlassian Crowd," <http://atlassian.com/software/crowd/overview> [accessed May 2016].
- [22] "Amazon identity and access management (IAM)," <http://aws.amazon.com/iam> [accessed May 2016].
- [23] "Amazon Cognito," <https://aws.amazon.com/cognito/> [accessed May 2016].
- [24] "Auth0," <http://auth0.com> [accessed May 2016].

Management, Analysis, and Visualization of Experimental and Observational Data – The Convergence of Data and Computing

E. Wes Bethel*, Martin Greenwald†, Kerstin Kleese van Dam‡, Manish Parashar§, Stefan M. Wild¶, and

H. Steven Wiley

*Lawrence Berkeley National Laboratory

†Massachusetts Institute of Technology

‡Brookhaven National Laboratory

§Rutgers University

¶Argonne National Laboratory

||Pacific Northwest National Laboratory

Abstract—Scientific user facilities—particle accelerators, telescopes, colliders, supercomputers, light sources, sequencing facilities, and more—operated by the U.S. Department of Energy (DOE) Office of Science (SC) generate ever increasing volumes of data at unprecedented rates from experiments, observations, and simulations. At the same time there is a growing community of experimentalists that require real-time data analysis feedback, to enable them to steer their complex experimental instruments to optimized scientific outcomes and new discoveries. Recent efforts in DOE-SC have focused on articulating the data-centric challenges and opportunities facing these science communities. Key challenges include difficulties coping with data size, rate, and complexity in the context of both real-time and post-experiment data analysis and interpretation. Solutions will require algorithmic and mathematical advances, as well as hardware and software infrastructures that adequately support data-intensive scientific workloads. This paper presents the summary findings of a workshop held by DOE-SC in September 2015, convened to identify the major challenges and the research that is needed to meet those challenges.

I. INTRODUCTION

The Department of Energy (DOE) Office of Science (SC) operates dozens of national science user facilities that span many disciplines [9]. These facilities include accelerators, colliders, supercomputers, light sources, and neutron sources, as well as facilities for studying the nanoworld, genomes, the environment, the atmosphere, and the cosmos. In Fiscal Year 2014, over 33,000 researchers from academia, industry, and government laboratories, from all fifty states and the District of Columbia, utilized these unique facilities to perform new scientific research. Each of these facilities generates vast amounts of scientific data, and thanks to advances in technology, the size, rate, and complexity of this data is rapidly increasing. A growing concern is that advances in the science programs will not be able to keep pace with increasing data rates due to a lack of resources, the need for research and development of tools as well as with platforms and infrastructures needed to manage, analyze, and act on the growing collections of data. All of the above are needed to derive novel scientific insights from data.

With the goal of understanding, inventorying, and articulating the data-centric needs and challenges of the experimental and observational science (EOS) community in DOE-SC, the Office of Advanced Scientific Computing Research (ASCR) held a workshop, from 29 September 2015 through 1 October 2015 in Bethesda, MD. The workshop report [4] consists of inputs from representatives from a sampling of DOE-SC EOS facilities and researchers in mathematics and computer science. The findings of the workshop, drawn from detailed discussions of participants who reviewed a wide range of exemplary *science use cases*, indicate that there are acute and urgent needs regarding the management, analysis, and visualization of experimental and observational data (EOD) collected and generated by EOS at DOE-SC user facilities.

The science needs articulated in the workshop report, along with the findings, recommendations, and detailed discussion of issues, collectively are consistent with the vision articulated in the National Strategic Computing Initiative (NCSI) [19], [13] and the Big Data Research and Development Initiative [24], [18]. The workshop report also describes multiple opportunities for cultivating a research, development, and deployment path that will help realize that vision. Specifically, the science use cases reveal a trend toward the *convergence of data and computing*: data- and compute-centric needs and suggests that opportunities in these research areas are increasingly intertwined, interrelated, and symbiotic. Advances in our ability to collect data will require advances in computational capabilities to understand, preserve, share, and make optimal use of data, and can positively impact the quality and value of our science by improving the quality and reusability of the data we collect.

This paper makes two primary contributions to the eScience community. First, it contains a canonical science use case that captures many of the high-level characteristics of how large-scale EOS projects in DOE-SC make use of the EOD they collect. This canonical use case is a composite sketch drawn from eleven specific use cases provided by the DOE-SC EOS community. Second, this paper presents a set of data-centric challenges, distilled from the eleven science use cases in [4],

TABLE I
THE ELEVEN USE CASES PROVIDED BY THE DOE-SC EOS COMMUNITY.

1	Environmental Molecular Sciences Laboratory
2	Climate Simulation and Analysis
3	Atmospheric Radiation Measurement Climate Research Facility
4	Advanced Light Source
5	Linac Coherent Light Source
6	Oak Ridge National Laboratory Neutron Sources
7	Scanning Probe and Electron Microscopies
8	Advanced Photon Source
9	Deep Underground Neutrino Experiment
10	Open Numerical Laboratories
11	DOE HEP Cosmic Frontier

that DOE-SC EOS projects face now and in the future. These two contributions are a summarization of ideas presented in the September 2015 ASCR EOD workshop report [4], which contains additional qualitative and quantitative information about the specific EOS projects.

We begin with a discussion of a canonical use case scenario (§II) to provide context and orientation to DOE-SC science facilities. That section lays the groundwork for the subsequent sections (§III–§IX), which in turn present the primary findings of the workshop. We discuss related work (§X) before making concluding remarks.

II. A CANONICAL SCIENCE USE CASE SCENARIO

The canonical science use case in this section is a composite of eleven use cases (see Table I) submitted by the DOE-SC EOS community. Those eleven use cases span a diversity of science topics from the DOE-SC offices of Biological and Environmental Research (BER), Basic Energy Sciences (BES), and High Energy Physics (HEP). Prior to the workshop, EOS researchers from those areas provided detailed information specific to their EOS project(s) that answered the following set of questions:

- **Present- or near-term issues.** Each EOS project provided a description of the science facility, how the facility or experiment “does science” with the EOD they collect, and a “flowchart” (verbal or pictorial) describing the data lifecycle starting with data acquisition and including all processing stages, all the way through dissemination.
- **Future issues.** Each EOS project provided information from the same categories for both the present- or near-term issues as well as looking ahead, usually 3–5 years, into the future.
- **Data lifecycle.** Each EOS project provided information about how data is used and the key issues that need to be addressed, at each of the primary data lifecycle stages, in both present and future scenarios.
- **Data requirements.** For each stage in the data lifecycle, each EOS project provided information about data “speeds and feeds,” throughput requirements, and specific

data-centric capabilities needed for the specific science use case.

- **Impediments, gaps, needs, and challenges.** Each EOS project provided a list of data-centric impediments or barriers they presently face and expect to face in the future.

The canonical use case below identifies the major stages in the data lifecycle and the types of processing that might take place there. The challenges faced by EOS projects in achieving these objectives are the subject of later sections.

Figure 1 is an illustration that gives a general overview of the major data lifecycle and processing stages in DOE-SC EOS projects. Although Figure 1 is a generalization of the eleven use cases in the workshop report, it captures all their major thematic elements. The following subsections discuss these major thematic elements.

A. Data Products

First and foremost, the primary objective for these EOS products is to support scientific research that generates large amounts of data from experiments and observations. In some cases, this EOD will be turned into “a product” that is then given to a single principal investigator (PI). In other cases, data products are created and shared by an entire community. In all cases, as elaborated below, these EOS projects have clear requirements for carefully capturing *provenance* (i.e., information about the conditions under which data is collected), what types of processing it underwent, and so forth, and preserving this information in the form of *metadata*. There are numerous challenges associated with growing data size and complexity, and all EOS projects point to the fact that EOD may have a long lifespan, which in turn creates challenges regarding long-term data storage and dissemination. These challenges are all the subject of later sections in this paper.

B. Use of HPC Computing Facilities

The background in Figure 1 is shaded to indicate that EOS projects may do some data processing “close to” the instrument at the science user facility, while some of the processing requires, due to data size or other factors, access to and use of large-scale HPC computing facilities. The balance point, between computing at science user facilities and at HPC facilities, highly varies from project to project and depends on specific project needs. In later sections, we delve into the challenges of EOS projects’ use of HPC computing facilities, which span diverse topics ranging from meeting time-sensitive computational requirements, to long-term data storage and dissemination.

C. Data Processing, Analysis, and Understanding

Hamming’s statement that “the purpose of computing is insight, not numbers” [14] applies equally to EOS projects and their use of data. Figure 1 shows that operations on data, which include various types of processing, such as filtering, reduction, cataloging, analysis, and provenance capture, occur at multiple stages in the data lifecycle.

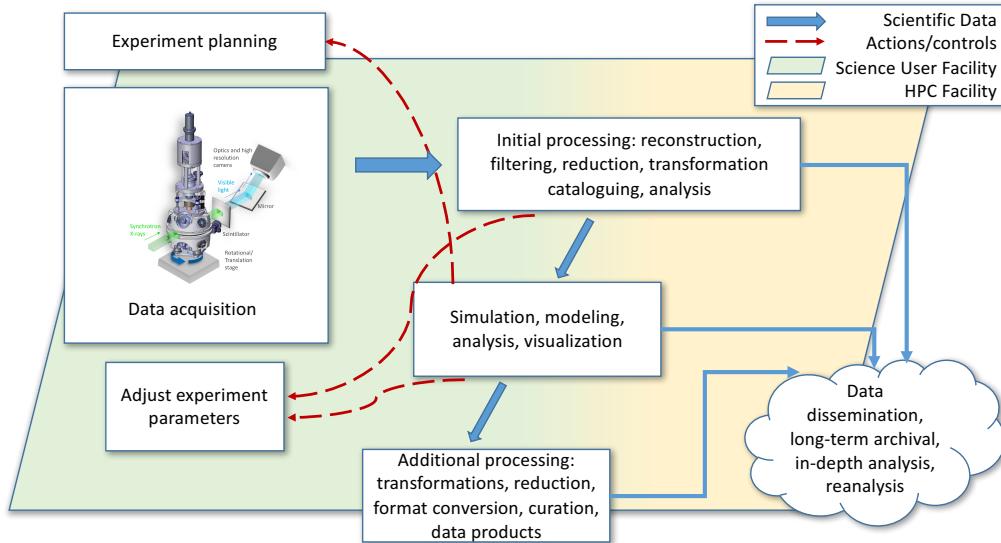


Fig. 1. An illustration showing the various stages in the EOS data lifecycle.

D. Data Sharing and Collaboration

Although it may not be readily apparent from Figure 1, the workshop's science use cases reveal a theme that collaboration and sharing are central to EOS science. Data produced by EOS projects is almost always turned into data products that are disseminated to a single PI, to a small group, or to a much larger, geographically distributed community. To be useful, however, such data needs to be curated and made easily understandable and accessible by all users.

Efficient use of shared data requires adequate metadata, if only to inform the community about the precise conditions under which it was collected. Typically, any given data set will be interpreted with respect to prior knowledge or will be combined with previously collected data. Thus, EOS data sets are usually a part of a much larger research study in which the metadata needed to set the context of the data is as important as the data itself. In addition, almost all EOS project data must be extensively processed by specialized software packages to convert the raw instrument data into a form that can be used in broader scientific studies.

E. Uses of EOD

Although Figure 1 shows a generalization of major data lifecycle stages, the paragraphs below summarize a few of the many uses of EOD identified in the eleven scientific use cases presented at the workshop.

a) Data products from EOS: In the early stages of experiment planning, a single PI (or community of investigators) tends to focus on a particular science mission or objective. The experiment is conducted, and the "first light" output of these experiments consists of a data product of some type. Here, the phrase "data product" could mean something relatively narrow and focused, such as a lab analysis of a sample, or it could

mean something much broader, such as a large collection of data from multimodal sources like a sky survey.

b) Primary analysis: With the data product(s) in hand, EO researchers then proceed to perform in-depth analysis as they focus on hypothesis testing, verification that the experiment went as intended, and/or any number of other tasks within the primary mission's focus.

Some of this analysis processing may require non-trivial steps, including assimilation with simulation or model-based approaches and statistical inverse problems, in order to extract key measurements from the EOD. These analysis results may produce outputs that could become data products in their own right, such as catalogs of features or properties extracted from the EOD.

c) Reanalysis: After an experiment and its initial study are completed, the resulting primary and supplementary data products may have a long lifespan. Over time, there may be the possibility for new scientific inquiry focusing on new hypotheses not envisioned at the time of the original experiment. For example, the Sloan Digital Sky survey (first light in 1998 [22]) is still operating today and its data are actively used by a worldwide science community.

d) Reference data sets: Once generated, a given data set may serve as an invaluable reference in many different ways in the future. Here, a significant issue is that not every EO data set will be equally used by the public, and over the years some of them will fade into irrelevancy while others emerge as a community reference. It is these latter data sets that need to be kept for a long time, even if just for comparison and reference. For such collections, which are used by many different refereed publications, reproducibility of these analyses will become another reason to keep the data, even when better and higher resolution alternatives become available.

e) *Broad dissemination of results:* In 2013, the Office of Science and Technology Policy issued a memorandum directing federally funded programs to make the results of their research freely available to the public, generally within one year of publication [12]. This broad, public access to research data of all types, and EOD in particular, could have far-reaching impacts. It could entail uses of EOD that go far beyond those envisioned in our 2015 workshop report.

III. CHALLENGES OF EXPLODING DATA SIZE, RATE, COMPLEXITY, AND DIVERSITY

Data size and rate of collection at science user facilities are growing at a rapid rate. Each of the eleven EOS use cases provide details about expected and anticipated growth in data rates. One of the primary drivers for increasing data size is the increase in the resolution of the instrument sensors and detectors. From these summaries of projected growth rates, we see a near future where individual facilities, of which there are dozens, are each generating collections of data in the range of tens to hundreds of petabytes per year. These projections suggest, when integrating across the entire program, that these *science user facilities will be soon collectively acquiring exabytes of data per year*. Affordable data storage, effective data access, distribution, and curation, and meaningful analysis are key challenges that these facilities increasingly face.

All EOS projects represented at this workshop are having difficulty coping with the demands and opportunities that the flood of data offers. The complexity of the data, new challenges in analytics and visualization, difficulties in capturing sufficient metadata, and ease-of-use problems are impediments to use and adoption of many types of data-centric tools and infrastructure, hampering the effort to harness the wealth of data in the service of scientific discovery.

A key limitation today is our [in]ability to analyze and visualize the acquired data due to its volume, velocity and variety.

—Advanced Photon Source [3]

The problems associated with data size and velocity are compounded when a given EOS community relies on multiple instruments, such as the DOE-SC's Cosmic Frontier projects, which carry out sky surveys (e.g., the Sloan Digital Sky Survey [22]) using multiple instruments. These are expected to produce data estimated to be on the order of hundreds of petabytes. These data need to be available to the research community over a long period of time. Survey data with a long shelf life can be very valuable because they can be mined and analyzed in many different ways, thereby providing a stable resource for developing new approaches for data exploration and analysis, in addition to the intrinsic science value.

The growth in data volumes creates challenges beyond issues of processing and storage, but also for data transfer, particularly in experiments that rely on real-time feedback to

the instrument operator. This problem is complicated even further by the fact that many experiments can be run concurrently with parallel data flows coming from independent detectors.

Another challenge associated with increased data size and complexity is the need to support data integration and data discovery through the appropriate collection and management of metadata and derived data products associated with experimental and simulation studies.

An ongoing concern in EOS projects is the need to detect errors in automated, high-throughput workflows. There is a clear convergence here between computing and data, where computational methods can be brought into the picture to ensure the best possible data are collected during an experiment. In some cases, errors occur during data acquisition. These errors can be mitigated and/or corrected after taking data through advanced algorithms that can model the dynamical effects of the acquisition instrument to produce a data set with minimized and/or quantified error.

A related concern is the loss of science and opportunities for science discovery due to data loss. One example is the Cosmic Frontier projects, where data loss can occur in studies of transients because of possible inefficiencies in detection technology, classification algorithms, and lack of follow-up resources. Other issues that prevent making use of the complete data set are technical issues such as lack of understanding of foregrounds, incomplete modeling of the atmosphere, and detector noise.

Although coping with the increasing size and rate of data inflows from experiments and observations is challenging, there is a corresponding set of issues at the other end of the data pipeline. EOS projects typically also produce data products that are derived from raw EOD and, in many cases, from the results of numerical calculations. Some data products are produced for individual users, while other data products are intended to be used by entire communities or as reference data sets. As mentioned above, it is necessary to have a clear record of information (metadata) about the data for these data products to be useful and usable, as well as a long-term plan for addressing the archiving, curation, and Dissemination of these data products.

IV. SCIENCE USE OF LARGE-SCALE HPC FACILITIES

EOS projects' use of tools and facilities, which are designed for HPC workloads, have realized varying degrees of success. Meeting the challenges of the explosion of data from EOS projects requires computational platforms, networking, and storage of greater capacity and lower latency, along with software infrastructure suited to their needs. However, existing HPC platforms and software tools are designed and provisioned for high-concurrency HPC workloads, single-project data products, and comparatively simpler data needs. Future HPC architectures are expected to be more I/O-challenged.

EOS projects look to large-scale HPC computing facilities to help serve workloads that can be characterized as: requiring fast turnaround for computing tasks, having processing

pipelines that are distributed in nature and involve the movement of very large amounts of data, long-term storage of data, as well as providing access to data to a potentially diverse set of stakeholders and consumers.

Each beamline operates with unique capabilities and an independent scientific mission. ... Computational needs and strategies may differ considerably across beamlines, but computation is required for nearly every aspect of the facility.

—Advanced Photon Source

The issue of fast turnaround is so significant that we expand on related findings below. In brief, EOS projects using instruments such as beamlines, require computational resources within minutes, or perhaps seconds, of when data are generated, which is incompatible with the queued structure employed on today's leadership-class HPC machines.

The EOS science use cases presented at the workshop describe variants of data handling and processing activities that can be characterized as distributed computing models. The typical design pattern involves first collecting data at the instrument, performing some processing close to the instrument, then moving data to a large-scale facility for more lengthy calculations and preparation of data products, followed by dissemination of data products. The way each project implements this pattern varies according to their needs and available resources.

As another example, the Deep Underground Neutrino Experiment (DUNE, [8]) experiment presently uses a combination of local, on-site computing and HPC facilities at Fermi National Accelerator Laboratory, which also is expected to host a full replica of data recorded by the prototype instrument. DUNE plans to keep full data replicas off site for redundancy, as well as to opportunistically leverage computing resources, including those outside of DOE-SC. DUNE is targeting the design and development of project-wide software infrastructure designed to maintain portable and accessible software that can be used at any particular institution and run transparently on modern grid and/or cloud resources as part of a distributed-processing, data-centric workflow.

Procedures for moving data from place to place, including tools for automating [a] resilient workflow for orchestrating distributed data-related operations are a bottleneck.

—Accelerated Climate Modeling for Energy project [1]

There is a clear need for community- or facility-centric data repositories for data storage and dissemination with sufficient bandwidth and with an easy interface for interacting with tools for data analytics. The most useful platforms need to

be massively parallel to support a combination of visualization and analysis tools. It is very likely that DOE facilities (both HPC facilities and science user facilities) will take on a significantly larger role in data archiving, transfer, and analysis. It is also possible that commercial cloud resources could become a major resource in these areas—although several outstanding questions remain (e.g., cost models, data archiving and transfer); this disruptive possibility needs to be continuously explored. The main new hardware trend of interest for DOE facilities—in the relatively near-term—is the evolution and integration of HPC systems within a data-centric usage model.

The needs of data sharing sites are quite distinct from those simply designed to store or analyze data. Data sharing and dissemination software must have robust features in searching for specific data types and for linking the data to people, studies, scientific fields, and published results.

V. TIME-SENSITIVE COMPUTING

Many projects have time-critical data needs (e.g., arising from human-in-the-loop experiment steering). These projects require a low-latency, high-throughput infrastructure for data movement, analysis, processing, and storage. However, computing platforms currently available to such researchers are insufficient in capacity or turnaround. The lack of large and capable facilities tuned to EOS needs are common across many disciplines.

Many EOS projects use, or hope to use, large-scale HPC platforms and high-speed networking to do real-time processing of experiment data. The desire is for high-throughput, fast turnaround to enable adjustment of experiment parameters while the experiment is in progress, thereby maximizing the scientific outcomes of the experiment.

For example, at the very first stages of the analysis workflow, scanning electron microscopy projects—such as those at the Center for Nanophase Materials Sciences [5]—are interested in collecting full detector response at the fastest meaningful rates in order to assess tool performance and adjust parameters on-the-fly. Fast visualization schemes would also be useful for monitoring samples and quality of output signals.

VI. THE RISK OF UNUSABLE DATA

Scientific data is increasingly at risk of being unusable, untraceable, or unreproducible. Without adequate metadata and provenance, scientific data has limited usefulness because its origins are undocumented and unknown, thereby limiting the ability to validate results or to make use of such data for other purposes. However, today the capture of these critical information often relies on manual, non-digital and non-sharable approaches, hindering scientific discovery particularly in increasingly high-velocity, high-volume data environments.

In some projects, data-centric operations, which involve management, analysis, movement, and distribution, are the responsibility of an individual user, with whatever limited knowledge and capability is available to them. As a result, only a fraction of collected data is ever analyzed, and only a

fraction of that data is ever published and made available for community-wide use.

One very real consequence of current data management systems is that most of the data is difficult to use by anyone other than the original group that generated it. This problem must be solved if making data publicly available is intended to have any useful purpose. In addition, much necessary metadata is never collected because of the lack of understanding of what is required for data sharing by the primary investigator(s) and the lack of easy-to-use tools to capture it. The overall cost and complexity of metadata recording and consolidation is currently prohibitive, which is a primary reason why metadata is rarely collected. Unfortunately, this means that the associated data cannot be easily exchanged or reused. Systematic collection of the metadata that describes the provenance of stored data is typically inadequate, limiting the integrity, traceability, and reproducibility of research products.

...relevant data should be made available to the scientific community after some amount of time. But more than data preservation is required—proactive data curation is necessary for the data to be really useful. ...The benefit of curation would be to reduce duplication of effort in data creation, but also for the re-use of data for further high quality research.

—Advanced Light Source [2]

There is interest in having access to data after the current research is published. Such access needs to ensure that enough metadata is stored so that the data can be analyzed appropriately. Although certain classes of metadata (e.g., the who, the when) can be captured automatically, there is a need to capture the reason why certain aspects of an analysis or data transformation or reduction operation was performed. This information needs to be provided by the data generators and archived with the data so that subsequent access is useful, and can be utilized by researchers beyond the group that acquired it originally.

VII. THE CENTRAL ROLE OF COLLABORATION AND SHARING

Collaboration and sharing of data, tools, and methodologies are central to modern EOS projects, yet there is insufficient infrastructure to facilitate such interactions. Common tools and methodologies for sharing and collaboration in data-intensive sciences have not been widely developed, deployed, or adopted. The limit is generally not simply data transfer, but rather a lack of widely-used tools to produce and consume well-characterized data collections that include the desired level of annotation, metadata, and provenance. Collaborations also require an ability to share software tools, source code, data models, and formats, and to provide workflows that are reproducible. Beyond established collaborations, there is a clear need to share tools and approaches between groups and

disciplines to minimize the unnecessary duplication of effort. In many cases, existing tools are inadequate or too difficult to use.

By their nature, the mission focus for EOS projects is to collect data, and to share it. This theme is present in all the use cases presented at the workshop. The projects differ in some key ways: some projects' immediate focus is on sharing data with a primary principal investigator or group, while others focus on sharing data with larger communities. Although making data accessible for download over the Internet lowers the barrier to accessibility for a potentially large number of consumers, doing so is only a small part of a larger landscape of collaboration and sharing.

Understanding the process of how science is actually done, what information needs to be captured and where the data is generated are key issues that must be addressed to enable effective data sharing.

—Environmental Molecular Sciences Laboratory [11]

The use cases provide several compelling reasons why collaboration and sharing is important. First, sharing software has the potential effect of reducing costs, particularly of software development. The idea is that redundancy of effort—software development—is reduced when key methods and tools can be reused across different projects. Sharing data, particularly curated data, would be to reduce the duplication of effort in data creation, as well as for data re-use for further high-quality research. Another benefit would be that it could lead to more algorithms and software being made available to the community, as researchers write code that can be benchmarked and used against curated data.

One concept that is central to achieving the ability to share data and tools is the idea of community-centric, or “standard” data models and formats for both data and metadata. The climate community, for example, has realized a degree of success in sharing data as well as software tools for working with data, due to its use of a data model or format that has broad community support. This idea is identified as a need or an impediment in several of the science use cases.

Current technologies are inadequate for sharing [...] data between group members. The EOS community needs a more fluid means for sharing data and working together.

—Environmental Molecular Sciences Laboratory

The use cases identify several different ideas that are needs for or impediments to collaboration and sharing. One is that the issue of data and software sharing does not have program-wide visibility. As a result, progress in this space is *ad hoc*,

with solutions for distributing data or software emerging on a per-facility or per-PI basis, with little or no coordination. Thus, there are many different sources of data and software, resulting in duplication of effort as well as a high barrier to finding data or software. Impediments that are most detrimental are related to the issues of data sharing and collaborating in large groups, methodological transparency, and dissemination and archival capabilities. Data and/or software that is “custom” and not curated is unlikely to be widely used. Better methods—interfaces and software tools, infrastructure—are needed to search and subset data without having to download an entire data set.

Impediments that are most detrimental are related to the issues of data sharing and collaborating in large groups, methodological transparency, and dissemination and archival capabilities.

—Environmental Molecular Sciences Laboratory

There is a deep interplay between the topic of collaboration, and the related but orthogonal topics of the overall data lifecycle, the usability of data and the associated challenges of metadata/provenance capture and long-term data archival and curation, and EOS’s use of computing and data facilities. The interactions between these different focus areas is made more challenging by the rapid rate of growth in data size and the rate of data acquisition. Stated differently, successes in these related areas are building blocks for success in all areas of collaboration and data sharing.

VIII. DATA LIFECYCLE

The term data lifecycle refers to all stages of data collection, movement, processing, analysis, management, curation, and sharing. Data collected by observation or experiment has a potentially long lifespan, and a potentially large set of consumers, but there presently is no solution or approach for data curation, quality management, and long-term distribution within DOE-SC that is generally and broadly applicable. At the same time, data retention policies at DOE-SC computing facilities are not designed for long-term retention nor for widespread dissemination.

EOS projects have “data lifecycle” needs that are significant, well defined, and that go well beyond what is provided by the current set of programs and projects in the ASCR computing facilities and research portfolio. EOS can have a long lifespan, yet there is no program-wide view or approach for the long-term curation, storage, and dissemination of such data; one EOS project indicated that it relies on whatever capabilities are provided by journals in association with publications as its solution to this problem.

...our only archival process right now is that provided by the published journal.

—Spallation Neutron Source [23]

Two key motivations for retaining data sets for a long period of time are for having a reference data set for use in evaluating the effectiveness of new methods over time, and for the opportunity for new discoveries not originally foreseen at the time the data was collected. Over the years, some data sets produced by simulation will emerge as a community reference. For such collections, which are likely to be used by many different authors in refereed publications, reproducibility of these analyses will become another reason to keep the data, even when better and higher resolution alternatives become available. For example, in tomography, the resulting tomogram is of comparable size to the raw images, which are also usually retained for the purposes of comparing the results of different tomographic reconstruction algorithms. Results from projects like sky surveys may initially be focused on a few key science missions, but over time, a diverse set of science activities can be carried out with substantial discovery potential.

Current strategies for managing (accessing, processing, and keeping track of) the large number of data products are awkward at best, requiring a combination of methods. Science user facilities like the Advanced Photon Source (APS, [3]) do not provide a centralized and robust long-term data archive, since this service is categorized as a user responsibility. Most science user facilities have no explicit method for long-term archival and curation, and this is identified as an impediment. It is likely that in the future, science user facilities will be called upon to provide long-term storage and archival services. One stop-gap approach for long-term archival is to rely on the infrastructure and policies provided by the journal where a given paper is published. A welcome addition in the data universe would be a centralized DOE-SC facility that provides a mechanism for data archiving and retrieval, that could be provided as an option to users at low or no cost.

Providing more access to the data, in a manner that can be used by more scientists, will improve efficiency, increase the impact of the science, and result in more papers per experiment.

—Spallation Neutron Source

The issues related to data lifecycle management are broad, and cut across many different areas. We have identified challenges and research needed in areas germane to this topic: the automation of processing stages and automated data movement in EOS, data storage and retrieval, metadata and provenance, software engineering and infrastructure, data curation, collaboration, and interaction with computing service facilities.

IX. THE CENTRAL ROLE OF SOFTWARE

Software is a critical element for all EOS projects in all aspects of working with data and in meeting the challenges of increasing data size and complexity. Software is used for collecting, processing, and analyzing data, for preparing data products, and for automating complex multi-stage operations that may span distributed resources.

An important outcome of the workshop is the recognition of common needs across all the science domains. Although the computing needs of EOS projects vary from one project to the next, all EOS projects need computing and data storage/dissemination, along with a sustainable software ecosystem that can evolve over time to accommodate its data-centric requirements. This finding suggests that priority attention should be directed toward approaches that develop and support solutions that can be widely used by many EOS projects and facilities.

Software methods need to be capable of supporting the time-critical analysis and display tasks summarized above. Software methods, such as advanced algorithms for analysis, play a key role in improving the quality of data collected during an experiment, thereby improving the efficiency and quality of science. The idea is that EOS projects like beamlines require computational resources within minutes or perhaps seconds when data are available and cannot abide with the queued structure employed on leadership machines.

Because of the central role that software plays in nearly all aspects of working with data, EOS projects are particularly vulnerable to inefficiencies and increased costs resulting from inadequate software. For example, time inefficiencies result when data-centric pipelines and data movement activities must be executed manually rather than being automated and resilient. Inefficiencies in cost can result when a customized software component is created for one user but is not readily customizable or applicable to other users in the same facility, or across other science facilities.

The biggest challenge to the facility is how to create the scientific software needed to run it: software for improving the experimental process; for implementing beamline data movement and reduction workflows; to perform preliminary quality assurance, visualization and reduction; for data analysis and interpretation; for automating analysis workflows and distribution to users.

The most serious impediment the APS encounters is a lack of a DOE-wide view of software needs across the BES mission. Since each lab has its own portfolio of responsibilities, it devotes resources to those goals.

—Advanced Photon Source

Software technology also plays a key role in encapsulating complexity and as an enabling technology. EOS projects want and need to be able to make use of advances in computational architectures, such as using HPC platforms for performing data-centric operations on larger data and with a faster turnaround. However, developing software for those platforms is often beyond the reach of a typical scientist-developer who may not have HPC software development skills. When it comes to the development of HPC code, there are fewer tools that ease the process for scientist-software developers (as opposed to computational experts) to transition from prototype code to HPC production code. The same idea extends to other areas of technology, such as creating data-centric pipelines that span distributed resources.

Increasingly, both simulations and experimental data analysis are elements of integrated workflows, which should resiliently automate key components of the data-handling pipeline, from collection to processing, analysis, archival, and dissemination. Many contemporary EOS projects articulate the need to combine computing with the experiment in real time, so as adjust experimental parameters on-the-fly to obtain the best possible data and science result from the experiment. Software methods need to be capable of supporting the time-critical analysis and display tasks summarized above. Meeting these challenges will require more powerful computing and networking infrastructure combined with a capable, robust and sustainable software ecosystem focusing on EOS needs. The flood of data available now and in the near future presents an opportunity that can be met only through concerted, coordinated, and sustained efforts to improve the software tools, methods, and facilities (computing, data) available to the EOS community.

Software is “digital data” that needs to undergo the rigors of curation, in the same way as do data from experimental and observational sciences, to facilitate its long-term archival preservation and widespread dissemination. To be useful, software, like other forms of digital data, needs to have associated metadata along with documentation and examples of use. To be long-lived, it needs to be supported, maintained, and disseminated, something that is often not a part of the cost model. Wide-scale adoption of common tools will only occur if users are convinced that those tools will be maintained. Several use cases pointed to the desire to distribute software together with data, to expand the usability of data and to promote the repeatability of results as well as to promote the use of reference data and methods. One use case pointed out that their only practical avenue for doing so was to rely on the archival capabilities provided by the journal where results are published. The issues and motivations related to software curation, preservation, and dissemination are similar to those for other types of scientific data.

X. RELATED WORK

The increasingly important role of data and data understanding in science is well recognized [16] along with attendant challenges [17]. Within the scientific community, there

have been concerted efforts to capture data-centric challenges and requirements. In 2004, the *The Office of Science Data-Management Challenge* report [10] suggests that successfully addressing challenges related to the management and understanding of data one of the primary obstacles facing modern science.

These topics were revisited a decade later in 2013 in the *Data Crosscutting Requirements Review* [15], with many of the same observations as the 2004 report, but that also draw distinctions to the differences and similarities between data-centric issues specific to the sciences and those elsewhere, such as finance, health care, and so forth. Also in 2013, a National Research Council report *Frontiers in Massive Data Analysis* [6] summarized analysis challenges in particular, and highlighted the cross-disciplinary knowledge required to address these challenges. These issues are not unique to Department of Energy science areas. The biological and medical sciences, for example, have also grappled with issues of large-scale distributed data sets that require significant computational resources for their analysis along with potential approaches [20] that are similar to those described in the 2015 EOD report.

From 2015, *The Future of Scientific Workflows* report [7] examines data issues from a distributed computing perspective, where scientific data must be handled and acted on as part of an end-to-end workflow, which may involve processing stages cited at multiple geographic locations. The 2015 EOD report, which is the subject of this paper, takes a broader view than these earlier reports, to include the symbiotic relationship between the science user facilities and computational and networking infrastructure, along with key research targets in mathematics and computer science, that would come into play to service those data-centric workloads and meet scientific knowledge discovery challenges.

The increasing interplay between high-performance computing and large-scale data challenges is the subject of a recent article by Reed and Dongarra [21]. They suggest that advances in scientific research require growth in infrastructure for both computing and for analyzing data. They also point out the divergence in software architectures used for traditional HPC computing and for Big Data handling, due in part to the economics of widespread use of commodity components in industry.

The 2015 EOD report provides perspective on some of the challenges facing data-intensive science use of HPC facilities that have historically serviced computationally focused workloads. These challenges go far beyond what processors, interconnects and software components are in the system, and includes issues related to how knowledge is gained from data, how data is specifically used by several different data-intensive EOD science projects, and the specific impediments standing in the way of scientific progress in the face of a daunting deluge of scientific data.

XI. CONCLUSIONS AND SUMMARY

Like many other areas of science, the science user facilities and projects operated by the U.S. Department of Energy's Office of Science are challenged by an increasing deluge of scientific data. The focus of this paper is on summarizing some of the key findings of a workshop convened in September 2015 to identify key challenges and new research and development needed to meet those challenges. Whereas this paper focuses on the key findings and challenges, that workshop report provides considerably further depth to the findings, as well as information contributed by workshop participants that articulates new research and development needed to meet those challenges.

The 2015 EOD report contains a diversity of specific recommendations for next steps towards meeting data-centric needs. Space constraints here limit a substantial discussion of these recommendations. In brief, they encompass both high- and detailed-level views of the problem space. For example, high-level issues focus on program-wide coordination of responses and efforts so as to overcome the fragmented and duplicative nature of how science user facilities approach cultivating solutions to data-centric challenges. Detail-level views focus on, for example, the need for new computational and mathematical methods for approximate and fast modeling and analysis calculations to meet the emerging use case of time-constrained use of HPC facilities so as to tune experiments on-the-fly with the objective of obtaining higher-quality experimental data.

Realizing success in meeting many of the data-centric challenges discussed here will likely entail efforts that require program-wide coordination and visibility. Many of the problems are simply too large and too far reaching to be tackled by an individual investigator or individual science user facility. For example, the objective of long-term data archival and dissemination is something that is needed by all EOS projects, and is something that requires significant forethought and attention to how EOD may be used in the future. This observation is applicable far beyond the set of science use cases represented in the workshop's report, and can help guide and shape how data-centric problems in all sciences might be approached elsewhere.

The science use cases in our workshop report reveal a trend toward the *convergence of data and computing*. The uses cases articulate both data- and compute-centric needs, and suggest that opportunities in these research areas are increasingly intertwined, interrelated, and symbiotic. Advances in our ability to collect data will require advances in computational capabilities to understand, preserve, share, and make optimal use of data, and can positively impact the quality and value of our science by improving the quality and reusability of the data we collect.

ACKNOWLEDGMENTS

We are grateful to more than 50 workshop participants and 16 other co-authors for their invaluable contributions to the report summarized in this paper. This work, and the associated workshop, was supported in part by the Director, Office of Science, Office of Advanced Scientific Computing Research,

of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231, program manager Dr. Lucy Nowell.

REFERENCES

- [1] Accelerated Climate Modeling for Energy (ACME). <http://climatedevelopmentscience.energy.gov/projects/accelerated-climate-modeling-energy/>, last accessed May 2016.
- [2] Advanced Light Source. <https://www-als.lbl.gov/>, last accessed May 2016.
- [3] Advanced Photon Source. <http://www.aps.anl.gov/>, last accessed May 2016.
- [4] E. Wes Bethel and Martin Greenwald (eds.). Report of the DOE Workshop on Management, Analysis, and Visualization of Experimental and Observational Data – The Convergence of Data and Computing. Technical report, Lawrence Berkeley National Laboratory, Berkeley, CA, USA, 94720, May 2016. LBNL-1005155.
- [5] Center for Nanophase Materials Sciences. <https://www.ornl.gov/facility/cnms/>, last accessed May 2016.
- [6] National Research Council. *Frontiers in Massive Data Analysis*. The National Academies Press, Washington, DC, 2013.
- [7] Ewa Deelman and Tom Peterka (editors). The Future of Scientific Workflows, April 2015. http://science.energy.gov/~media/ascr/pdf/programdocuments/docs/workflows_final_report.pdf.
- [8] Deep Underground Neutrino Experiment. <http://www.dunescience.org/>, last accessed May 2016.
- [9] US DOE. Office of Science User Facilities. <http://science.energy.gov/user-facilities/user-facilities-at-a-glance/>, last accessed May 2016.
- [10] Richard Mount (editor). The Office of Science Data Management Challenge – Report from the DOE Office of Science Data Management Workshop Series. Technical report, Stanford Linear Accelerator Center, 2004. SLAC-R-782, <http://www.er.doe.gov/ASCR/ProgramDocuments/Docs/Final-report-v26.pdf>.
- [11] Environmental Molecular Sciences Laboratory. <https://www.emsl.pnl.gov/>, last accessed May 2016.
- [12] Expanding Public Access to the Results of Federally Funded Research. <https://www.whitehouse.gov/blog/2013/02/22/expanding-public-access-results-federally-funded-research/>, last accessed May 2016.
- [13] FACT SHEET: National Strategic Computing Initiative, July 2015. https://www.whitehouse.gov/sites/default/files/microsites/ostp/nsci_fact_sheet.pdf, last accessed March 2016.
- [14] Richard W Hamming. *Numerical Methods for Scientists and Engineers*. Dover Publications, Inc., New York, 2nd edition, 1986.
- [15] Bruce Hendrickson and Arie Shoshani (editors). Data Crosscutting Requirements Review. Technical report, U. S. Department of Energy, Office of Science, April 2013.
- [16] Tony Hey, Stewart Tansley, and Kristin Tolle, editors. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, Redmond, Washington, 2009.
- [17] H. V. Jagadish, Johannes Gehrke, Alexandros Labrinidis, Yannis Papakonstantinou, Jignesh M. Patel, Raghu Ramakrishnan, and Cyrus Shahabi. Big data and its technical challenges. *Communications of the ACM*, 57(7):86–94, 2014.
- [18] Tom Kalil and Fen Zhao. Unleashing the power of big data, April 2013. <https://www.whitehouse.gov/blog/2013/04/18/unleashing-power-big-data/>, last accessed March 2016.
- [19] Barak Obama. Executive Order – Creating a National Strategic Computing Initiative, July 2015. <https://www.whitehouse.gov/the-press-office/2015/07/29/executive-order-creating-national-strategic-computing-initiative/>, last accessed March 2016.
- [20] Russell A Poldrack and Krzysztof J Gorgolewski. Making big data open: data sharing in neuroimaging. *Nature Neuroscience*, 17(11):1510–1517, October 2014.
- [21] Daniel A. Reed and Jack Dongarra. Exascale Computing and Big Data. *Commun. ACM*, 58(7):56–68, June 2015.
- [22] Sloan Digital Sky Survey. <http://www.sdss.org/>, last accessed May 2016.
- [23] Spallation Neutron Source. <https://neutrons.ornl.gov/sns/>, last accessed May 2016.
- [24] Rick Weiss and Lisa-Joy Zgorski. Obama Administration Unveils "Big Data" Initiative: Announces \$200 Million in New R&D Investments, March 2012. https://www.whitehouse.gov/sites/default/files/microsites/ostp/big_data_press_release_final_2.pdf, last accessed March 2016.

Developing a Citizen Science Web Portal for Manual and Automated Ecological Image Detection

Marshall Mattingly III*, Andrew Barnas†, Susan Ellis-Felege†, Robert Newman†, David Iles‡, Travis Desell*

Department of Computer Science*, Department of Biology†

University of North Dakota

Grand Forks, North Dakota 58202

Email: marshall.p.mattingly@und.edu, andrew.barnas@my.und.edu, susan.felege@email.und.edu,
robert.newman@email.und.edu, tdesell@cs.und.edu

Department of Wildland Resources and the Ecology Center‡

Utah State University

Logan, Utah 84322-5230

Email:david.thomas.iles@gmail.com

Abstract—Image recognition is challenging in the field of wildlife ecology as samples of a specific species can be rare, making manual detection cumbersome. With over 2,060,000 images taken from motion-sensor trail cameras and unmanned aerial vehicle flights, a touch enabled web interface has been developed to allow citizen scientists and ecologists to categorize positive samples. To minimize categorization errors, the same images are shown to multiple separate users. The observations of each user are then compared using two novel validation strategies: percentage of overlapping area and maximum corner distance. Two novel methods for the extraction of final images from validated results are presented and compared as well: average corner points and area intersection. These methods were evaluated using a set of 142 images with a total of 811 observations of objects generated by citizen scientists that were manually inspected for ground truth. Results show that for this research a maximum corner distance of 10 pixels and the use of area intersection provided the best extracted imagery for future use as training and testing data by computer vision methods.

I. INTRODUCTION

The Wildlife@Home project started as a hub for citizen scientists, non-field scientists who wish to help the processing of data, to help catalog video for a number of sub-projects with the goal of producing machine learning algorithms to automatically detect different species and their behaviors. With over 100,000 hours of video, cataloging would be impossible to do in a timely manner with ecologists and other trained professionals. This time limitation necessitated the creation of the Wildlife@Home web portal to allow citizen scientists to help. To date, citizen scientists have watched 34,234 hours of video and recorded 62,395 observations. This paper builds upon the Wildlife@Home project by allowing citizen scientists to review images, not just video. Citizen scientists can catalog different species and parameters in wildlife imagery to allow for machine learning algorithms to detect objects from the imagery in the future.

There are three current image sub-projects on Wildlife@Home: (1) trail camera (trailcam) images of common eiders in the Hudson Bay area of Canada; (2)

trailcam images of lesser snow geese in the Hudson Bay; and (3) unmanned aerial system (UAS) imagery taken from 75m, 100m, and 120m altitudes in the Hudson Bay consisting primarily of lesser snow geese. These sub-projects present different challenges, with the trailcam images being prone to obscuration by growing vegetation throughout monitoring and the UAS images having relatively small top-down species objects for detection. Even further, some species have evolved with *cryptic coloration* (camouflage), making them very difficult to identify. It is therefore important to ensure that citizen scientists are given proper training for both types of images to ensure objects extracted from the imagery can be used to train machine learning algorithms to automate the object detection in the future.

Similar to the video on Wildlife@Home, there are too many images for ecologists to manually inspect in a reasonable amount of time, with over two-million images currently collected and an estimated two-million more to collect each summer. Citizen scientists provide an excellent resource to review and catalog the images; however, given that the citizen scientists are not guaranteed to be field experts, precautions must be taken to minimize the potential of bad data entering the training dataset for machine learning.

This paper is organized into the following sections. Section II presents related works using crowd sourcing and citizen scientists to review field-specific imagery or video; Section III describes the different image data sets in use by Wildlife@Home; Section IV details the methodology describing the components of the image review interface and algorithms used to extract objects from the observations made by citizen scientists; Section V presents results showing a comparison of the algorithms used for object extraction; and the paper concludes with Section VI discussing the results and future areas of research.

II. RELATED WORK

Crowd sourcing has been successfully used by citizen science projects to tackle problems requiring human feedback. GalaxyZoo [1], [2] has had great success in using volunteers to classify galaxies in images from the Sloan Digital Sky Survey [3]; and PlanetHunters [4] has been used to identify planet candidates in the NASA Kepler public release data. More recently, Snapshot Serengeti [5] has been created to classify images from camera traps in the Serengeti National Park.

In avian ecology, Cornell's NestCams project [6] has provided an outstanding resource for environmental education and gained popularity through the use of nest cameras to attract the public's interest in environmental science. NestCams primarily focuses on public outreach where video is collected opportunistically from cameras primarily installed in bird houses, capturing a variety of cavity-nesting species. The CamClickr project has sparked applications of nest video archives for education in collegiate-level animal behavior courses [7]. More recently, eBird [8] is a citizen science project which allows users to upload observations of birds through handheld devices, providing spatio-temporal information about the bird distribution and abundance. Aside from CamClickr, few ecological citizen scientist projects have the volunteers reviewing images or videos to be used by researchers.

III. WILDLIFE@HOME IMAGE DATA SETS

A. Hudson Bay Trail Cameras

Data for the Hudson Bay Project (a long-term ecological monitoring program in the Arctic that makes use of remote sensing technologies such as trail cameras and unmanned aircraft systems) was collected as part of David Iles' PhD research through the Utah State University [9]. Cameras were deployed to learn about predators destroying nests and in particular to estimate changes in predation rates by species such as polar bears (*Ursus maritimus*) which are coming ashore earlier than historically due to climate change [10], [11]. A total of 85 cameras and approximately 100 nests are monitored in each year between the common eiders and lesser snow geese (see Fig. 1).

The images are collected by placing trail cameras on a stake at a nest located along the Hudson Bay near Churchill, Manitoba. Because eiders and snow geese are colonial nesters, multiple nests can be monitored using one camera. Time-lapse photography was used to monitor behaviors such as attendance patterns (time birds spend incubating the nest rather than off the nest for their own self-maintenance). Therefore, one image was taken every 2 minutes and then if movement was detected at the nest, a burst of 30 photos would be taken. Currently there are over 2 million photos, resulting in over 2 terabytes of data.

B. Hudson Bay UAS Imagery

Manned aerial wildlife surveys are one of the leading causes of death for wildlife biologists, accounting for 66% of work related mortality from 1937-2000 [12]. As such, UAS have

become an area of high interest for performing these surveys more safely [13]–[23]. In summer 2015, a Trimble UX5¹ fixed wing UAS was flown at Wapsusk National Park in Manitoba, Canada. Survey flights were conducted from 11-24 June during the nesting season of lesser snow geese and common eiders and 11-15 July during the post-hatch period, producing over 60,000 images which resulted in over 1 terabyte of data. Figure 2 shows three examples of UAS imagery. Flights were conducted at 75m, 100m and 120m above ground level. A 16 megapixel Sony red, green, blue (RGB) camera flown in the Nader position was used to capture imagery along pre-defined transects with 80% overlap.

From the overlapping images, 10 mosaics were created using Trimble Business Center (version 3.51)². These mosaics were then processed into 100 smaller images for presentation on the Wildlife@Home image review interface due to size limitations.

IV. METHODOLOGY

In order to automate detection of objects from wildlife imagery within Wildlife@Home, three facets were required: (1) an interface to show images for citizen scientists to create their observations; (2) algorithms to pair up observations of a single object from unique citizen scientists for each image; and (3) extraction of sub-images representing objects based on the set observations to create a training dataset of images for machine learning.

To create a consistent collection of citizen scientist observation data, several students from the Biology and Computer Science departments of UND were invited to review the mosaic UAS imagery on 18 April, 2016. In total, 8 out of 10 mosaics were reviewed, with 4 mosaics having observations from 2 unique citizen scientists and 1 mosaic having observations from 3 unique citizen scientists.

A. Web Interface Creation

The first step of this project was to create an accessible and easy-to-use web-based interface for citizen scientists to review images from the over two-million set of wildlife imagery from trailcam and UAS flights in the Hudson Bay area of Canada. Wildlife@Home already had a robust interface for citizen scientists to watch and categorize events and species from video content. This video review interface served as a guide for the creation of the image review interface seen in Fig. 3. The major obstacles to overcome during development of the image review interface were: (1) images can be significantly larger than the typical viewport of a desktop monitor, especially in the case of the UAS imagery; (2) objects that are too small do not make good candidates for machine learning; and (3) the interface must be accessible on a wide-variety of operating systems and interfaces, including touch interfaces.

Users are able to double-click or double-tap to add boxes, definite rectangular sections of the image the user can define as containing an object of interest which are movable and

¹<http://uas.trimble.com/ux5>

²<http://www.trimble.com/Survey/trimble-business-center.aspx>



Fig. 1. Imagery taken using motion sensing cameras along the Hudson Bay (credit David Iles). Photos can have multiple nests, *e.g.*, there are two nests in these pictures, one in the foreground and another in the background. The photos also have varying lighting conditions. Original photo resolution is 2048 x 1536 pixels.



Fig. 2. Imagery taken using an unmanned aerial system with a camera attached along the Hudson Bay. Photos can have multiple lesser snow geese and include both white and blue-phase lesser snow geese. Each image has two snow geese, with the right image having a blue phase snow goose to the right of a white phase snow goose. Original photo resolution varies and each image shown is 200x200 pixels and centered on the objects.

Image #: 4017308
Species ?
Interface ?

Selection 1

Species: Lesser Snow Goose, Blue Phase

On Nest?: No nest

Comments:

Skip
There's Nothing Here
Submit



1

1.40x

Fig. 3. Screenshot of the web-based image review interface on Wildlife@Home. The left half of the interface shows, from top to bottom: (1) the current image number; (2) a discussion button that allows users to share the image on the Wildlife@Home forums; (3) help buttons for both species identification and interface elements; (4) selection input to identify the species and parameters for the observation; (5) comment area to include text comments about the image; and (6) buttons to skip the image and get a new image, indicate that there is nothing in the image, or submit the observations for the image. The right half of the interface includes the HTML5 Canvas element with the image for review, scrollbars on the right and bottom indicating the current location within the image, and a current scale multiplier in the bottom right.

resizable, to the image to signify a object, as shown surrounding the blue-phase lesser snow goose on the right-middle of Fig. 3. After the box is overlayed on to the image, the user is then able to specify the species of the creature and whether or not it is on a nest, as shown on the left side of Fig. 3. After the citizen scientist presses the submit button in the bottom of Fig. 3, the location, dimension, species, and on-nest status for each observation is stored in a database for processing by the object extraction algorithms before being used as the input training dataset for machine learning. If the citizen scientist does not notice any objects in an image, they may instead click the nothing here button in the bottom of Fig. 3 which is also stored in the database.

To overcome the issue of images significantly larger than a typical viewport, a script was written to break larger images down into 25 or 100 smaller images, depending on the size of the image. The resultant partial images are constrained to approximately 1280 pixels wide, allowing them to fit within a typical 1920 pixel width viewport, and stored in the reference database with their offset and dimension within the master image. In the case that the viewport is still smaller than the image being presented, an HTML5 Canvas element allows the user to pan along the image in all four directions and zoom in and out, as needed. The bars above and below the image in Fig. 3 indicate the current location within the image and the current zoom level is shown in the bottom right, with numbers greater than 1.0x indicating a zoom in and number less than 1.0x indicating a zoom out.

Machine learning relies on quality representative data to be able to automatically detect objects in future images. This means images that are too small, determined to be less than 25-pixels square in the case of the UAS imagery, make it difficult for our machine learning algorithms to detect objects. This issue is dealt with by enforcing a hard-limit on the minimum box size in the interface and providing clear instructions to citizen scientist on what should and should not be categorized which can be reviewed by pressing the interface help button in the left-top of Fig. 3. As shown in the results section, it is still not guaranteed that all users will adhere to these instructions. Presenting the same image to three users and selecting the objects for machine learning based on all responses helps to mitigate single-user errors.

An HTML5 Canvas library, built on JavaScript, was developed for the interface and released as a separate open-source project. Creating a stand-alone library was important as the interface is used in multiple sub-projects on the Wildlife@Home website, specifically trailcam images from the Hudson Bay, UAS imagery from the Hudson Bay, and mosaic images created from the UAS imagery in the Hudson Bay. The modularity of a separate library allows each of these project to control specific parameters and callbacks related to the project, while allowing all cross-platform development and optimization to happen separately. Using HTML5, JavaScript, and the open-source touch-interface library Hammer.JS³, the library is

able to accept input from multi-touch devices (phones, tablets) and traditional mouse / keyboard devices while scaling to the viewport. Future work for the library will be focused on adding further mobile interface improvements, specifically related to layout.

B. Matching Observations

Before accepting observations by citizen scientists for inclusion in the output training dataset, we first compare all the observations in a single image with the observations of other citizen scientists on the same image. For this paper, all observations were produced on mosaic images from the UAS imagery and collected during a single event on 18 April, 2016 by students from the Biology and Computer Science departments at UND. For future work, three citizen scientists are required to view a single image before processing. To compare observations against each other, two different algorithms were developed: an area-overlap algorithm that compares the total amount of area shared between two observations and returns the percentage overlap, as shown in Eq. 1, and a corner-point distance algorithm that calculates the maximum distance between the each of the four corners of two observations, as shown in Eq. 2. Each of these algorithms is tested in the results section with different parameter values and compared against the actual matching observations as determined by manual examination.

$$\begin{aligned} l_{\text{intersect}} &= \min \|x_{12}, x_{22}\| - \max \|x_{11}, x_{21}\| \\ h_{\text{intersect}} &= \min \|y_{12}, y_{22}\| - \max \|y_{11}, y_{21}\| \\ A_{\text{intersect}} &= \max \|0, l_{\text{intersect}} * h_{\text{intersect}}\| \\ A_{\text{union}} &= A_1 + A_2 - A_{\text{intersect}} \\ A_{\text{overlap}} &= \frac{A_{\text{intersect}}}{A_{\text{union}}} \end{aligned} \quad (1)$$

where,

$l_{\text{intersect}}$ is the length of the intersection of two observations
 $h_{\text{intersect}}$ is the height of the intersection of two observations
 $A_{\text{intersect}}$ is the area of intersection of two observations
 A_{union} is area of the union of two observations
 A_1 is the area of observation 1
 A_2 is the area of observation 2
 A_{overlap} is ratio of overlap of two observations

$$\begin{aligned} c_0 &= \sqrt{(x_{11} - x_{21})^2 + (y_{11} - y_{21})^2} \\ c_1 &= \sqrt{(x_{12} - x_{22})^2 + (y_{11} - y_{21})^2} \\ c_2 &= \sqrt{(x_{12} - x_{22})^2 + (y_{12} - y_{22})^2} \\ c_3 &= \sqrt{(x_{11} - x_{21})^2 + (y_{12} - y_{22})^2} \\ c_{\text{max}} &= \max \|c_0, c_1, c_2, c_3\| \end{aligned} \quad (2)$$

where, c_0, c_1, c_2 , and c_3 are the distances between the top-left, top-right, bottom-right, and bottom-left corners of two observations, respectively, and c_{max} is the maximum distance between any two corners.

³<http://hammerjs.github.io/>

C. Extracting Objects

After objects identified by citizen scientists have been paired together, a single sub-image is extracted to represent the object as agreed upon by the matches. Automated object recognition relies heavily on an input dataset that minimizes negative space around an object while maximizing the amount of the object in the data. This is particularly challenging when using data gathered by citizen scientists due to potential lack of expertise and human error. Two methods were developed to attempt to satisfy the needs of the machine learning algorithms.

The average object extraction method, algorithm seen in Eq. 3, takes the average of the all four corners of a set of paired observations and uses that as the corner points to extract a partial image with the given extents to represent the object detected. This method gives equal weight to all observations by the citizen scientists, which allows a single observation to skew the resultant object. When the set of paired observations has little variability, the average method produces images that represent the objects well; however, as the variability of paired observations increases, the results can trend toward the poor observations, producing images that poorly represent objects by having too much negative space.

$$\begin{aligned}(x_0, y_0) &= \left(\frac{\sum_{i=1}^n x_{i0}}{n}, \frac{\sum_{i=1}^n y_{i0}}{n} \right) \\ (x_1, y_1) &= \left(\frac{\sum_{i=1}^n x_{i1}}{n}, \frac{\sum_{i=1}^n y_{i1}}{n} \right)\end{aligned}\quad (3)$$

where,

- (x_0, y_0) is the top-left corner of the object
- (x_1, y_1) is the bottom-right corner of the object
- n is the number of observations in the matched set
- x_{i0} is the left-most extent of the i^{th} observation
- y_{i0} is the top-most extent of the i^{th} observation
- x_{i1} is the right-most extent of the i^{th} observation
- y_{i1} is the bottom-most extent of the i^{th} observation

The intersection object extraction method, algorithm seen in Eq. 4, uses the intersection of the set of paired observations to extract a partial image with the given extents to represent the object detected. This method gives the most weight to the smallest observation in the matched set, which allows a single observation to skew the resultant object. When paired observations are all large enough to surround the actual object being detected, the intersection method ensures that the negative space is minimized, producing excellent object results; however, if a single observation is smaller than the object being detected, the resultant partial image will be an object with too little positive space. As shown in the results section, the drawbacks of the average algorithm are more severe than the drawbacks of the intersection algorithm.

$$\begin{aligned}(x_0, y_0) &= \left(\max_{i=1}^n \|x_{i0}\|, \max_{i=1}^n \|y_{i0}\| \right) \\ (x_1, y_1) &= \left(\min_{i=1}^n \|x_{i1}\|, \min_{i=1}^n \|y_{i1}\| \right)\end{aligned}\quad (4)$$

where the variables definitions are the same as in Eq. 3.

V. RESULTS

Two observation equivalence algorithms were run against a set of 142 images with a total of 811 observations of objects made by citizen scientists. Two observations by two different citizen scientists are said to match if they represent the same detected object, while observations that do not have any matches are considered non-matched. Each image in the set comes from mosaics created from UAS imagery from the Hudson Bay, Canada and was viewed and cataloged by 2 or 3 citizen scientists using the Wildlife@Home image review interface. The observations were manually inspected to record the actual observation equivalence set, which was used to determine the ratio of correct matches and non-matches, as well as the false positives and false negatives of the two observation equivalence algorithms.

The matched ratio and non-matched ratio are key values used throughout this section to determine the success rate of the observation equivalence algorithms. The matched ratio is the number of matched observation pairs created by an observation equivalence algorithm divided by the actual number of matched observation pairs as determined via manual examination. The non-matched ratio is the number of observations created by an observation equivalence algorithm that are not matched with any other observations in the set divided by the actual number of non-matched observations as determined by manual examination. A ratio of 1.0 indicates perfect matching or non-matching while a ratio below 1.0 means the object equivalence algorithm is missing matches or non-matches and a ratio above 1.0 means the object equivalence algorithm has too many matches or non-matches.

A. Overall Observation Equivalence Algorithm Accuracy

The overall accuracy of the two observation equivalence algorithms is shown in Table I with multiple parameter values which are then compared against the actual observation equivalence set as determined by manual examination of the observations. The area overlap equivalence algorithm detected 0.88 times as many observation matches as actually exist (matched ratio) while failing to match 1.34 times as many objects as should actual fail to match (non-matched ratio). As the area overlap requirement is increased, there is a linear decrease in number of matches and an exponential increase in non-matched observations, as in Fig. 4.

The corner-point equivalence algorithm starts at a maximum distance of 5-pixels between all corners, resulting in a low 0.59 matched ratio and a high 3.75 non-matched ratio, shown in Table I. However, increasing the threshold to 10-pixels gives a dramatic improvement resulting in a 0.95 matches ratio and

TABLE I
OBSERVATION EQUIVALENCE ALGORITHM RESULTS AND COMPARISONS AGAINST THE ACTUAL OBSERVATION EQUIVALENCE SET AS DETERMINED BY MANUAL EXAMINATION OF THE 811 OBSERVATIONS FROM 142 IMAGES IN THE TEST DATASET

Algorithm	Matches	Non-Matched	Matched Ratio	Non-Matched Ratio	False Positives	False Negatives
Actual	400	91	1.00	1.00	0	0
Area (50%)	352	122	0.88	1.34	0	54
Area (60%)	329	159	0.82	1.74	0	92
Area (70%)	266	282	0.66	3.10	0	214
Area (80%)	186	440	0.46	4.83	0	370
Area (90%)	81	649	0.20	7.13	0	566
Point (5px)	238	341	0.59	3.75	0	272
Point (10px)	379	106	0.95	1.16	0	24
Point (15px)	404	91	1.01	1.00	8	0
Point (20px)	414	91	1.03	1.00	18	0

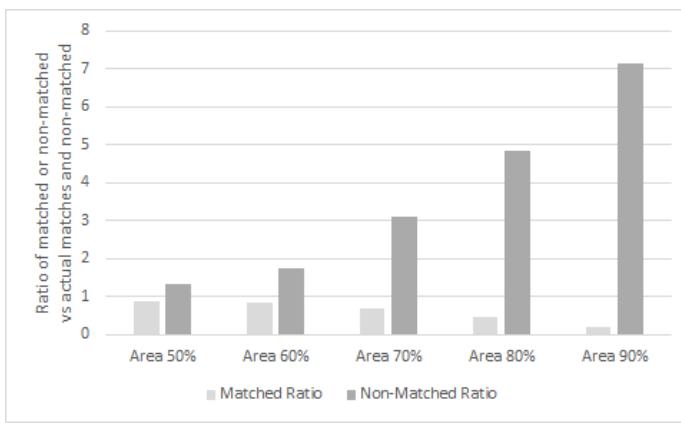


Fig. 4. This chart shows the matched and non-matched ratio of the area overlap equivalence algorithm when compared against the actual matches and non-matches as determined by manual examination of the observations. The area overlap equivalence algorithm is shown with 50%, 60%, 70%, 80%, and 90% minimum area overlap requirements.

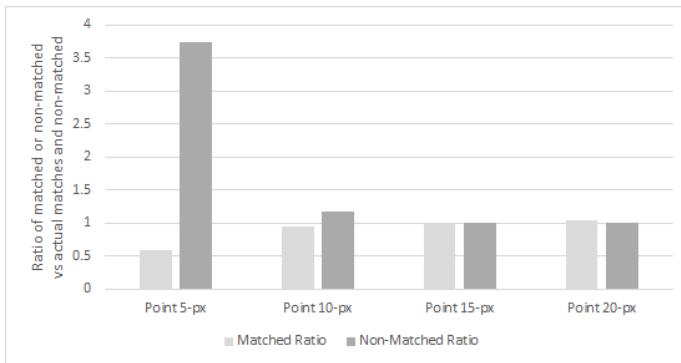


Fig. 5. This chart shows the matched and non-matched ratio of the corner-point equivalence algorithm when compared against the actual matches and non-matches as determined by manual examination of the observations. The corner-point equivalence algorithm is show with 5-px, 10-px, 15-px, and 20-px maximum distances.

a 1.16 non-matched ratio, significantly better than best result from the area overlap equivalence method (0.96 and 1.34 matches and non-matched ratios, respectively). While the area overlap algorithm provided an exponential change between cutoffs, the corner-point algorithm actually plateaus at 10-pixels, as in Fig. 5. In fact, the corner-point algorithm begins to over-match as the cutoff is increased, meaning the parameters are too loose, resulting in observations being matched not only to their true matches as confirmed by manual examination, but to additional observations nearby.

B. False Positive / Negative Analysis

The previous section looked at the overall accuracy between the two observation equivalence algorithms with various parameters when compared to the results of manual examination. These results provide an excellent understanding of basic effectiveness of the algorithms with different parameters; however, to determine the best algorithm to use on a large dataset, the error-rates, seen as false positives and false negatives in Table I, of the equivalence algorithms must also be known. False positives indicate observations that were matched by the algorithm, but do not match as determined by the manually examined observation equivalence set. False positives result in the over-matching seen in the corner-point algorithm when the cutoff was 15-pixels or more and will pollute the training dataset with incorrect or misaligned objects. False negatives are observations that the algorithm failed to match, but should be matched according to the manually examined observation equivalence set. False negatives leave the training dataset with too few objects for machine learning. The effects of these two must be taken into strong consideration when determining the appropriate observation equivalence algorithm.

The area overlap observation equivalence algorithm in Fig. 6 does not produce any false positives for the given parameters and observation set; however, the false negatives increase exponentially as the area overlap algorithm is tightened. The corner-point equivalence algorithm in Fig. 7 starts with a significant amount of false negatives and no false positives. The false negatives disappear quickly while the false positives actually increase from 0 to 18 at a corner-point distance of 20-pixels. This phenomenon confirms the over-matching in Fig. 5.

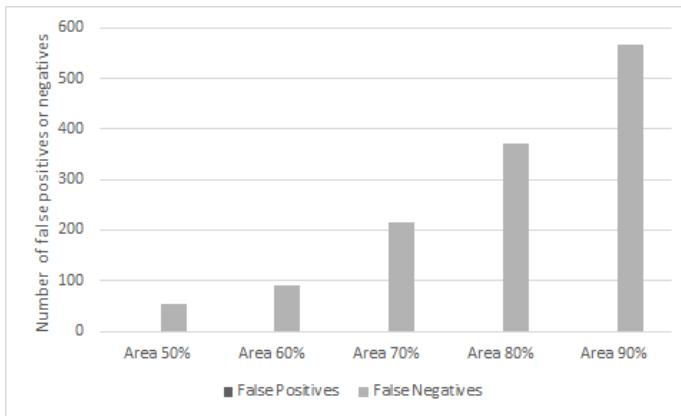


Fig. 6. This chart shows the total number of false positives and false negatives created by the area overlap equivalence algorithm when compared against the actual observation equivalence set as determined by manual examination. The area overlap equivalence algorithm false positives and false negatives are shown with 50%, 60%, 70%, 80%, and 90% minimum area overlap requirements.

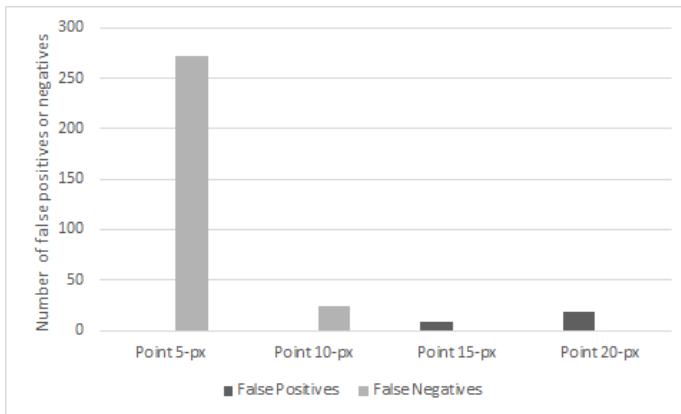


Fig. 7. This chart shows the total number of false positives and false negatives created by the corner-point equivalence algorithm when compared against the actual observation equivalence set as determined by manual examination. The corner-point equivalence algorithm false positives and false negatives are shown with 5-px, 10-px, 15-px, and 20-px maximum distances.

False positives correlate well with the overall matched ratio of a given observation equivalence algorithm. If the matched ratio is below 1.0, false positives are held to 0; however, as the matched ratio increases above 1.0, known as over-matching, false positives occur. Similarly, false negatives correlate well with the overall non-matched ratio of a given observation equivalence algorithm. As the non-matched ratio increases, the false negatives similarly increase with an r^2 of 0.997. This suggests that maximizing the matched ratio below 1.0 and minimizing the non-matched ratio above 1.0 should result in the best equivalence algorithm, resulting in the corner-point equivalence algorithm at 10-pixels as the best overall method. However, it will be important to continue testing with more algorithms, more parameters, and different datasets to ensure that the results hold.

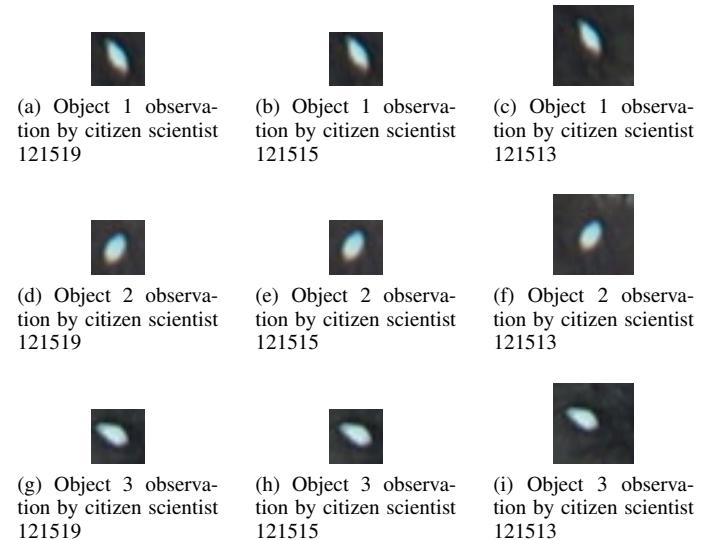


Fig. 8. Comparison of the partial images created from three sets of observations made by three different citizen scientists on the same three objects. The left and middle observations by citizen scientists 121519 and 121515, respectively, are similar and would make for decent inputs for machine learning. The right observations by citizen scientist 121513 are skewed and create too much negative space for machine learning.

C. Observation Variability by User

Variability in observations on the same object made by multiple citizen scientists makes pairing observations and selecting the correct bounds that represent the actual object difficult. A single citizen scientist can create an observation that may potentially skew the object extraction by creating bounds that are too large and potentially not centered on the object, as in Fig. 8.

To examine the variability in citizen scientist observations, a subset of 9 images with a total of 25 objects observed by three citizen scientists was extracted from the manually inspected observation equivalence set. This allowed for the comparison of the corner-point distances of the three matched observations between the citizen scientists, *i.e.*, the observation pair of citizen scientists 1 and 2; the observation pair of citizen scientists 1 and 3; and the observation pair of citizen scientists 2 and 3 for the same object. This data for each image is enumerated in Table II, where Observation Pair (1, 2), Observation Pair (1, 3), and Observation Pair (2, 3) are the data for the pair of observations of citizen scientists 1 and 2, the pair of observations of citizen scientists 1 and 3, and the pair of observations of citizen scientists 2 and 3, respectively. The *Mean* and *STDEV* results were calculated for each matched observation pair set and included for analysis.

The mean corner-point distance for the matched observation pairs of citizen scientists 1 and 2 was 2.50 pixels with a standard deviation of 1.23 pixels, meaning citizen scientists 1 and 2 created similar observations of the 25 objects. The mean corner-point distance for the matched observation pairs with citizen scientist 3 was 8.99 pixels and 9.27 pixels with a standard deviation of 2.51 pixels and 2.36 pixels for the



Fig. 9. This figure shows the similarity between the average corner-points extraction method (left) and the area intersection extraction method (right) when the observations by citizen scientists have near identical bounds around the object detected.



Fig. 10. This figure shows the potential major difference between the average corner-points extraction method (left) and the area intersection extraction method (right) when the observations made by citizen scientists have greatly different bounds around the object detected. The average corner-points extraction method produces skewed bounds around the object with a large amount of negative space, while the area intersection extraction method produces bounds around the object with comparatively little negative space.

matched observation pairs with citizen scientist 1 and 2, respectively. Smaller corner-point distance average (less than a third) and standard deviation (less than half) of citizen scientists 1 and 2 when compared against their respective observations with citizen scientist 3, it is apparent that the citizen scientist 3 created poor observations of the objects. In fact, 20 out of 50 (or 40%) of the matched observation pairs with citizen scientists 3 had a corner-point distance greater than 10 pixels and would fail to be matched by the best performing observation equivalence algorithm from Table I.

D. Selecting Final Object for Machine Learning

The previous sections have been concerned with pairing observations identified by citizen scientists. After the observations have been paired, a singular partial image of the object is extracted from the image using a combination of the bounds in the observations. Similar to the area overlap and corner-point equivalence algorithm, two methods for selecting a single partial image representing an object are compared: (1) an image using the average corner-points of the paired observations; and (2) an image with the intersect of the paired observations.

The average corner-points method returns similar results, as in Fig. 9, to the intersect method when citizen scientists are consistent in placing boxes around objects. However, a single citizen scientist that produces a box significantly too large or too small can have negative impact on resultant partial image for the object, leaving the machine learning algorithm with an object that has too much negative space, as in Fig. 10, or not enough positive space.

The intersection method deals with this issue by creating the smallest box possible based on the input observations from the citizen scientists. This ensures that a citizen scientist who produces too large a box cannot negatively affect the resultant partial image for the object; however, a citizen scientist producing too small a box can still impact the quality

of the resulting objects by not producing enough positive space. Initial manual analysis indicates that citizen scientists produce more boxes that are too large than too small; in fact, there wasn't a single instance in the observation sets where the smallest observation in the set cut off more than a couple pixels of the object. This suggests that the intersection method is the most consistent method to ensure good partial images of objects for machine learning.

VI. CONCLUSION

Overall, citizen scientists did an excellent job agreeing on objects in a given image, with only 91 out of 811 (11.2%) observations failing to be matched to another observation. The corner-point object equivalence algorithm with a cutoff of 10-pixels did the best job pairing observations of a single object together, with a 0.95 matches ratio and 1.16 non-matched ratio. After observations are paired together, the intersection method produced the most consistent results to minimize the negative space around objects while maximizing the positive space of the object.

The 10-pixel equivalence algorithm will therefore be used to create a training dataset for a neural network for the automated detection of objects. After the neural network has been successfully trained, a BOINC⁴ client will be created to allow citizen scientists to provide computer resources to aid in the detection of objects in future imagery. The creation, training, and BOINC client for the neural network will be the focus of a complimentary paper.

Citizen scientist provide their time to help identify the objects in the imagery. However, as shown in the results sections, not all citizen scientists produce usable bounding boxes around objects. It is therefore important to both study the situations in which poor bounding boxes are created and to provide incentives to create better bounding boxes.

All citizen scientists responses in this paper were students at UND from multiple disciplines giving their time in return for free food. This incentive external to identifying objects could potentially create bias in some respondents; it is therefore the intention of the authors to keep a log of any such events where respondents are given an external reward for providing data and comparing the bounding boxes with those of citizen scientists who are providing their time and effort without external factors. Additional research of bias in respondents, especially consistent outliers, will be a focus of future research.

To provide an incentive to citizen scientists working in their free time a system of gamification will be implemented. Currently, citizen scientists on the Wildlife@Home project are given points based on computer resources provided and for identifying objects and events in video. After the neural networks are trained and automated, citizen scientists will be given points for: (1) any identified objects that make it into the training set (*i.e.* that another citizen scientist also identifies); (2) any identified objects that are confirmed by the neural

⁴<http://boinc.berkeley.edu/>

TABLE II
MEAN ACTUAL CORNER-POINT DISTANCE (IN PIXELS) BETWEEN OBSERVATION PAIRS OF THREE CITIZEN SCIENTISTS WHO MADE OBSERVATIONS ON THE SAME 25 OBJECTS FROM 9 IMAGES AS DETERMINED BY MANUAL EXAMINATION.

Image ID	Object #	Observation Pair (1, 2)	Observation Pair (1, 3)	Observation Pair (2, 3)
4016613	1	1.41	9.22	9.22
4016613	2	1	7.81	7.07
4016616	1	2.24	4.24	5
4016616	2	2	10.8	9.22
4016616	3	2	3.16	4.24
4016617	1	2	10.3	10.8
4016618	1	1.41	12.7	12.8
4016618	2	2.24	7.07	9.22
4016618	3	2	9.43	10
4016618	4	3.16	10	10.5
4016621	1	3	10	12
4016622	1	2.24	8.06	10
4016622	2	2	6.4	7.81
4016622	3	4.47	7.21	7.81
4016622	4	3	12.8	14.1
4016622	5	6.4	10	10.6
4016624	1	2.23	14.9	12.7
4016624	2	1.41	9.43	9.22
4016624	3	2.83	10.3	12.2
4016624	4	4.12	9.22	10
4016624	5	1.41	7.07	8.48
4016627	1	1.41	7.81	7.81
4016627	2	1.41	9.43	8.6
4016627	3	3.16	8.48	7.81
4016628	1	4	8.6	7.07
Total	25	62.55	224.44	234.28
Mean		2.50	8.98	9.37
STDEV		1.23	2.51	2.36

networks; and (3) providing computer resources via BOINC to aid in the automated detection of objects.

Further future work will be focused on: (1) resolving issues where only one or two citizen scientists make an observation of an object (is the object ignored because it was not observed by all three citizen scientists, or do we accept objects with at least two observers); (2) enhancing the web interface for mobile devices; (3) providing clearer instructions to citizen scientists to increase the probability of each citizen scientists making good observations; (4) comparing the citizen scientists against observations made by field experts to show that the citizen scientists produce results good enough for machine learning; and (5) providing a large annotated data release of training and testing imagery for the computer vision community.

ACKNOWLEDGMENTS

We appreciate the support and dedication of the Wildlife@Home citizen scientists who have spent significant amounts of time reviewing images and classifying objects within them. Funding was provided by North Dakota EPSCoR, the Hudson Bay Project, Central and Mississippi Flyways, National Geographic, and UND's College of Arts and Sciences.

REFERENCES

- [1] C. J. Lintott, K. Schawinski, A. Slosar, K. Land, S. Bamford, D. Thomas, M. J. Raddick, R. C. Nichol, A. Szalay, D. Andreescu, P. Murray, and J. Vandenberg, "Galaxy zoo: morphologies derived from visual inspection of galaxies from the sloan digital sky survey," *Monthly Notices of the Royal Astronomical Society*, vol. 389, no. 3, pp. 1179–1189, 2008. [Online]. Available: <http://dx.doi.org/10.1111/j.1365-2966.2008.13689.x>
- [2] C. Lintott, K. Schawinski, S. Bamford, A. Slosar, K. Land, D. Thomas, E. Edmondson, K. Masters, R. C. Nichol, M. J. Raddick, A. Szalay, D. Andreescu, P. Murray, and J. Vandenberg, "Galaxy zoo 1: data release of morphological classifications for nearly 900,000 galaxies," *Monthly Notices of the Royal Astronomical Society*, vol. 410, no. 1, pp. 166–178, 2011. [Online]. Available: <http://dx.doi.org/10.1111/j.1365-2966.2010.17432.x>
- [3] J. e. a. Adelman-McCarthy, "The 6th Sloan Digital Sky Survey Data Release, <http://www.sdss.org/dr6/>," July 2007, *apJS*, in press, arXiv/0707.3413.
- [4] D. A. Fischer, M. E. Schwamb, K. Schawinski, C. Lintott, J. Brewer, M. Giguere, S. Lynn, M. Parrish, T. Sartori, R. Simpson, A. Smith, J. Spronck, N. Batalha, J. Rowe, J. Jenkins, S. Bryson, A. Prsa, P. Tenenbaum, J. Crepp, T. Morton, A. Howard, M. Beleau, Z. Kaplan, N. vanNispen, C. Sharzer, J. DeFouw, A. Hajduk, J. P. Neal, A. Nemec, N. Schuepbach, and V. Zimmermann, "Planet hunters: the first two planet candidates identified by the public using the kepler public archive data," *Monthly Notices of the Royal Astronomical Society*, vol. 419, no. 4, pp. 2900–2911, 2012. [Online]. Available: <http://dx.doi.org/10.1111/j.1365-2966.2011.19932.x>
- [5] Lion Research Center, University of Minnesota, [Accessed Online, 2012] <http://www.snapshotserengeti.org/>.

- [6] R. Bonney, C. B. Cooper, J. Dickinson, S. Kelling, T. Phillips, K. V. Rosenberg, and J. Shirk, "A developing tool for expanding science knowledge and scientific literacy," *BioScience*, vol. 59, pp. 977–984, 2009.
- [7] M. A. Voss and C. B. Cooper, "Using a free online citizen-science project to teach observation and quantification of animal behavior," *American Biology Teacher*, vol. 72, pp. 437–443, 2012.
- [8] C. Wood, B. Sullivan, M. Iliff, D. Fink, and S. Kelling, "ebird: engaging birders in science and conservation," *PLoS biology*, vol. 9, no. 12, p. e1001220, 2011.
- [9] D. Iles, D. Koons, R. Rockwell, C. Mulder, and S. Ellis-Felege, "Unpublished data," 2013-2014.
- [10] L. J. Gormezano and R. F. Rockwell, "Dietary composition and spatial patterns of polar bear foraging on land in western hudson bay," *BMC ecology*, vol. 13, no. 1, p. 51, 2013.
- [11] ——, "What to eat now? shifts in polar bear diet during the ice-free season in western hudson bay," *Ecology and evolution*, vol. 3, no. 10, pp. 3509–3523, 2013.
- [12] D. B. Sasse, "Job-related mortality of wildlife workers in the united states, 1937-2000," *Wildlife society bulletin*, pp. 1015–1020, 2003.
- [13] M. Israel, "A uav-based roe deer fawn detection system," in *Proceedings of the International Conference on Unmanned Aerial Vehicle in Geomatics (UAV-g)*, H. Eisenbeiss, M. Kunz, and H. Ingensand, Eds, vol. 38, 2011, pp. 1–5.
- [14] B. E. Wilkinson, "The design of georeferencing techniques for unmanned autonomous aerial vehicle video for use with wildlife inventory surveys: A case study of the national bison range, montana," Ph.D. dissertation, University of Florida, 2007.
- [15] R. P. Breckenridge, M. Dakins, S. Bunting, J. L. Harbour, and S. White, "Comparison of unmanned aerial vehicle platforms for assessing vegetation cover in sagebrush steppe ecosystems," *Rangeland Ecology & Management*, vol. 64, no. 5, pp. 521–532, 2011.
- [16] W. Koski, P. Abgrall, and S. Yazvenko, "A review and inventory of unmanned aerial systems for detection and monitoring of key biological resources and physical parameters affecting marine life during offshore exploration and production activities," *IWC paper SC/61 E*, vol. 9, 2009.
- [17] P. Soriano, F. Caballero, A. Ollero, and C. A. de Tecnologias Aeroespaciales, "Rf-based particle filter localization for wildlife tracking by using an uav," in *On: 40th International Symposium of Robotics, Barcelona, España*, 2009.
- [18] M. Leonardo, A. M. Jensen, C. Coopmans, M. McKee, and Y. Chen, "A miniature wildlife tracking uav payload system using acoustic biotelemetry," in *ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, 2013, pp. V004T08A056–V004T08A056.
- [19] R. P. Breckenridge, R. Lee, A. Piscitella, and C. Eckersell, "Using unmanned aerial vehicles to assess vegetative cover in sagebrush steppe ecosystems," Idaho National Laboratory (INL), Tech. Rep., 2005.
- [20] J. Linchant, C. Vermeulen, J. Lisein, P. Lejeune, and P. Bouché, "Using drones to count the elephants: a new approach of wildlife inventories," 2013.
- [21] K. A. Steen, A. Villa-Henriksen, O. R. Therkildsen, H. Karstoft, O. Green *et al.*, "Automatic detection of animals using thermal imaging," in *International Conference on Agricultural Engineering*, 2012.
- [22] C. Cariappa, W. Ballard, S. Breck, A. J. Piaggio, and M. Neubaum, "Estimating population size of mexican wolves noninvasively (arizona)," 2008.
- [23] C. Vermeulen, P. Lejeune, J. Lisein, P. Sawadogo, and P. Bouché, "Unmanned aerial survey of elephants," *PloS one*, vol. 8, no. 2, pp. 1–7, 2013.

Applying Data Mining Methods for the Analysis of Stable Isotope Data in Bioarchaeology

Markus Mauder*, Eirini Ntouts†, Peer Kröger*, Christoph Mayr‡, Gisela Grupe§, Anita Toncal‡, Stefan Hözl¶

*Institute for Informatics, Data Science Lab, Ludwig-Maximilians-Universität München, Germany

Email: {mauder,kroeger}@dbs.ifi.lmu.de

†Faculty of Electrical Engineering and Computer Science, Leibniz Universität Hannover, Germany

Email: ntoutsi@kbs.uni-hannover.de

‡Institute for Geography, Friedrich-Alexander Universität Erlangen-Nürnberg, Germany

Email: c.mayr@lrz.uni-muenchen.de

§Bio-Center, Ludwig-Maximilians-Universität München, Germany

Email: {g.grupe,anita.toncal}@lrz.uni-muenchen.de

¶RiesKraterMuseum Nördlingen, Germany

Email: s.h@lmu.de

Abstract—Data science methods have the potential to benefit other scientific fields by shedding new light on common questions. One such task is choosing good features for analysis. In this paper, we introduce a data science framework that was designed to allow domain experts to consider their domain knowledge in assembling suitable data sources for complex analyses. The structure of experimental data as represented by a clustering is used to measure the relevance as well as the redundancy of each feature. We present an application of this technique to bioarchaeological data from a region in the European Alps, a transalpine passage of eminent archaeological importance in European prehistory, the Inn-Eisack-Adige passage, spanning Italy, Austria, and Germany. These results are applied to the task of provenance analysis. The application of the presented data mining technique leads to new insights which were not found using standard bioarchaeological approaches.

I. INTRODUCTION

Multivariate evaluation of measurement data has become common practice in many sciences. However, faced with multi-dimensional data many domain scientists struggle to understand analysis results. A common task in multi-variate data analysis is assessing which features to generate and consider in the first place. While there are fully automatic multi-variate feature analysis methods, they do not allow the domain scientist to evaluate a feature's merit in light of domain-specific considerations. In this paper we introduce a feature evaluation technique that allows domain scientists to understand each feature's role in the data distribution and to evaluate its importance for the analysis at hand.

The project which motivated this paper aims at the construction of a large scale isotopic map covering a specific transect across the European Alps, namely the Inn-Eisack-Adige passage via the Brenner pass. This transect has been in use at least since the Mesolithic and is therefore of eminent archaeological importance. The isotopic mapping of the transect aims at answering open archaeological questions related to

transalpine mobility and culture transfer¹. The term *isotopic landscape* describes *maps of isotopic variation produced by iteratively applying (predictive) models across regions of space using gridded environmental data sets*, whereby one common use of isoscapes is as a source of estimated isotopic values at unmonitored sites, which can be an important implementation for both local- and global-scale studies if the isoscape is based on a robust and well-studied model [1]. In bioarchaeology, variations of isotopes (atoms with different numbers of neutrons) are used to predict patterns that characterize the origin of geological and biological materials at a small spatial scale. Such isotopic maps are empirically generated by sampling the relevant environmental components and measuring their isotopic signatures. However, the vast majority of stable isotope studies in this field are small scale projects that lack the fundamental capabilities of prediction and modeling.

In this paper, we use isotope data to investigate the question which features should be measured in order to keep the costs for generating a reliable data source for this isotope map moderate. We describe a framework that was developed to solve this problem and, thus, supports the domain experts in making decisions during data generation. The data mining task was to discover which features were most relevant or most redundant and therefore irrelevant for analysis. The results are of high value for the domain scientists. Our framework for solving the aforementioned data mining task is based on a technique that explores the relevance and redundancy of individual variables to a clustering in comparison to a reference clustering. There is no obvious reference clustering, because no ground truth is known. However, domain specific knowledge and assumptions can be used to generate several *plausible* reference clusterings that are estimations of a ground truth. Thus, we explore how the relevance and the redundancy of single features behave under different ground truth

¹See www.for1670-transalpine.uni-muenchen.de

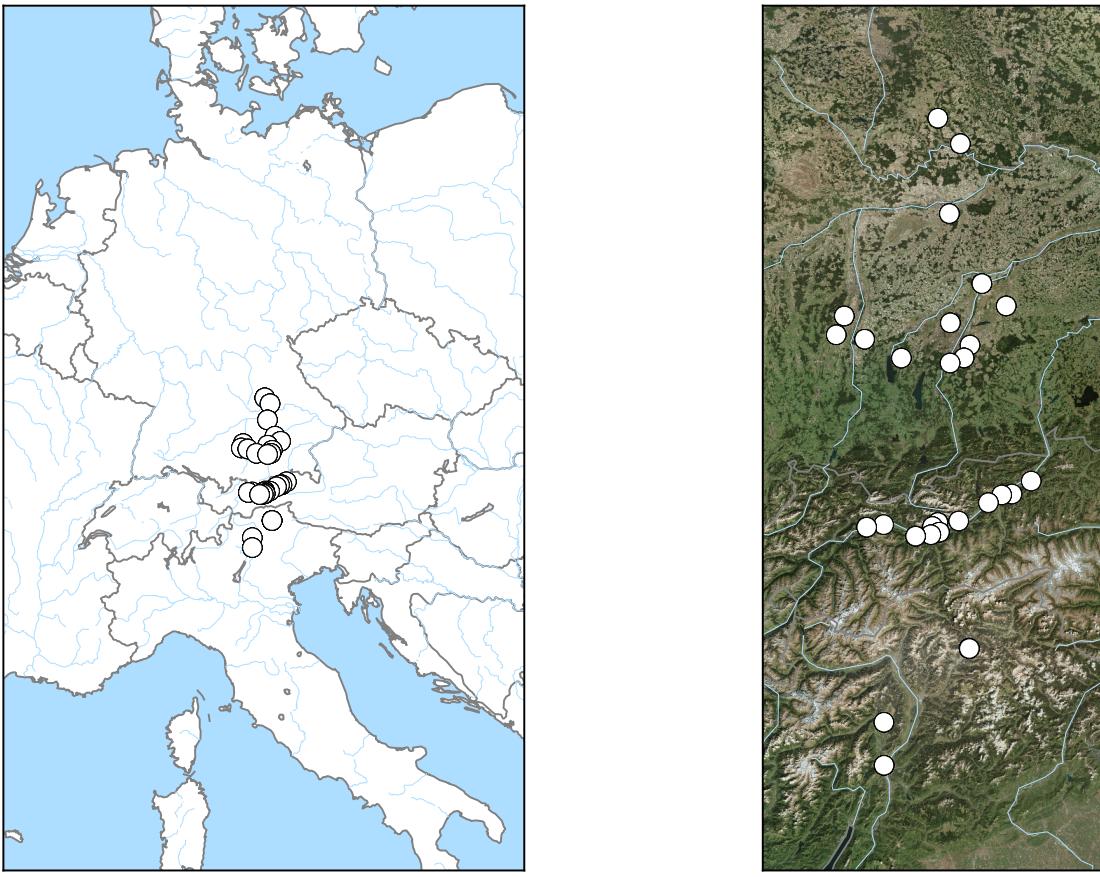


Fig. 1: Sampling sites across the transalpine Inn-Eisack-Adige passage. Data from these locations is used in the evaluation in Section III.

assumptions and derive conclusions from these observations. The results have been carefully discussed with the domain experts and the resulting conclusions confirm that we were able to benefit their work.

In summary, our contributions are as follows:

- We describe a new framework that solves a real problem in the application domain. The framework supports identifying relevant features that need to be measured and redundant features that need not to be measured. The framework is a real data science product, because it emerged in tight interdisciplinary collaboration.
- We describe a new application from archaeobiology that strongly benefits in multiple ways from an interdisciplinary data science approach.
- Based on problems in the application domain, we identify new challenges for the data mining community that will enable researchers from archaeobiology to improve experiment design and understand the resulting data.
- We discuss the resulting conclusions and added values for the domain experts.

The rest of the paper is organized as follows: In Section II we describe our data-driven approach for the analysis and the evaluation of individual features. In Section III we introduce

the task of isotopic mapping, the underlying data, and important challenges. We also present our dataset, which comes from a specific region in the Alps in Europe, experimental results, evaluated by domain experts, and insights gained through this study. In Section IV we conclude our study and highlight challenges and opportunities in this domain for the data mining community.

II. FEATURE EVALUATION

In our approach, the quality of a feature, or feature subset, is assessed based on its contribution to one or more reference data structures. In particular, we assess how stable (i.e., unchanged) the data structure is across feature space projections. Our assumption is that a highly relevant projection will result in a data structure that resembles the reference data structure.

The task of assessing the importance of a feature for provenance analysis is reminiscent of feature selection and feature ranking. Feature selection generates a subset of the most suitable features for a given task, whereas feature ranking returns an ordering of features according to their importance for the task [2]. Most of the common approaches are supervised, meaning that they require class labels for accessing the quality of a feature or feature subspace [3]. Such information is not available for the discussed usecase, therefore we have

to rely on unsupervised feature selection approaches [4]. In particular, we follow a wrapper-based approach [5] where we use a learning algorithm (EM clustering in our case) for the evaluation of a feature or a subspace. A big part of research on feature selection and feature ranking methods is focused on reducing the exponential search space of possible solutions. In our case, the feature space is low-dimensional but the domain scientists are interested especially on understanding i) the importance of each feature for the final model and ii) whether there are other features in the feature space that can replicate the “contribution” of that feature. The reason is that feature acquisition is an expensive process as domain experts have to follow lengthy and time consuming processes of cleaning the findings and measuring the isotope values. Moreover, in some cases it is not possible to measure all different isotopes for all available samples. This is the case for our project, where the oxygen isotope cannot be measured for cremated human findings. So it is extremely important for the domain experts to understand whether oxygen is a key feature for the analysis and also whether the remaining features can compensate for oxygen’s contribution to the final model. Therefore, we follow a clustering-based feature evaluation approach, where we compare unsupervised learning results that convey aspects of the data structure (from a single feature point of view) with the data structure (as captured by the reference clustering). That is, the data structure is represented by the clusters extracted from the data. To assess the effect of a projection on the data structure, we compare the projection-based partitioning to the reference data partitioning.

Our proposed unsupervised feature evaluation framework consists of three steps:

- 1) data structure extraction (clustering)
- 2) data structure comparison (Adjusted Rand Index)
- 3) feature evaluation

Before explaining each of these steps, we introduce some notation: Let D be a dataset in a feature space F . Let $F_0 \subset F$ be the feature set from which a model of the reference data structure is extracted by clustering; we refer to Θ^{F_0} as the *reference clustering* and to F_0 as the *reference feature space*. Let $F_v \subset F$ be a set of features to investigate w.r.t. their quality for the reference data structure, Θ^{F_0} . Note that F_v and F_0 are treated as being independent from each other even though they need not be disjoint.

A. Unsupervised data structure extraction

To extract structure from the data, we use a clustering approach [6]. Domain knowledge suggests continuous values for the measurements, which can be best modeled as a mixture model of continuous distributions, like a Gaussian Mixture Model. To extract a robust indication of the data’s structure in an unsupervised way, we applied the Expectation-Maximization (EM) algorithm [7]. EM fits a number of multivariate normal distributions over the given data set. The result is a soft-clustering; in our dataset though the assignment is typically fairly hard. A typical run over the isotope dataset (see Section III-C) results in a standard deviation of 0.115

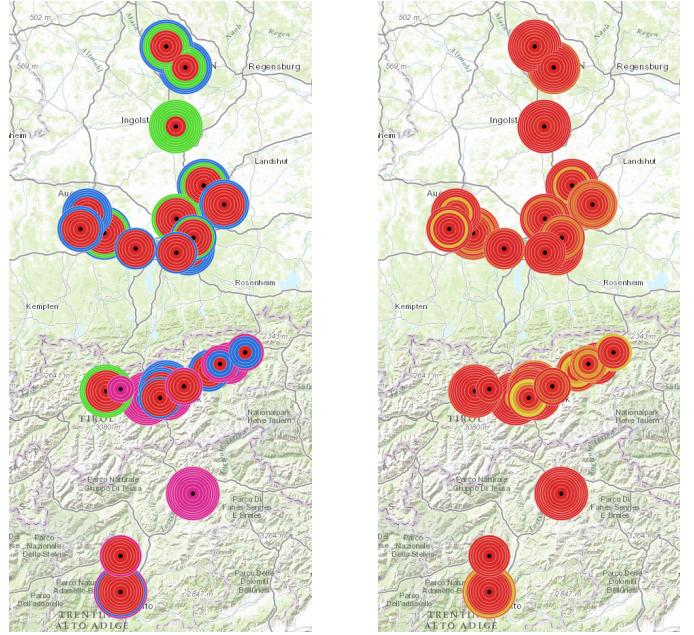


Fig. 2: Example EM clustering on isotope data. Image best viewed in color. Left: spatial projection of hard clustering (maximum likelihood cluster). Right: membership likelihood of soft clustering (red highest, yellow lowest).

for the maximum likelihood cluster labels. As an example, Figure 2 depicts the membership and spatial structure of a sample clustering done on the set of 217 bones where only the seven isotopic features are used. For easier handling, we convert the cluster probabilities to hard cluster assignment by their maximum likelihood. The result of the clustering is a set of partitions, $\Theta^F = \{\theta_1, \theta_2, \dots, \theta_k\}$, where k is the number of clusters (optimized by cross-validation, see below).

B. Comparing clusterings

To compare how well a clustering Θ^{F_v} extracted from an investigated feature projection F_v reflects the structure of a reference clustering Θ^{F_0} , we employ the *Adjusted Rand Index* (ARI) [8] of the two clustering partitionings:

$$s(F_0, F_v) := ARI(\Theta^{F_0}, \Theta^{F_v})$$

ARI evaluates the agreement between two clusterings by counting pairs assigned to the same cluster under both clusterings and pairs assigned to different clusters versus the total number of pairs in the dataset. ARI was proposed to reduce the influence of randomness on the traditional Rand Index (RI) [9] and has been proven to perform better when the number of clusters in the two clusterings is not the same [10], [11]. Like the rand index, ARI has a maximum value of 1 and takes the value 0 when the index equals its expected value. However, negative values are also possible and indicate an agreement that is even less than one expected between two random clusterings.

C. Unsupervised feature evaluation

Not all attributes are equally important for a given analysis task: A feature may be unnecessary to describe the result of a given analysis or the data reflected in the feature may be noise or encompassed by other attributes. By selecting a suitable comparison feature space we investigate the *structural relevance* of a feature (i.e., how well it captures the structure in isolation) for a clustering as well as its *structural redundancy* (i.e., if the clustering becomes unstable without this particular feature).

To generate these scores, we extract a single feature $f \in F_v$. Let D_f be our original dataset projected onto dimension f and let Θ^f be the clustering over $D_f : \Theta^f = \{\theta_1, \theta_2, \dots, \theta_{k'}\}$, where k' is the number of clusters. We refer to Θ^f as the *univariate clustering*. Let $f_- = F_v \setminus f$ be the complementary feature space, that is, all dimensions in F_v except for the investigated feature f . Let D^{f_-} be the complementary dataset, i.e., the dataset projected onto the complementary feature space f_- . Applying EM on D^{f_-} generates a clustering $\Theta^{f_-} = \{\theta_1, \theta_2, \dots, \theta_{k''}\}$ where k'' is the number of clusters. We refer to Θ^{f_-} as the *complementary clustering*.

To calculate the structural relevance of f , we compare the univariate clustering Θ^f derived from the specific feature f to the reference clustering Θ^{F_0} :

$$s_{relevance}(f, F_0) := ARI(\Theta^f, \Theta^{F_0})$$

To calculate the structural redundancy of f , we compare the complementary clustering Θ^{f_-} derived from the complementary feature space f_- to the reference clustering Θ^{F_0} :

$$s_{redundancy}(f, F_0) := ARI(\Theta^{f_-}, \Theta^{F_0})$$

The first comparison evaluates the structural relevance of f for Θ^{F_0} , whereas the second evaluates whether f 's contribution can be reproduced by other features in the feature space. In that sense, the first score derives the specific feature's structural relevance and the second score its structural redundancy due to the existence of other feature(s) in the feature space.

Combining structural relevance and structural redundancy scores in a single score is not straightforward, due to their complimentary semantics. We characterize each feature f in terms of both structural relevance and structural redundancy. To help a domain expert glance the effect a feature may have on their analysis, we combine the two scores in one plot where the x-axis reflects the structural relevance score and the y-axis the structural redundancy. In other words, the x-axis represents the degree to which the reference clustering structure is evident in a single dimension f , while the y-axis shows whether the reference clustering structure can be captured by the rest of the dimensions. These plots can be seen in Figures 4, 5, and 6. They will be explained in detail later in this paper. We present a study using this technique in the following section.

III. DATA ANALYSIS IN BIOARCHAEOLOGICAL SCIENCES

The presented technique was conceived for an international and interdisciplinary project involving classical archaeology,

bioarchaeology, biology, geology, and computer science. The final goal of the project is to generate an isotopic map of the reference region based on isotopic measurements in bone samples of three vertebrate taxa from excavation sites along the Inn-Eisack-Adige passage. A geographic map of the reference region including the sample sites is shown in Figure 1. The envisioned isotopic map will represent the common, local isotopic signatures (or fingerprints) characteristic for a given spatial region. Archaeologists can apply this map to differentiate between local and non-local finds, and to define the place of origin (*provenance analysis* [12], [13]) of the latter in order to answer the aforementioned scientific questions regarding mobility, trade, and cultural transfer. The reason behind this application is that knowledge of the spatial distribution of stable isotopes in the environment allows identifying outliers that represent primarily non-local individuals.

Based on the experience of the domain experts, seven isotopic systems from three elements (oxygen, strontium, and lead) were identified as being potentially relevant for the differentiation between local and non-local finds, and the definition of the place of origin of the latter, i.e., the construction of the envisioned map. From the sites displayed in Figure 1 samples were selected and for each seven isotope ratios were measured. The goal of this study is to identify which of the isotopic systems (oxygen, strontium, lead) to use for provenance analysis in this reference region. The processing of the material and the measurement of isotopic signatures is costly, time consuming, and wastes precious archaeological material. Thus, the design of the underlying data collection (which samples and isotopic systems are measured, etc.) are crucial for the inference of a sound and reliable map.

It should be stressed that the results presented in this study only hold for the specific reference region, i.e., the Inn-Eisack-Adige passage. However, our framework is quite generic and is applicable to other reference regions and/or other isotopic systems and even entirely different data sets from a multitude of disciplines and use cases.

A. A Brief Introduction to Isotopic Mapping

Bioarchaeologists are frequently faced with the task of distilling a relatively simple model from measurements that are tainted by the complexity of the biological processes involved. This is especially characteristic of isotopic mapping. Nevertheless, stable isotopes are indispensable markers for the monitoring of the flow of matter through biogeochemical systems. Isotopes are atoms of the same element that have the same number of protons and electrons, but differ in the number of neutrons. Isotopes are generated, e.g., by the decay of parent isotopes, or by reactions with subatomic particles in the environment. For example, the three stable isotopes of oxygen are ^{16}O , ^{17}O , and ^{18}O . All of these have 8 protons and 8 electrons, but range from 8 (^{16}O) to 10 neutrons (^{18}O). An isotope is called “stable” if it does not decay into another isotope. Oxygen atoms with fewer (e.g. ^{15}O) or more (e.g. ^{19}O) neutrons are unstable and will eventually decay into other stable isotopes.

Differences in the number of neutrons results in different atomic masses and lead to differences in molecular bond strength and vibration energies. This, and the different thermodynamic reactivity of light and heavy isotopes leads to isotopic fractionation (i.e., uneven partitioning of isotopes between source and product). Isotopic fractionation and mixing in an ecosystem generate compartments with characteristic isotopic signatures. For example, evaporation and condensation in the course of hydrological processes lead to predictable distributions of oxygen isotopes in the atmosphere and in precipitation. Isotopic labels, which are shared by certain ecological components such as soil, water, plants, microbia, and animals, have been successfully used for the generation of *isotopic maps* or *isoscape*s for the investigation of landscape ecology. Such isotopic maps representing the common, local isotopic signatures (or fingerprints), can later be applied to distinguish local and non-local finds: a local outlier, i.e., a sample found at location l that has an isotopic fingerprint different to the local fingerprint of l according to the map, is interpreted as non-local. If the isotopic fingerprint of the non-local sample matches the isotopic fingerprint of another location o , it is likely that o is its place of origin. Both the knowledge of outliers and their potential places of origin is very valuable for answering research questions in bioarchaeology. For example, isotopic fingerprints of ivory samples are used to predict the place of origin of this ivory sample (potentially classifying this sample as illegally harvested) [14].

Isotopic maps are empirically generated by sampling the relevant environmental components and by measuring their isotopic signatures. In bioarchaeology, such samples are human and animal remains found in archaeological sites. However, due to intricate biological and chemical processes, these samples do not directly reflect the geological characteristics. Examples of such processes include metabolic differences between species and individuals (some of the inter-species differences can be reduced by applying empirically determined formulas), aging, integration over various environmental conditions, weathering of bones, metabolization, etc. Thus the geological characteristics of a region are only one of a few factors contributing to the measured isotope ratios. Since we cannot know the details of the metabolism that crucially influence the isotopic composition of an organism, the only realistic way of modeling the distribution of isotopes in animals found in a region is by building a model based on the measurements associated with them. One way to allow all other influences to average out is to aggregate over samples from spatially close sites. However, the resulting values may not be applicable as a model to a single sample, which is still subject to individual variability as outlined above. In addition, on the one hand, non-local finds at a specific site will most likely impact the aggregated local model in an undesired way. But, on the other hand, the identification of non-local finds requires a reliable local model. As a consequence, the local isotopic fingerprints provided by isotopic maps will never be as reliable as the term fingerprint may suggest, but are always subject to probabilistic interpretation.

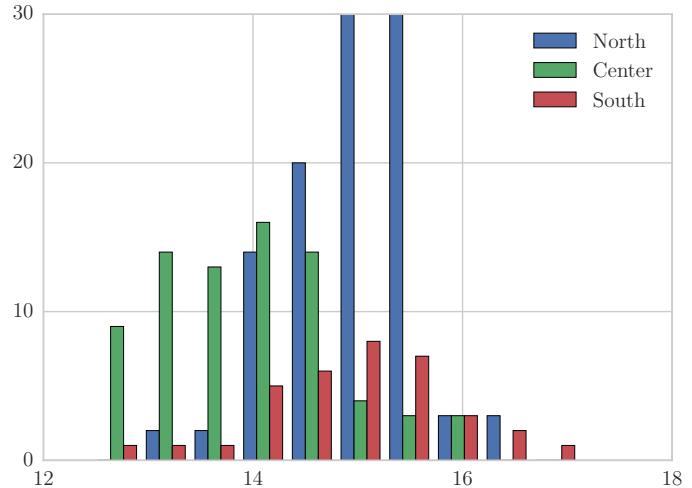


Fig. 3: Distribution of $\delta^{18}\text{O}$ by region. Although very large regions were picked, there is only a very weak correlation discernable.

Historically, stable isotopes in bioarchaeological finds were measured and simply compared to the known spatial distribution of the isotopic system under study such as $^{87}\text{Sr}/^{86}\text{Sr}$ in geological maps, or the climate and habitat dependent distribution of C₃- and C₄-plants which is reflected in the ^{13}C -values of the consumers' tissues. Outliers, detectable by conservative statistics (e.g. [15]), were readily interpreted as immigrant individuals. Very often, this was simply done by measuring one specific isotopic system, e.g. $\delta^{18}\text{O}$ from phosphate in bones, and manually determining local models and outliers in the univariate plots of the resulting values. However, growing insights into small-scale variabilities in isotopically characterized ecogeographical compartments gave rise to more fruitful discussions on mobility versus migration and trade in the past. Pretty soon it became obvious that measurement of stable isotopes for the reconstruction of migration and trade in bioarchaeology can not be looked at in isolation but rather necessitates collecting a lot of accompanying data (e.g., analysis of not only human, but also animal bones, or soil sampled from the same site) for the assessment of ecogeographical baseline values to account for the small-scale variability in time and space.

One important effect of the complicated nature of this area is that there is no ground truth available. Handling the problem in an unsupervised way means that the investigation cannot rely on an authorized reference, but has to make and use its own references heavily based on assumptions and experiences of the domain experts. This makes this application ideal for a data-driven approach that is underpinned by domain-expertise.

B. The Application: Mobility in the European Alps

The prehistoric transalpine passage following the Inn, Etsch, and Adige rivers from modern-day Germany, through Austria, into Italy is of great archaeological interest for having been an important trade route over the Alps mountains. An

international research group has investigated 30 archaeological sites along this route and at its southern and northern ends (see Figure 1) to generate a dataset based on which an isotopic map can be derived for this reference region. The application of this map will leverage the investigation of questions about transfer of humans, goods, and culture through the passage. Of particular interest were animal remains that were uncovered at the examined sites with a focus on archaeological bones of three vertebrate taxa: pig, cattle, and red deer. These three animals have different characteristics in terms of mobility.

In general, it is very time consuming and costly to generate the data an isotopic map is based on. The measurement of an isotopic value requires a complex procedure including, among others, the extraction of a suitable part of the material, cleaning, etc. Typically, the material used for one measurement is destroyed in the analysis and cannot be used for further runs. Thus, one crucial first step towards the establishment of the final map is to decide which isotopic measures to generate. For that purpose, a case study was done with a small set of 217 samples derived from 30 investigated sites. From each investigated specimen, seven isotopes were measured: ^{18}O , ^{86}Sr , ^{87}Sr , ^{204}Pb , ^{206}Pb , ^{207}Pb , and ^{208}Pb . Due to technical particularities of isotope measuring, the strontium (Sr) and lead (Pb) isotopes were measured and recorded as fractions of isotopes of the same element, yielding the fractions $^{87}Sr/^{86}Sr$, $^{208}Pb/^{204}Pb$, $^{207}Pb/^{204}Pb$, $^{206}Pb/^{204}Pb$, $^{208}Pb/^{207}Pb$, and $^{206}Pb/^{207}Pb$. The oxygen isotope was normalized against ocean water isotope levels and recorded as $\delta^{18}O$. This yields a 7-dimensional feature vector for each recovered sample. In addition to these isotope measurements, each sample was annotated with a spatial description (latitude, longitude, altitude) based on the discovery area. Also, each sample was recognized as one of the three animal species (pig, cattle, and red deer).

Due to its popularity in provenance analysis, we first focused on oxygen as a marker to distinguish between local and non-local finds. A preliminary manual analysis of the $\delta^{18}O$ of 118 of the 217 animal bone samples by a domain expert revealed a highly significant correlation ($r = -0.68$) between $\delta^{18}O$ and altitude, whereby the averaged $\delta^{18}O$ values plotted exactly on the regression between altitude and $\delta^{18}O$ in precipitation in the Alps as published by Kern et al. [16]. However, although the $\delta^{18}O$ values behaved as expected and, thus, are potentially suitable for provenance analysis as a single marker, in this study it proved impossible to distinguish even rough spatial compartments (such as north, center, and south of the Alps). The histogram of $\delta^{18}O$ values shown in Figure 3 illustrates this variability. The samples from the three very coarse compartments north, center, and south derived from a hypothesis of the domain experts are marked in different colors. The same observation was made when considering species-specific fractionation factors. Interindividual variability remained high and did not permit for a firm assignment of individual animals to spatial regions.

These first results already gave the domain experts some very interesting hints and further questions arose. In particular,

TABLE I: Notations for the different subsets of features used to derive reference clusterings.

Reference clustering	Description (feature set)
I	all 7 isotopic features ($^{87}Sr/^{86}Sr$, $^{208}Pb/^{204}Pb$, $^{207}Pb/^{204}Pb$, $^{206}Pb/^{204}Pb$, $^{208}Pb/^{207}Pb$, $^{206}Pb/^{207}Pb$, $\delta^{18}O$)
I^{-O}	all isotopic features except oxygen ($^{87}Sr/^{86}Sr$, $^{208}Pb/^{204}Pb$, $^{207}Pb/^{204}Pb$, $^{206}Pb/^{204}Pb$, $^{208}Pb/^{207}Pb$, $^{206}Pb/^{207}Pb$)
S	all 3 spatial attributes ($altitude$, $latitude$, $longitude$)
S^{-lon}	all spatial features except longitude ($altitude$, $latitude$)

the domain experts were even more interested in deeper insights into the relevance and redundancy of the measured isotopes. Especially, whether the hypothesis can be confirmed that oxygen isotopes can be omitted or easily replaced by a (set of) other isotopes. Then it would be possible to use datasets of samples where oxygen isotopes could not be measured for the same analyses. These samples are common, because oxygen isotopes are not stable at high temperatures, i.e., in cremated material. Cremated material is very common for human remains in this reference region. The measuring of the other isotopes is slightly easier. If the subset without oxygen is enough for origin prediction, a more detailed model might be derived by augmenting the dataset with human cremation data. Thus, the data science task was to score the measured isotopic features in the dataset of the case study in terms of relevance and redundancy with respect to provenance analysis.

C. The Results

The definition of the reference clustering is crucial for our ARI-based feature evaluation presented in the previous section. However, there is no ground truth reference clustering available for the region under inspection. A purely data-driven approach is also not possible since we cannot be sure about the originality of each finding, i.e., we do not know if a bone found a specific site s in fact originates from s or is a non-local outlier. As a consequence, even if we explore local isotopic outliers within each site, we are not sure if the outliers are non-local finds or local ones. However, the domain experts have some assumptions and hypotheses available about possible spatial compartments that could be used to derive potentially plausible approximations of the ground truth. Thus, we follow a mixture between a data driven approach enriched by domain expertise.

1) *Reference clusterings:* Instead of using just one potential reference clustering, we investigated several possible definitions for the reference clustering based on the available features, in close collaboration with the domain experts. In other words, we generated reference clusterings using a data driven approach based on clustering, but rely on the domain experts for deciding which features were used for the clustering. As a result of this process we decided on multiple feature spaces

to generate reference clusterings: from containing all isotope and spatial features to containing only single domain features, i.e., isotopes or spatial coordinates. In the following, the set $I := \{^{87}\text{Sr}/^{86}\text{Sr}, ^{208}\text{Pb}/^{204}\text{Pb}, ^{207}\text{Pb}/^{204}\text{Pb}, ^{206}\text{Pb}/^{204}\text{Pb}, ^{208}\text{Pb}/^{207}\text{Pb}, ^{206}\text{Pb}/^{207}\text{Pb}, \delta^{18}\text{O}\}$ denotes all isotopic features, $I^{-O} := \{^{87}\text{Sr}/^{86}\text{Sr}, ^{208}\text{Pb}/^{204}\text{Pb}, ^{207}\text{Pb}/^{204}\text{Pb}, ^{206}\text{Pb}/^{204}\text{Pb}, ^{208}\text{Pb}/^{207}\text{Pb}, ^{206}\text{Pb}/^{207}\text{Pb}\}$ denotes all isotopic features except oxygen. In addition, we use $S := \{\text{altitude}, \text{latitude}, \text{longitude}\}$ for all spatial attributes and $S^{-lon} := \{\text{altitude}, \text{latitude}\}$ refers to the spatial attributes without longitude (see Table I for an overview). We list the different set-ups for the feature spaces that we used to generate the reference clusterings based on the input of the domain experts below. For this first set of experiments, the set of features under investigation is always the set of isotopic features, i.e., $F_v := I$.

$F_0 = I \cup S$ (**Isotopes + Spatial**) The feature space consists of all available isotopic features and spatial features. This is the most information we have and, thus, serves as the starting point of the study.

$F_0 = I \cup S^{-lon}$ (**Isotopes + (latitude, altitude)**) From the spatial attributes only those that have been found to have an effect on the isotopes are retained, namely altitude and latitude. Since the passage under inspection is mostly north/south, the domain experts expect that longitude has only minor influence on the spatial compartments.

$F_0 = I$ (**Isotopes only**) The feature space consists only of the isotopic features. There is no spatial influence. Such a feature space is typically used for fingerprinting and predicting the origin of new samples (with unknown spatial coordinates).

In a second analogously conducted series of experiments, oxygen was removed from the reference clusterings since the domain experts wanted to test the hypothesis that oxygen is much less relevant than other isotopes in this reference region and for this sample selection. This has been observed in other provenance studies. Especially the sample selection using a mix of three different species may have a blurring impact on the $\delta^{18}\text{O}$ -values according to the domain experts. Analogously, the set of features under investigation is always the set of isotopic features without oxygen, i.e., $F_v = I^{-O}$. The resulting configurations are similar to the four alternatives listed above:

$F_0 = I^{-O} \cup S$ (**Isotopes (except oxygen) + Spatial**) The feature space consists of all isotopes minus oxygen and all spatial features.

$F_0 = I^{-O}$ (**Isotopes only (except oxygen)**) Only the isotope description, without the oxygen feature.

$F_0 = I^{-O} \cup S^{-lon}$ (**Isotopes (except oxygen) + (latitude + altitude)**) Isotope description, without the oxygen feature and spatial coordinates except longitude.

A supplementary feature space to be used as a reference clustering is purely spatial:

$F_0 = S$ (**Spatial only**) The feature space consists only of spatial coordinates. Isotopic values do not play any role

and findings from spatially close sites are considered to be the same cluster (compartment). This ground truth scenario must be complemented by a corresponding set of investigated features, i.e., $F_v = I$, and $F_v = I^{-O}$.

2) *Experiments*: For each of the feature spaces described above, we apply EM to derive the reference clustering and we evaluate how each isotope attribute "contributes" to the corresponding reference clustering. We illustrate these results in the structural relevance-vs-structural redundancy plots presented in the previous section. For the EM, the number of clusters was selected by cross-validation as in the Weka data mining framework [17]. When examining the presented reference attribute sets, we chose $F_v = I$ or $F_v = I^{-O}$ to reflect the isotopes in the reference attributes. That is, where F_0 contains I , F_v becomes I , where F_0 contains only I^{-O} , F_v becomes I^{-O} . A special case is $F_0 = S$, which does not contain any isotopes to compare with. In these scenarios, we investigated both $F_v = I$ and $F_v = I^{-O}$ for completeness.

The results of reference clusterings containing isotopes including oxygen are presented in Figure 4, experiments with isotopes excluding oxygen are presented in Figure 5, and those with only spatial attributes are presented in Figure 6. Regarding the ARI values, a score of zero indicates random behavior while a score of one indicates identical clusterings.

In the following we discuss the individual experiments showing structural redundancy and structural relevance for all described reference feature sets and discussing potential explanations for the observed values.

a) $F_0 = I \cup S$, $F_v = I$: See Figure 4a. *Strontium* is the most prominent attribute as it has the highest structural relevance score and the lowest structural redundancy score. *Lead* isotopes depict a similar behavior, scoring average relevance and redundancy scores. An exception is $^{208}\text{Pb}/^{204}\text{Pb}$, which has a very low relevance; a closer inspection of the results shows that a clustering based on $^{208}\text{Pb}/^{204}\text{Pb}$ only places all instances in the same cluster, i.e., the values in this feature follow one Gaussian distribution. *Oxygen* has also a very low relevance score.

b) $F_0 = I \cup S^{-lon}$, $F_v = I$: See Figure 4b. F_0 now includes no longitude information. There is little difference in the rankings compared to the *IS* case, although the scores are higher. An interesting change is the repositioning of oxygen: its redundancy became lower and relevance became higher.

c) $F_0 = I$, $F_v = I$: See Figure 4c. The removal of all spatial information from F_0 pits the isotopes against each other. This might be due to the better quality clusterings we obtain by also employing spatial information. *Strontium* is still the top relevant isotope, however two lead isotopes score very close, namely $^{206}\text{Pb}/^{207}\text{Pb}$ and $^{206}\text{Pb}/^{204}\text{Pb}$. Both isotopes re-positioned in the plot after the removal of the spatial information from the reference clustering. In particular they became more relevant and less redundant. Also, the redundancy of oxygen increased.

d) $F_0 = S$, $F_v = I$: See Figure 6a. This scenario tests how well the isotope's structure lines up with the spatial structure. We expect very little alignment as the spatial structure

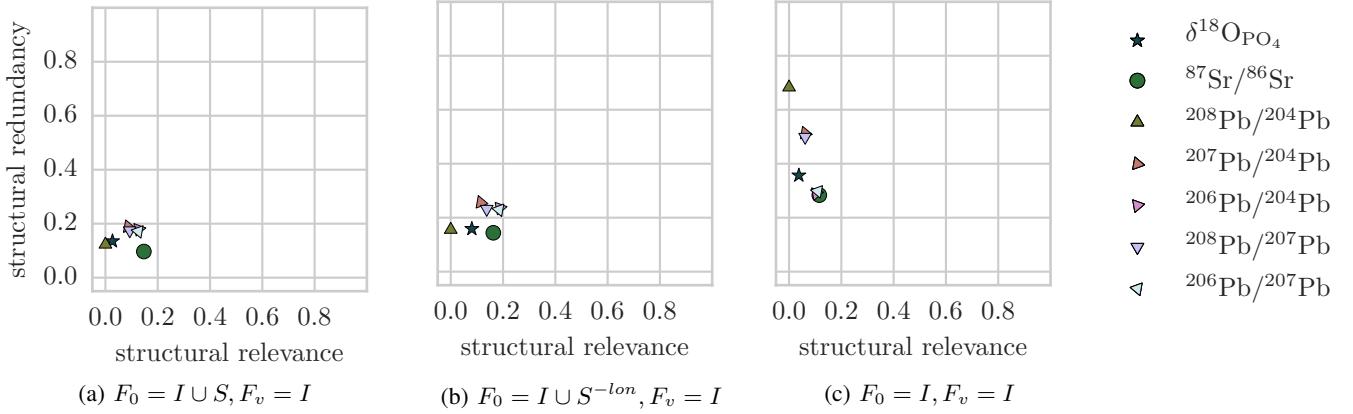


Fig. 4: Structural relevance-vs-structural redundancy plots using reference clusterings with all isotope features.

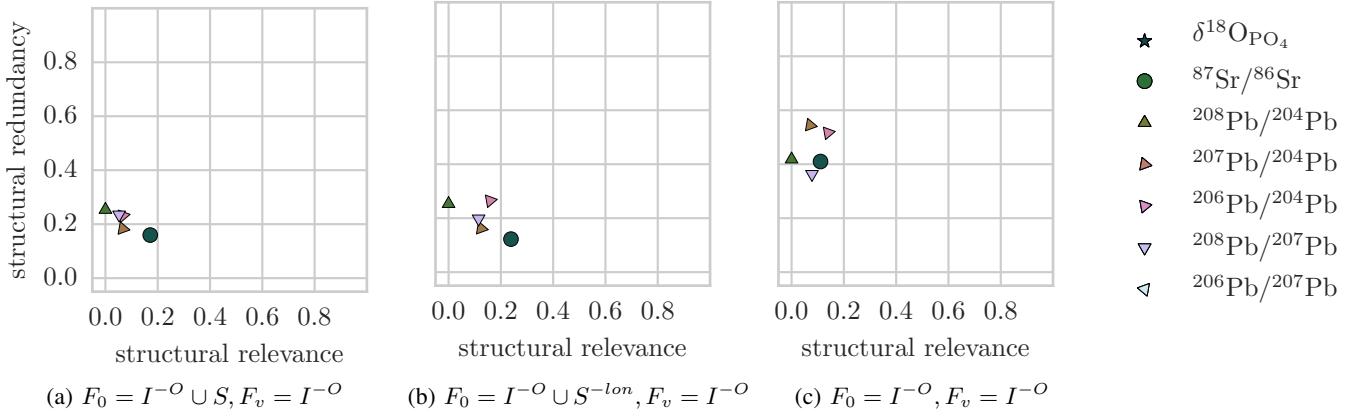


Fig. 5: Structural relevance-vs-structural redundancy plots using reference clusterings with all isotopes except oxygen.

will be dominated by the density of sample sites, which the isotope values reflect indirectly at best. The lead isotopes have very low redundancy and relevance scores, indicating that they neither reflect the spatial structure, nor does their complimentary feature space do so. Strontium seems to reflect all the structure: the sets of isotopes that contains strontium achieve a median score of 0.08 and strontium by itself achieves a score of 0.13. All other isotopes have relevance scores around zero (oxygen scoring highest at 0.04).

e) $F_0 = I^{-O} \cup S, F_v = I^{-O}$: See Figure 5a. Without oxygen, strontium is again the most prominent attribute, whereas the relevance of lead decreases.

f) $F_0 = I^{-O} \cup S^{-lon}, F_v = I^{-O}$: See Figure 5b. Compared with the previous scenario containing the entire set of spatial attributes, strontium retains very similar scores, but some lead isotopes' relevance increases. This indicates a stronger role of those lead isotopes in the formation of the structure, possibly because longitude supports other structural elements that are now being expressed less strongly.

g) $F_0 = I^{-O}, F_v = I^{-O}$: See Figure 5c. Removal of all spatial information (and oxygen), affects the ranking of strontium. Lead isotopes are now more relevant comparing to strontium, but still more redundant than strontium.

h) $F_0 = S, F_v = I^{-O}$: See Figure 6b. If oxygen is omitted, the situation changes only marginally compared to the original setup with $F_v = I$. This indicates that oxygen had little influence on the structure of the isotope space, consistent with the analysis above.

3) *Discussion:* We already pointed out that each experiment refers to a specific reference clustering and therefore it is not straightforward to compare scores between them. However we can draw some conclusions about the dataset from the interpretation of all experiments.

First of all, it is clear that the choice of reference clustering influences the ranking of different isotopes with respect to their structural relevance and structural redundancy. There are however some features which are repeated across different configurations. In particular, there is a much better separation in the rankings when spatial coordinates are considered, cf. Figure 4a and 4b. A possible explanation is that the reference clustering is much more differentiated when considering the complete feature space. A similar observation holds when we remove oxygen from the feature space, cf. Figure 5. The worst distinction is manifested when we consider all isotopes, including oxygen, cf. Figure 4c.

Regarding the behavior of the different isotopes, *strontium* and *lead* are the top *structurally relevant* isotopes, i.e., they

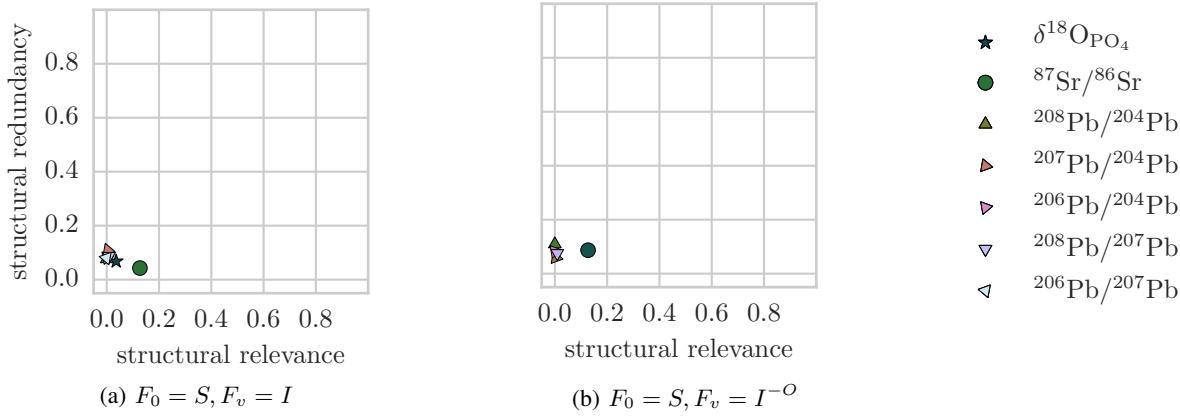


Fig. 6: Structural relevance-vs-structural redundancy plot using reference clustering on spatial data. The investigated clustering are based on all isotopes (a) and all isotopes except oxygen (b).

display higher values in the structural relevance axis. This implies that in isolation these isotopes manage to capture most of the reference clustering structure. *Oxygen* depicts a low structural relevance score, meaning that alone it is not a good indicator of the reference clustering.

With respect to *structural redundancy*, strontium has the lowest redundancy, implying that the information in strontium is not replicated by some other isotope or combination of isotopes in the dataset. The lead isotopes display high redundancy as expected since we have five different lead isotopes in our dataset. Related behavior within the lead group is expected, because the lead isotopes used in this study are measured relatively against baseline isotopes ^{204}Pb and ^{207}Pb . This implies that each isotope is measured multiple times as part of different fractions. Although they are measured independently, these fractions can be mathematically converted into one another by multiplying two of each data point's measurements and the derived values are generally very close. Accordingly, lead isotope ratios generally score higher on the redundancy score than other isotopes. This multiplicative relationship is not clearly reflected in the data structure and the redundancy scores of lead isotopes are therefore not exceptionally high. It is interesting to note that the lead isotope ratios generally change uniformly and that correlations between the scores of fractions that share an isotope can only be observed in a few cases. Two that do behave similarly are $^{206}\text{Pb}/^{207}\text{Pb}$ and $^{206}\text{Pb}/^{204}\text{Pb}$. There seems to be no obvious explanation for why these systems in particular show this behavior, but it may indicate that stable lead isotopes can be used for provenance analysis. This warrants further research.

Overall low relevance scores indicate that no isotope alone reflects the full structure of the data. This supports the emerging trend to use multivariate analyses in the domain sciences.

The bad scores achieved by all isotopes against the reference clustering including only spatial coordinates illustrates that there is no trivial correspondence between the two domains, isotope and spatial. Domain knowledge suggests a connection, but it is not pronounced enough to be automatically reflected

by the isotope feature set. Therefore the combination of both domains to extract a spatially coherent isotope map is also not trivial and will require more complex models.

D. Insights

Our study resulted in two major insights that were previously uncertain and represented major added values for the domain experts.

Insight 1: *A multivariate isotopic fingerprint is needed instead of a univariate analysis relying on oxygen only.*

Our analysis showed that despite its popularity, oxygen does not provide exceptional structure to the dataset (average structural relevance), nor is it unique in the role it plays (no exceptionally low structural redundancy values). Thus, at least in this reference region, provenance studies based solely on oxygen is bound to fail. On the other hand, the implication from our results is that the envisioned isotopic map can benefit strongly from a multi-isotopic fingerprint that includes strontium and lead isotopes as well.

Insight 2: *Omission of oxygen in the isotopic fingerprint does not considerably decrease the quality of the fingerprinting.*

Oxygen did not show a particularly low redundancy. Its redundancy scores were always comparable with other isotopes, reaching values of up to 35%. This indicates that oxygen does not play an exceptional role in the data's structure and that other isotopes can provide much the same information as oxygen. Its low relevance score indicates that oxygen does not dominate the structure (i.e., other isotopes are needed).

Implications and Added Values: The fact that oxygen seems not very relevant to provenance analysis in the reference region, opens up several opportunities.

- Although the inclusion of oxygen does not seem to diminish the clustering results, its omission also has little negative impact. This makes oxygen a potential candidate to save costs and time when generating the data source on which an isotopic map is based.
- So far, the isotopic map was designed to rely on animal bones only. Including human remains would be generally

beneficial but ancient human remains in this reference region typically are cremated, and, thus oxygen values cannot be derived. The low relevance of oxygen opens up the possibility to explore this cremated material on a larger scale.

IV. CONCLUSION

This paper presented a technique for domain scientists to assess the relevance of features for analysis. The technique's purpose is to inform decisions about features, such as whether to record a variable in the first place, as well as guide further investigations into the role of a feature. After analysis, domain scientists are presented with two scores for each isotope: the structural relevance, which indicates to what degree the data's structure is represented in a given feature, and the structural redundancy, which indicates how much of the data structure is lost without the feature.

By splitting the result into two independent scores (structural relevance and structural redundancy) we allow domain scientists to grasp two important orthogonal properties of the data that could otherwise not be discerned from univariate and bivariate visualizations. A variable that is structurally relevant, but redundant, may still be less important than one that is structurally less relevant, but cannot be replaced by a combination of different isotopes, or the other way around. In low-dimensional datasets individual variables are expected to be generally more relevant than in higher-dimensional ones. However, no single variable is indispensable if multi-variate analysis is employed. Indeed if the analysis could be based on only a single variable, multi-variate analysis would not be necessary for the application at hand.

In an application context these measurements inform further investigations of the role of features in domain models. In the presented case study, domain scientists were presented with scatter plots of the structural relevance and structural redundancy scores of each isotope system in an archaeological dataset. The presented study was only an early step towards the overall goal of the interdisciplinary research project of mobility and cultural transfer in the past in a specific reference region. This analysis was important since it revealed many new insights for the domain experts, mainly in terms of how the data should be generated (e.g., which findings to include into the analysis, which isotopic values should be measured or can be omitted, etc.) so that a reliable map can be derived.

Analysing the presented data to generate the aspired isotopic map presents further data science challenges. Suitable methods will be needed to identify the small scale compartments with characteristic isotopic fingerprints. Based on these compartments and their characteristic fingerprints, a predictive model needs to be derived in order to classify local findings and non-local findings as well as the places of origin for the latter. To be useful to domain scientists, a visualization of the isotopic map, i.e., of a multi-dimensional fingerprint, is planned that allows

for some insights without the full complexity of the underlying model. Finally, the resulting isotopic map must be extendable to surrounding regions like other alpine passages in Austria, Switzerland, and France as well to other archaeological strata.

In this study, we have demonstrated that datascience techniques like the one presented in this paper can generate new insights, inputs, and impulses for domain sciences.

ACKNOWLEDGEMENT

This work is supported by the German Research Foundation within the interdisciplinary DFG Research Group FOR 1670 “Transalpine mobility and cultural transfer”. Details at: <http://www.en.for1670-transalpine.uni-muenchen.de>

REFERENCES

- [1] G. Bowen, “Isoscapes: Spatial pattern in isotopic biogeochemistry,” *Annual reviews of earth and planetary science*, pp. 161–187, 2010.
- [2] I. Guyon and A. Elisseeff, “An introduction to variable and feature selection,” *The Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.
- [3] F. Keinosuke, “Introduction to statistical pattern recognition,” *Academic Press Inc*, 1990.
- [4] M. Dash, H. Liu, and J. Yao, “Dimensionality reduction of unsupervised data,” in *Tools with Artificial Intelligence, 1997. Proceedings., Ninth IEEE International Conference on*. IEEE, 1997, pp. 532–539.
- [5] R. Kohavi and G. H. John, “Wrappers for feature subset selection,” *Artificial intelligence*, vol. 97, no. 1, pp. 273–324, 1997.
- [6] M. Mauder, E. Ntouts, P. Kröger, and G. Grupe, “Data mining for isotopic mapping of bioarchaeological finds in a central european alpine passage,” in *Proceedings of the 27th International Conference on Scientific and Statistical Database Management*. ACM, 2015, p. 34.
- [7] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.
- [8] L. Hubert and P. Arabie, “Comparing partitions,” *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.
- [9] W. M. Rand, “Objective criteria for the evaluation of clustering methods,” *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971.
- [10] G. W. Milligan and M. C. Cooper, “Methodology review: Clustering methods,” *Applied psychological measurement*, vol. 11, no. 4, pp. 329–354, 1987.
- [11] N. X. Vinh, J. Epps, and J. Bailey, “Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance,” *The Journal of Machine Learning Research*, vol. 11, pp. 2837–2854, 2010.
- [12] J. Ericson, “Strontium isotope characterization in the study of prehistoric human ecology,” *Journal of Human Evolution* 14, pp. 503–14, 1985.
- [13] T. Molleson, D. Eldridge, and N. Gale, “Identification of lead sources by stable isotope ratios in bones and lead from Poundbury Camp, Dorset,” *Oxford Journal of Archaeology* 5, pp. 249–253, 1986.
- [14] S. Ziegler and D. Jacob, “Development of a spatial reference database for ivory,” *TRAFFIC Bulletin*, vol. 23, no. 1, 2010.
- [15] G. Grupe, T. Price, P. Schröter, F. Söllner, C. Johnson, and B. Beard, “Mobility of Bell Beaker people revealed by strontium isotope ratios of tooth and bone: a study of southern Bavarian skeletal remains,” *Applied Geochemistry* 12, pp. 517–525, 1997.
- [16] Z. Kern, B. Kohán, and M. Leuenberger, “Precipitation isoscape of high reliefs: interpolation scheme designed and tested for monthly resolved precipitation oxygen isotope records of an Alpine domain,” *Atmospheric Chemistry and Physics* 14, pp. 1897–1907, 2014.
- [17] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “EM,” <http://weka.sourceforge.net/doc.dev/weka/clusterers/EM.html>, 03 2015, [Online; accessed 2015-03-10].

Data-oriented neuron classification from their parts

Evelyn Perez Cervantes

Institute of Mathematics and Statistics,
University of São Paulo, São Paulo, Brazil
Email: eperezc@ime.usp.br

Roberto Marcondes Cesar Junior

Institute of Mathematics and Statistics,
University of São Paulo, São Paulo, Brazil
Email: cesar@ime.usp.br

Cesar Henrique Comin

So Carlos Institute of Physics,
University of São Paulo, São Carlos, SP, Brazil
Email: chcomin@gmail.com

Luciano da Fontoura Costa

So Carlos Institute of Physics,
University of São Paulo,
PO Box 369, 13560-970, São Carlos, SP, Brazil
Email: ldfcosta@gmail.com

Abstract—The shape of a neuron can reveal many interesting properties about its function. Therefore, organizing neuronal cells into appropriate classes according to their respective shape is a fundamental endeavor in neuroscience. Available online datasets allow new data-oriented approaches to solve such neuroscience problems. Here we analyze the feasibility of classifying neurons according not to their respective wholes, but to its constituent parts. Such a study may reveal interesting insights, including whether parts of the neuronal dendritic arborization preserve proper information about the morphology of the whole neuron. Experimental results using open datasets are reported, thus corroborating our approach.

I. INTRODUCTION

The problem of classifying neurons has been addressed from the beginning of neuroscience, because a systematic census of neuronal cell types can provide subsidies for better understanding the brain [1], [2]. Neuronal classification, in a strict sense, is the process of dividing a set of neurons into groups or classes, which can be pre-determined (supervised classification) or inferred from the own data relationships (unsupervised classification, categorization or clustering) [3]. A standard neuron has three main structures: the cell body also called soma, the axon and the dendrites. The soma encloses the nucleus, which stores the cell's genes; the axon is a long branch that transports the electrical signals toward other neurons, and the dendrites are typically shorter branches that receive signals from other neurons. The axon communicates with the dendrites, axons and soma of other neurons through synapses and gap junctions [4].

Commonly, a neuron is characterized by its morphology, physiology and biochemistry [5], [6], [7]. It is known that the morphology of neurons provides information about their functionality [8], [3], [9], [10], [11]. A neuron can be classified by using any subset of its properties. Several attempts at neuronal classification have been formulated taking into account their axonal and dendritic structures [3], [12], [13], [14], [15], [16]. Amañazas and Ascoli in [3] present a review of automatic neuronal classification methods based on quantitative measures and machine learning.

Recent attempts have been made to categorize neurons according to their morphological properties [6], [7]. Some approaches consider the neural arbor branch density [17] as the main morphological feature. For instance, Sümbül et al. [13] proposed a method based on the relative position of the dendritic arbor to define neuronal types, and compared the identified types with molecularly defined ones. In addition to arbor density, layer-specific distribution and length have also been described as key parameters to define clusters [18]. Zhao and Plaza [19] proposed a location-based method, considering that neuron similarity is directly related to density overlaps between dendritic arbors. Lu et al. [20] considered neurons obtained from the NeuroMorpho.org [21] dataset and measurements calculated with the L-measure [22] software and applied a harmonic co-clustering algorithm based on diffusion properties. Harris et al. [23] investigated how the clustering algorithm called affinity propagation, applied to a collection of morphological and physiological variables, can identify neuronal types not revealed by morphological properties alone. Bayesian networks and clustering algorithms over axonal and dendritic arborizations were also applied to categorize interneurons [24]. Gillette et al. [25], [26] considered an encoding of axonal and dendritic arbors into sequences of characters representing bifurcations. This allowed the application of similarity measurements, commonly defined to compare gene sequences, to perform clustering analysis and define respective arbor types.

The advent and steady growth of public databases containing neuronal morphology data [27], [28], [29], such as NeuroMorpho.Org [21], have paved the way for more systematic approaches to neuronal classification [30], [31], [32], [33]. Because these databases contain data from different types of neurons, electrophysiology, laboratories, species, among other properties, it becomes possible to tackle questions, linking these various aspects with the neuronal morphology. Thus, it can give valuable insights about the diversity of neuronal shape as well as its relationship with function and evolution. However, the consideration of the neuronal morphology

typically takes into account whole cells, sometimes without axons. It would be interesting to investigate whether the morphological properties distinguishing different types of neurons extend when the neuronal morphology is approached at more microscopic levels (e.g. branches, spines, channels, etc.). This consists in the main motivation of the present article. More specifically, we approach the problem of supervised neuronal cell classification while considering not only several neurons from different classes but also compare the results when the neuronal dendritic arbors are firstly dismantled into smaller parts, corresponding to their main trees (i.e. sets of branches emanating from a single point in the soma). As a byproduct, when neurons are dismantled in such a way, the number of individuals in the database tends to grow significantly, enhancing the possibilities for big data approaches. The obtained results suggest that the morphological properties differentiating the considered neuronal groups tend to extend at the tree level, with some interesting biases.

The paper is organized as follows. In Section II the neuronal dataset used in the analysis is presented. Next, some basic concepts and methods used for characterizing the neurons are described in Section III. The neuron dismantling concept is introduced in Section IV, and the respective results obtained with the methodology are presented in Section V. The concluding remarks are in Section VI.

II. DATASET

The NeuroMorpho.Org repository was used as data source. This is a web-accessible dataset for digitally reconstructed neurons and one of the integrated resources in the Neuroscience Information Framework (NIF). It was initiated by researchers from the Krasnow Institute for Advanced Study and from the George Mason University. The first release of Neuromorpho was in 2006, with 1000 neuron reconstructions, and this dataset has been growing steadily with the contributions from several laboratories worldwide. Its current version contains 37712 neurons, belonging to 1442 different classes. In Figure 1 we show the evolution of the number of neurons contained in the dataset along the years. A noticeable increase in the number of neurons over the past three years can be noticed in the plot, evidencing the growth of interest in this kind of data [34], [35].

NeuroMorpho.Org provides information about neuronal structures as a plain text file, organized according to a format called swc. Each line of a swc file describes a respective neuronal sample point measured during reconstruction. The information gathered for each sample point include the sample index, the type (soma, axon, apical dendrite or basal dendrite), the x, y and z positions, the radius and the index of the parent sample. [22]. The repository also supplies 21 morphological measurements for each neuron, extracted using the L-Measure software [22].

A set of 2140 neuron reconstructions was considered from NeuroMorpho.Org (version 6.3). The obtained neurons are divided into 4 main classes according to the species the respective individual belongs and the biologically defined neuron

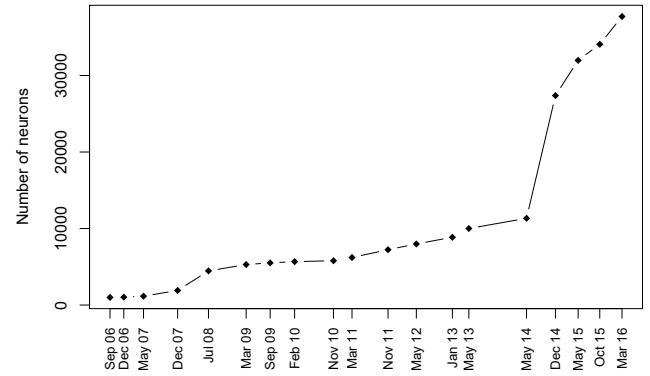


Fig. 1. Evolution along the years of the number of neurons available in the NeuroMorpho.Org repository.

type. The considered classes are: human pyramidal, mouse pyramidal, mouse retinal ganglion cells and rat interneuron. In Figure 2 we show four examples of neurons belonging to each considered class. In order to simplify the analysis, a similar number of instances per class (around 530) were used, and only neurons having more than two main stems (see next section) were used.

Pyramidal neurons owe their name to the shape of their somata (roughly triangular). They are found in several regions of the central nervous system, such as in the cerebral cortex, hippocampus and amygdala. Their ubiquitous presence across many animal species, as well as the diversity of their shape [2], [36], [37], leads to a rich set of data that is particularly suitable for pattern recognition. Regarding interneurons, which are commonly found in integrative areas of the nervous system, they provide communication between sensory or motor neurons and the central nervous system. They are usually divided into two types: (i) local interneurons, which possess short axons, and; (ii) relay interneurons, which have long axons and are even able to reach different brain regions. Interneurons are known to regulate synchronization [38] and are, therefore, sometimes associated with epilepsy [4].

Retinal ganglion cells make the bridge between the retina and distinct regions of the brain [39]. These neurons generate particular interest in the literature due to their selectivity to certain properties of images arriving at the retina, such as contrast and color [40], [41]. They also typically possess very large axons [42], [39]. Except for their axons, ganglion cells tend to be predominantly planar.

III. CONCEPTS AND METHODS

A. Morphological Features

The analysis of the neuronal shapes implies the selection of a set of objective measurements able to quantify distinct neuromorphological aspects. In the present work, we used the L-measure [22] software, which allows the calculation of a large number of morphological characteristics of neurons. A set of 20 measurements was considered, as presented in Table I. The names of some neuronal parts necessary for the understanding of these measurements are shown in Figure 3.

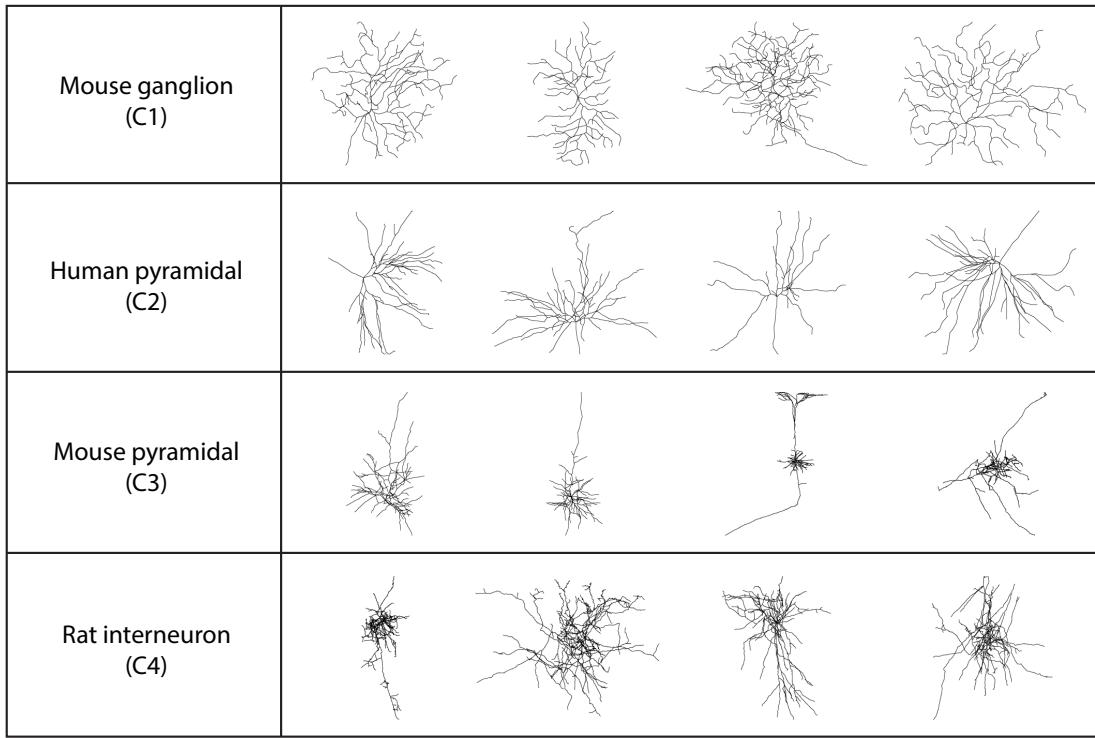


Fig. 2. Examples of neurons belonging to the four classes considered in the analysis.

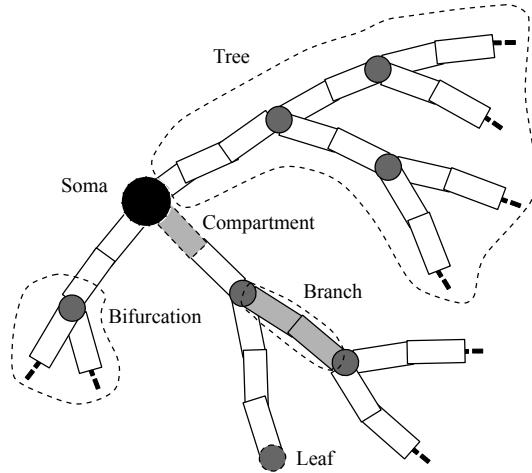


Fig. 3. Names of distinct parts used to represent a neuron.

A tree is a structure, representing dendrites or axons, attached to the soma. Each tree is composed by a group of branches, where a branch is a segment between two bifurcations or between a bifurcation and a termination point, called a leaf. The path followed by a branch is described by a sequence of compartments.

The somatic surface area is relevant for electrophysiological modeling of neurons, since it bears influence on the input resistance of the cell [43]. The number of stems indicates how many dendritic trees are attached to the neuron, and is one of the main properties considered in the current study.

The number of bifurcations underlies the degree of freedom allowed for variations of the dendritic arborization, in the sense that larger values of this feature indicate that the neuron can have a more complex shape. The features from 4 to 12, indicated in Table I, are all related to the size of the neuron. They characterize distinct aspects of the dendritic arborization, such as the extension of the neuron, which has important implications for the number of synapses [44], [45], [46], and the diameter of the dendrites, which bears direct influence on signal transmission [47].

The topological asymmetry quantifies, for each bifurcation, the difference in the number of leaves between the left and right sides of the bifurcation. In other words, if n_1 and n_2 are, respectively, the number of leaves on the left and right subtrees rooted at the bifurcation, then the topological asymmetry is given by $|n_1 - n_2|/(n_1 + n_2 - 2)$. The Rall's power indicates, for each bifurcation, the value n that best fits the equation $D_p^n = D_a^n + D_b^n$, where D_p is the diameter of the parent branch and D_a and D_b are the diameters of the daughter's branches. In case $n \approx 1.5$, the bifurcation can be described by an equivalent cylinder in the cable equation [48]. The fractal dimension quantifies the relationship between the Euclidean and path distances among branches endpoints. A unitary fractal dimension means that the neuronal branches are straight lines [49], while larger fractal dimension values are associated with neurons possessing more complex branches [49].

TABLE I
MORPHOLOGICAL MEASUREMENTS CONSIDERED FOR NEURONAL CHARACTERIZATION.

Nº	Measure description	Nº	Measure description
1	Soma surface area	10	Total arborization volume
2	Number of stems (trees) attached to the soma	11	Maximum Euclidean distance between the soma and leafs
3	Number of bifurcations	12	Maximum path distance between the soma and leafs
4	Neuronal height, difference between maximum and minimum on the x-coordinates	13	Maximum branch order
5	Neuronal width, difference between maximum and minimum on the y-coordinates	14	Average contraction
6	Neuronal depth, difference between maximum and minimum on the z-coordinates	15	Total fragmentation
7	Average branch diameter	16	Average topological asymmetry
8	Total arborization length	17	Average Rall's power
9	Total arborization surface area	18	Average local bifurcation angle
		19	Average remote bifurcation angle
		20	Fractal dimension

B. PCA

Principal Component Analysis (PCA) [50] is a powerful statistical procedure used to reduce the dimensionality of a dataset [51]. Given an original dataset containing M features, and therefore having objects defined in an M -dimensional space, PCA can be used to define C new features, where $C \leq M$, that are optimal in describing the variance in the dataset. Therefore, the C new features can be regarded as keeping most of the information in the dataset, while defining a reduced set of features to describe the objects in the dataset. Each of the C new features, called principal components, are defined as linear combinations between the original M features. The direction of the largest variance in the original data is indicated by the first principal component, the second largest variance is expressed by the second principal component, and so on. The variance of the data expressed by each PCA component is usually presented as a percentage, calculated as a fraction between the variance of the component and the sum of variances among all components.

C. Scatter Distance

The objective of many clustering algorithms is to find the class assignments that optimize a merit figure [52]. One common criterion for class separability is based on the so-called *scatter matrices* [53], [54]. This criterion is defined as follows. First, we define the within-class scatter matrix as

$$M_w = \sum_{i=1}^L N_i \Sigma_i, \quad (1)$$

where L is the number of classes, N_i the number of objects in class i and Σ_i the covariance matrix of class i . Next, we define the between-class scatter matrix as

$$M_b = \sum_{i=1}^L N_i (\mu_i - \mu_*) (\mu_i - \mu_*)^T, \quad (2)$$

where, μ_i is a vector containing the average values of the features for objects in class i and μ_* is a vector containing the overall averages for all objects.

The scatter distance between classes can then be defined as

$$S = \text{tr}(M_w^{-1} M_b), \quad (3)$$

where $\text{tr}(M)$ is the trace of matrix M .

IV. NEURON DISMANTLING

As mentioned before, the dendritic arborization of a neuron can be seen as a set of binary trees [55] that are attached to the soma. Therefore, a tree data structure provides an intuitive representation of the neuronal morphology and can be explored to calculate morphological features. Traditionally, a set of features is associated with the neuronal arborization and the resulting quantification is used to define classes for the neurons, and to compare neurons from distinct classes.

In the current study, we go one step further and analyze to what extent neuronal classes can be described when observing parts, instead of the whole neuron. More specifically, we dismantle the neuronal dendritic arborization into its set of composing dendritic trees rooted in the soma. In Figure 4 we show an example of a neuron being dismantled into its six dendritic trees. Morphological features are then obtained for each dendritic tree. This procedure can be understood as a characterization of the neuron at different detail levels, which may allow a more robust description of the morphology, as well as to provide interesting insights about the organization of neuronal shape.

The adopted description of neurons can be interpreted as follows. In Figure 5 we consider two neuronal types, A and B. Such types are defined by a region in a feature space, which characterize the typical morphology of each type. Nevertheless, each distinct neuron, being it type A or B, belongs to different positions in such a space. Some neurons might even have characteristics that are not typical of their class, and therefore be located outside the class region. Furthermore, we consider that each neuron is in turn defined by a set of trees. We expect that trees from the same neuron will share similar properties. Yet, a given tree might not be typical of the neuron.

V. RESULTS AND DISCUSSION

We initiated by comparing average values for features calculated over whole neurons with those obtained from their parts, i.e. the respective sets of neuronal trees. First, the

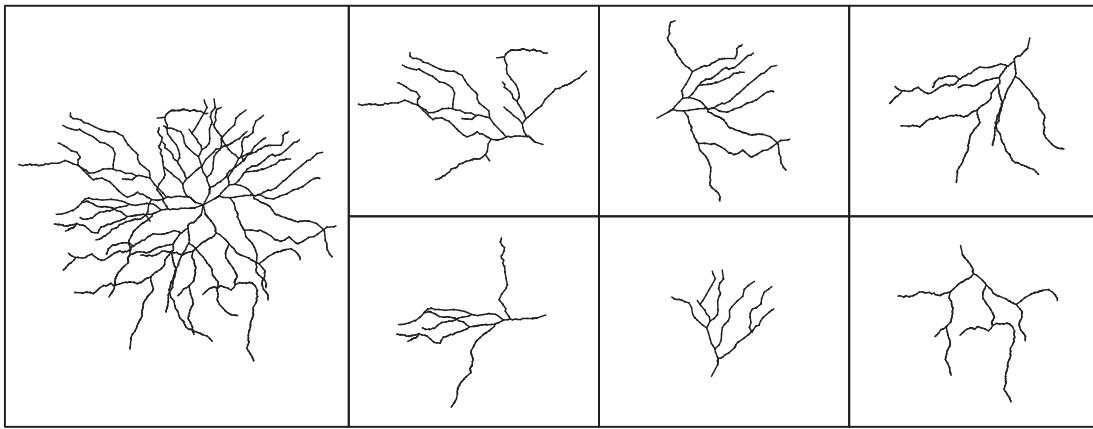


Fig. 4. Example of dismantling a neuron into a set of trees. The original neuron (left) contains six trees attached to the soma. These trees are shown on the right.

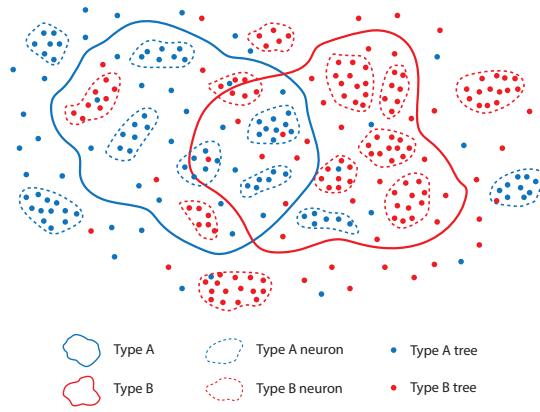


Fig. 5. The figure represents a 2-dimensional feature space used to characterize a set of neurons belonging to two classes, A and B. Solid lines indicate typical regions. Neurons are represented by dashed line groups.

morphological properties of neurons of the four considered classes were calculated. In Table II we present the average value and standard deviation of the considered features. Next, the same set of measurements was obtained for every dendritic tree contained in the dataset, and the calculated values are shown in Table III.

It should be observed that the neurons from all considered classes have six stems in the average. Pyramidal neurons from humans display a small standard deviation on the number of stems, while the other classes contain neurons with more diverse values.

A PCA technique was used to project the data from the original 20-dimensional space into a 2-dimensional one. This allows a proper visualization of the morphological similarities between the considered neuron classes. In Figure 6(a) we show the projection obtained using whole neurons. This result shows that mostly distinct regions were obtained for each neuronal type.

When applying the same procedure considering the neuronal trees, we obtain the result shown in Figure 6(b). In this case, the classes show larger overlap in the PCA space, meaning that

the dendritic trees tend to be more similar among themselves. This is mainly due to the information loss implied by the smaller number of neuronal compartments used for calculating each data point. Yet, the class separation was not completely undermined, indicating that the trees preserve features discriminating the neuron classes. It is also interesting to note that the neuronal trees resulted subdivided into subclasses. For example, rat interneurons seem to define a dense group on the upper right corner of the space, a smaller group can be observed below the main group. In addition, a third rat interneuron group could be defined by the outliers located to the right of the main group.

The PCA results indicate that the classes described by neuronal trees tend to be less separated in the feature space than when considering the whole cells. In order to better understand this effect, we estimated the scatter distances (presented in Section III-C) between each pair of neuron classes. All 20 features were included in the calculation. In Tables IV and V we show the computed distances when considering features related to, respectively, whole neurons and neuronal trees. It is clear that the two approaches result in widely distinct values for some class pairs. For instance, while the scatter distance between classes C2 and C4 is 11.47 for whole neuron features, it becomes 1.17 for neuronal tree ones. Therefore, this distance is 9.8 times larger for the former case (i.e. whole cells) than for the latter. Such differences are possibly a consequence of several causes, including the fact that the trees being smaller than the whole cells tend to produce largely different values of measurements such as diameter, arborization length, arborization surface area, etc. Yet, despite such changes, the relative distance between clusters may result relatively unaffected.

In order to better understand the above result, we calculated the ratio between the scatter distances obtained for the whole neuron and neuronal tree cases. The results are shown in Figure 7, where each circle represents a neuron class, and the numbers placed near the lines connecting circles indicate the respective ratio of scatter distances computed for the

TABLE II
MORPHOLOGICAL CHARACTERISTICS OF THE FOUR NEURON CLASSES ANALYZED. THE FEATURES WERE OBTAINED FROM THE WHOLE NEURONS. THE NUMBER AT THE LEFT OF EACH CLASS NAME INDICATES THE NUMBER OF NEURONS IN THE CLASS.

	Mouse ganglion (530)		Human pyramidal (530)		Mouse pyramidal (530)		Rat interneuron (550)	
	Avg	Std. dev.	Avg.	Std. dev.	Avg.	Std. dev.	Avg.	Std. dev.
Soma surface area (μm^2)	955.01	1225.78	1169.00	467.63	652.21	276.94	907.42	510.71
Number of stems	5.44	2.74	6.00	1.29	6.11	2.95	6.32	2.13
Number of bifurcations	73.76	41.99	25.60	7.61	28.75	28.73	203.55	143.78
Height (μm)	245.76	112.54	317.05	72.15	236.07	250.34	475.53	231.02
Width (μm)	274.85	127.23	301.78	73.04	436.07	413.06	638.38	293.54
Depth (μm)	22.58	21.14	102.71	17.84	50.32	41.95	185.84	128.33
Avg. branch diameter (μm)	0.83	0.94	1.03	0.24	0.68	0.49	0.33	0.15
Total length (μm)	4674.64	1821.82	3777.90	1187.58	3097.25	3696.01	17555.14	8954.96
Total surface area (μm^2)	15604.31	26110.79	12016.50	3935.43	6113.42	7491.46	17558.46	13061.73
Total volume (μm^3)	15586.25	36156.97	10274.76	4852.46	3897.63	3937.08	7020.61	6433.81
Max. Euc. dist. (μm)	227.59	98.17	255.16	47.57	353.95	340.34	613.74	270.50
Max. path dist. (μm)	290.87	130.30	317.09	59.87	443.95	457.10	999.70	406.39
Maximum branch order	16435.58	20143.35	1158.57	550.55	20537.86	45339.70	100601.17	91793.83
Average Contraction	0.88	0.04	0.89	0.03	0.87	0.06	0.83	0.04
Total fragmentation	3062.32	2836.02	431.79	162.77	3760.58	6233.94	10702.20	6964.99
Average top. asymmetry	0.50	0.07	0.42	0.08	0.51	0.12	0.55	0.06
Average Rall's power	9.54	17.21	6.43	5.09	10.76	16.04	27.00	33.66
Average local bif. angle	80.59	18.10	66.33	7.61	74.08	14.93	86.61	4.78
Average remote bif. angle	73.71	10.08	56.53	6.87	63.81	14.06	75.93	6.79
Fractal dimension	1.03	0.02	1.04	0.01	1.03	0.02	1.05	0.02

TABLE III
MORPHOLOGICAL CHARACTERISTICS OF THE FOUR NEURON CLASSES ANALYZED. THE FEATURES WERE OBTAINED CONSIDERING THE NEURONAL TREES. THE NUMBER AT THE LEFT OF EACH CLASS NAME INDICATES THE NUMBER OF TREES BELONGING TO THE CLASS.

	Mouse ganglion (2881)		Human pyramidal (3177)		Mouse pyramidal (3237)		Rat interneuron (3473)	
	Avg	Std. dev.	Avg.	Std. dev.	Avg.	Std. dev.	Avg.	Std. dev.
Soma surface area (μm^2)	27.90	72.50	335.87	324.26	16.13	29.38	28.33	27.35
Number of stems	1.00	0.02	0.99	0.09	1.00	0.04	1.00	0.02
Number of bifurcations	14.36	21.73	5.09	3.26	5.54	9.33	33.02	81.52
Height (μm)	119.65	83.63	138.40	76.07	88.93	132.01	151.73	181.91
Width (μm)	128.62	94.85	140.63	72.56	114.97	176.22	202.02	232.07
Depth (μm)	9.79	12.88	63.15	31.46	22.96	25.12	73.31	80.62
Avg. branch diameter (μm)	0.65	0.80	1.45	1.37	0.61	0.51	0.65	0.34
Total length (μm)	857.39	1072.02	671.43	435.87	503.40	1093.52	2770.11	6213.00
Total surface area (μm^2)	2702.92	8761.73	2181.85	1165.08	882.97	2044.06	2619.33	5613.28
Total volume (μm^3)	1782.82	9393.96	1071.39	1295.79	231.47	739.74	431.97	981.57
Max. Euc. dist. (μm)	143.95	84.90	179.01	70.37	137.00	189.06	222.32	220.05
Max. path dist. (μm)	194.01	111.90	260.50	88.57	179.65	247.45	328.60	351.06
Maximum branch order	3017.12	8555.50	193.53	206.00	3358.04	16033.11	15902.81	49519.20
Average contraction	0.87	0.05	0.82	0.09	0.83	0.07	0.82	0.06
Total fragmentation	564.81	1113.73	74.58	54.53	617.99	1635.15	1694.37	3992.69
Average top. asymmetry	0.42	0.25	0.41	0.25	0.42	0.28	0.48	0.24
Average Rall's power	1.92	6.17	1.24	1.43	1.85	4.40	4.70	13.14
Average local bif. angle	63.47	38.85	53.45	29.76	55.24	38.62	71.44	34.16
Average remote bif. angle	56.02	35.10	45.69	25.78	45.66	33.74	57.75	30.81
Fractal dimension	1.04	0.03	1.04	0.14	1.05	0.07	1.05	0.05

TABLE IV
SCATTER DISTANCES BETWEEN ALL PAIRS OF NEURON CLASSES, WHEN CONSIDERING WHOLE NEURON FEATURES.

C2	C3	C4	
C1	11.59	2.29	3.41
C2	-	3.12	11.47
C3	-	-	2.64

TABLE V
SCATTER DISTANCES BETWEEN ALL PAIRS OF NEURON CLASSES, WHEN CONSIDERING NEURONAL TREE FEATURES.

C2	C3	C4	
C1	4	0.65	0.7
C2	-	2.33	1.17
C3	-	-	0.28

two classes connected by the line. For example, the line connecting class C2 to C4 indicates 9.8 (equal to $11.47/1.17$) fold increase. The results indicate that several of the relative differences have approximately similar values (i.e. 2.9, 3.5, 4.9 and 9.3, 9.8), confirming for these cases that the respective

measurements are scaled by approximately the same amount, preserving the relative distances. In fact, the largest differences are observed for the C4 class, composed of rat interneurons, suggesting that the trees in this group are less representative of the whole cell characteristics.

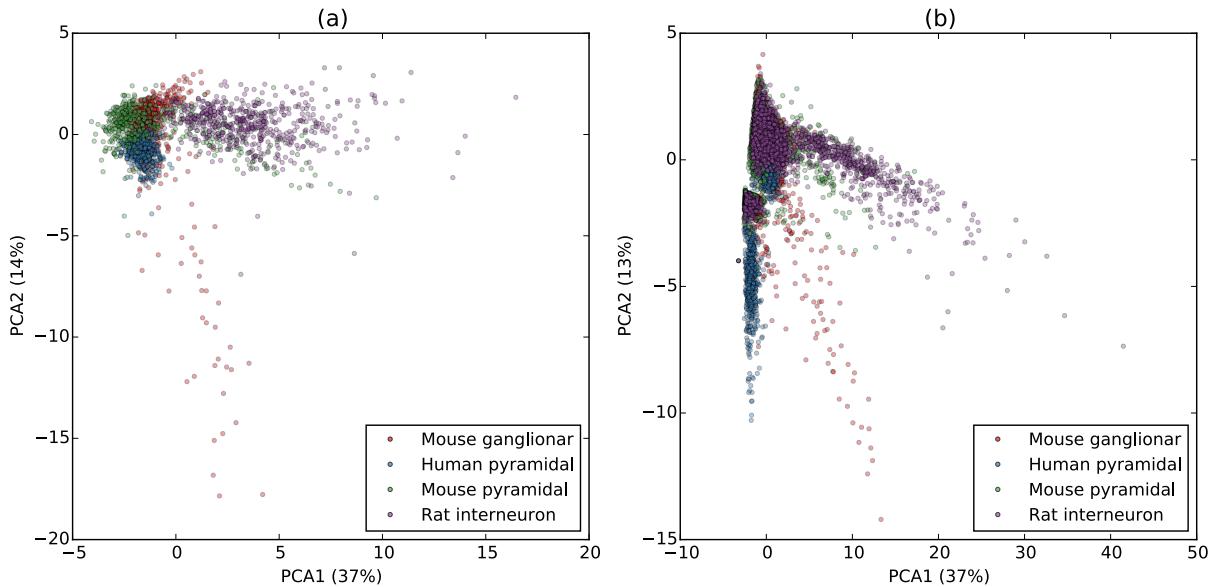


Fig. 6. Visualization of the first two principal components, found by the PCA algorithm, of the neuron dataset considered in the current study. The percentage of variance associated with each axis is indicated in the respective label of the axis. The principal components were calculated for features applied to the (a) whole neuron and (b) neuronal trees.

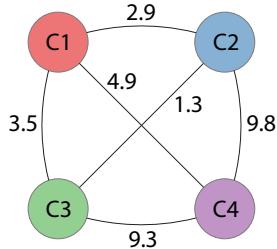


Fig. 7. Scatter distances between pairs of neuron classes. Each circle represents a class, and the relative distance between given two classes is indicated in the line connecting the respective circles. Please refer to Figure 2 for the name of each class.

VI. CONCLUSION

In this paper, we present a method to characterize neurons using their dendritic trees instead of the respective whole cells. The results obtained for 4 types of cells from the Neuromorpho database indicate that the features characterizing the neuronal morphology extend from the whole cells to the smaller spatial scale of the respective neuronal trees, suggesting that the relevant differentiating characteristics are already found at this level. This preservation of discriminability was not identical for all the four classes, with one of the categories deviating more markedly. Future works could consider more neuronal types, other features, and extend to the classification level. It would also be interesting to dismantle the neuronal cells not at the soma level, but along the hierarchy of the trees.

ACKNOWLEDGMENTS

CHC thanks FAPESP (Grant no. 15/18942-8) for financial support. LdFC thanks CNPq (Grant no. 307333/2013-2), NAP-

PRP-USP and FAPESP (Grant no. 11/50761-2) for support. EPC thanks FAPESP (Grant no. 15/01587-0) CAPES and NAP eScience - PRP - USP for support.

REFERENCES

- [1] H. Markram, "The human brain project," *Scientific American*, vol. 306, no. 6, pp. 50–55, 2012.
- [2] S. R. y Cajal and L. Azoulay, *Histologie du système nerveux de l'homme & des vertébrés*. Consejo superior de investigaciones científicas, Instituto Ramon Y Cajal, 1955.
- [3] R. Armañanzas and G. A. Ascoli, "Towards the automatic classification of neurons," *Trends in neurosciences*, vol. 38, no. 5, pp. 307–318, 2015.
- [4] E. R. Kandel, J. H. Schwartz, T. M. Jessel *et al.*, *Principles of neural science*. McGraw-hill New York, 2000, vol. 4.
- [5] H. S. Seung and U. Sümbül, "Neuronal cell types and connectivity: lessons from the retina," *Neuron*, vol. 83, no. 6, pp. 1262–1272, 2014.
- [6] Y. Uji, H. Yamamura *et al.*, "Morphological classification of retinal ganglion cells in mice," *Journal of Comparative Neurology*, vol. 356, no. 3, pp. 368–386, 1995.
- [7] L. M. McGarry, A. M. Packer, E. Fino, V. Nikolenko, T. Sippy, and R. Yuste, "Quantitative classification of somatostatin-positive neocortical interneurons identifies three interneuron subtypes," *Front. in neural circuits*, vol. 4, p. 12, 2010.
- [8] M. Botta and L. W. Swanson, "The neuron classification problem," *Brain res rev*, vol. 56, no. 1, pp. 79–88, 2007.
- [9] J. DeFelipe, "Cortical interneurons: from cajal to 2001." *Progress in brain research*, vol. 136, pp. 215–238, 2001.
- [10] N. G. Bazán and R. N. Lolley, *Neurochemistry of the Retina: Proceedings of the International Symposium on the Neurochemistry of the Retina Held in Athens, Greece, August 28-September 1, 1979*. Elsevier, 2013.
- [11] C. H. Comin and L. da Fontoura Costa, "Shape, connectedness and dynamics in neuronal networks," *Journal of neuroscience methods*, vol. 220, no. 2, pp. 100–115, 2013.
- [12] A. Mottini, X. Descombes, and F. Besse, "Axonal tree classification using an elastic shape analysis based distance," in *Biomedical Imaging (ISBI), 2014 IEEE 11th International Symposium on*. IEEE, 2014, pp. 850–853.
- [13] U. Sümbül, S. Song, K. McCulloch, M. Becker, B. Lin, J. R. Sanes, R. H. Masland, and H. S. Seung, "A genetic and computational approach to structurally classify neuronal types," *Nature communications*, vol. 5, 2014.

- [14] M. Halavi, K. A. Hamilton, R. Parekh, and G. A. Ascoli, "Digital reconstructions of neuronal morphology: three decades of research trends," *Front Neurosci*, vol. 6, no. 49, pp. 1–11, 2012.
- [15] I. D. Ruz and S. R. Schultz, "Localising and classifying neurons from high density mea recordings," *Journal of neuroscience methods*, vol. 233, pp. 115–128, 2014.
- [16] T. O. Sharpee, "Toward functional classification of neuronal types," *Neuron*, vol. 83, no. 6, pp. 1329–1334, 2014.
- [17] C. M. Teeter and C. F. Stevens, "A general principle of neural arbor branch density," *Current Biology*, vol. 21, no. 24, pp. 2105–2108, 2011.
- [18] J. A. Hosp, M. Strüber, Y. Yanagawa, K. Obata, I. Vida, P. Jonas, and M. Bartos, "Morpho-physiological criteria divide dentate gyrus interneurons into classes," *Hippocampus*, vol. 24, no. 2, pp. 189–203, 2014.
- [19] T. Zhao and S. M. Plaza, "Automatic neuron type identification by neurite localization in the drosophila medulla," *arXiv preprint arXiv:1409.1892*, 2014.
- [20] Y. Lu, L. Carin, R. Coifman, W. Shain, and B. Roysam, "Quantitative arbor analytics: Unsupervised harmonic co-clustering of populations of brain cell arbors based on l-measure," *Neuroinf*, vol. 13, no. 1, pp. 47–63, 2015.
- [21] G. A. Ascoli, D. E. Donohue, and M. Halavi, "Neuromorpho.org: a central resource for neuronal morphologies," *The J of Neurosc*, vol. 27, no. 35, pp. 9247–9251, 2007.
- [22] R. Scorcioni, S. Polavaram, and G. A. Ascoli, "L-measure: a web-accessible tool for the analysis, comparison and search of digital reconstructions of neuronal morphologies," *Nature protocols*, vol. 3, no. 5, pp. 866–876, 2008.
- [23] R. Santana, L. McGarry, C. Bielza, P. Larrañaga, and R. Yuste, "Classification of neocortical interneurons using affinity propagation," *Front neur circ*, vol. 7, p. 185, 2013.
- [24] B. Mihaljević, R. Benavides-Piccione, C. Bielza, J. DeFelipe, and P. Larrañaga, "Bayesian network classifiers for categorizing cortical gabaergic interneurons," *Neuroinformatics*, vol. 13, no. 2, pp. 193–208, 2015.
- [25] T. Gillette and G. Ascoli, "Topological characterization of neuronal arbor morphology via sequence representation. i," *Motif analysis*, 2015.
- [26] T. A. Gillette, P. Hosseini, and G. A. Ascoli, "Topological characterization of neuronal arbor morphology via sequence representation: II-global alignment," *BMC bioinformatics*, vol. 16, no. 1, p. 209, 2015.
- [27] R. Parekh and G. A. Ascoli, "Neuronal morphology goes digital: a research hub for cellular and system neuroscience," *Neuron*, vol. 77, no. 6, pp. 1017–1038, 2013.
- [28] S. Nanda, M. M. Allaham, M. Bergamino, S. Polavaram, R. Armañanzas, G. A. Ascoli, and R. Parekh, "Doubling up on the fly: Neuromorpho.org meets big data," *Neuroinformatics*, vol. 13, no. 1, p. 127, 2015.
- [29] R. Parekh, R. Armañanzas, and G. A. Ascoli, "The importance of metadata to assess information content in digital reconstructions of neuronal morphology," *Cell and tissue research*, vol. 360, no. 1, pp. 121–127, 2015.
- [30] J. S. Mirsky, P. M. Nadkarni, M. D. Healy, P. L. Miller, and G. M. Shepherd, "Database tools for integrating and searching membrane property data correlated with neuronal morphology," *Journal of neuroscience methods*, vol. 82, no. 1, pp. 105–121, 1998.
- [31] T. L. Saito, M. Ohtani, H. Sawai, F. Sano, A. Saka, D. Watanabe, M. Yukawa, Y. Ohya, and S. Morishita, "Scmd: *Saccharomyces cerevisiae* morphological database," *Nucleic acids research*, vol. 32, no. suppl 1, pp. D319–D322, 2004.
- [32] C. H. Comin, J. Tejada, M. P. Viana, A. C. Roque, and L. d. F. Costa, "Archetypes and outliers in the neuromorphological space," in *The Computing Dendrite*. Springer, 2014, pp. 41–59.
- [33] S. M. Sunkin, L. Ng, C. Lau, T. Dolbeare, T. L. Gilbert, C. L. Thompson, M. Hawrylycz, and C. Dang, "Allen brain atlas: an integrated spatio-temporal portal for exploring the central nervous system," *Nucleic acids research*, p. gks1042, 2012.
- [34] G. A. Ascoli, "Mobilizing the base of neuroscience data: the case of neuronal morphologies," *Nature Reviews Neuroscience*, vol. 7, no. 4, pp. 318–324, 2006.
- [35] ——, "Successes and rewards in sharing digital reconstructions of neuronal morphology," *Neuroinformatics*, vol. 5, no. 3, pp. 154–160, 2007.
- [36] Y. Kawaguchi, "Groupings of nonpyramidal and pyramidal cells with specific physiological and morphological characteristics in rat frontal cortex," *Journal of neurophysiology*, vol. 69, no. 2, pp. 416–431, 1993.
- [37] N. Spruston, "Pyramidal neurons: dendritic structure and synaptic integration," *Nature Reviews Neuroscience*, vol. 9, no. 3, pp. 206–221, 2008.
- [38] S. Cobb, E. Buhl, K. Halasy, O. Paulsen, and P. Somogyi, "Synchronization of neuronal activity in hippocampus by individual gabaergic interneurons," 1995.
- [39] E. Sernagor, S. J. Eglen, and R. O. Wong, "Development of retinal ganglion cell structure and function," *Prog in retina and eye research*, vol. 20, no. 2, pp. 139–174, 2001.
- [40] E. Kaplan and R. M. Shapley, "The primate retina contains two types of ganglion cells, with high and low contrast sensitivity," *Proceedings of the National Academy of Sciences*, vol. 83, no. 8, pp. 2755–2757, 1986.
- [41] R. Shapley and V. H. Perry, "Cat and monkey retinal ganglion cells and their visual functional roles," *Trends in Neurosciences*, vol. 9, pp. 229–235, 1986.
- [42] L. Stanford and S. M. Sherman, "Structure/function relationships of retinal ganglion cells in the cat," *Brain research*, vol. 297, no. 2, pp. 381–386, 1984.
- [43] W. Rall, "Core conductor theory and cable properties of neurons," *Comprehensive physiology*, 2011.
- [44] P. Poirazi and B. W. Mel, "Impact of active dendrites and structural plasticity on the memory capacity of neural tissue," *Neuron*, vol. 29, no. 3, pp. 779–796, 2001.
- [45] E. K. Scott and L. Luo, "How do dendrites take their shape?" *Nat neurosci*, vol. 4, no. 4, pp. 359–365, 2001.
- [46] Y.-N. Jan and L. Y. Jan, "Branching out: mechanisms of dendritic arborization," *Nature Reviews Neuroscience*, vol. 11, no. 5, pp. 316–328, 2010.
- [47] C. Koch, *Biophysics of computation: information processing in single neurons*. Oxford university press, 2004.
- [48] W. Rall, "Electrophysiology of a dendritic neuron model," *Biophysical journal*, vol. 2, no. 2 Pt 2, p. 145, 1962.
- [49] W. B. Marks and R. E. Burke, "Simulation of motoneuron morphology in three dimensions. i. building individual dendritic trees," *Journal of Comparative Neurology*, vol. 503, no. 5, pp. 685–700, 2007.
- [50] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.
- [51] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.
- [52] R. Xu, D. Wunsch *et al.*, "Survey of clustering algorithms," *Neural Networks, IEEE Transactions on*, vol. 16, no. 3, pp. 645–678, 2005.
- [53] K. Fukunaga, *Introduction to statistical pattern recognition*. Academic press, 2013.
- [54] K. J. Cios, W. Pedrycz, and R. W. Swiniarski, *Data Mining and Knowledge Discovery*. Springer, 1998.
- [55] L. R. Foulds, *Graph theory applications*. Springer Science & Business Media, 2012.

Detecting Wildlife in Uncontrolled Outdoor Video using Convolutional Neural Networks

Connor Bowley*, Alicia Andes†, Susan Ellis-Felege†, Travis Desell*

Department of Computer Science*

Department of Biology†

University of North Dakota

Grand Forks, North Dakota 58202

Email: connor.bowley@und.edu, alicia.andes@my.und.edu, susan.felege@email.und.edu, tdesell@cs.und.edu

Abstract—This paper explores the use of Convolutional Neural Networks (CNNs) to detect Interior Least Tern in uncontrolled outdoor videos for the Wildlife@Home project. To be able to use CNNs on this video, this work developed strategies to bridge the gap between video collected by wildlife biologists and the methodologies common for training and testing CNNs by utilizing a striding methodology to extract positive and negative training examples of a fixed size. Then in order to efficiently run trained CNNs over full videos, software was developed using OpenCL which was capable of utilizing multiple GPUs and other OpenCL capable compute devices concurrently. It was also shown that an already trained CNN can be further refined by training it further on new imagery, without having to retrain the whole network from scratch, saving significant time. Further, while the CNNs trained were only for detection of Interior Least Terns, they show promise for actually detecting behavior, as obvious peaks resulted for periods of video when a tern was in flight. To the authors' knowledge, this is the first attempt to utilize CNNs for the task of detecting wildlife in uncontrolled outdoor video.

I. INTRODUCTION

Over 100,000 hours of uncontrolled outdoor video of four different species – blue winged teal (*Anas discors*), Interior Least Tern (*Sternula antillarum*), piping plover (*Charadrius melanotos*) and sharptailed grouse (*Tympanuchus phasianellus* – have been collected for Wildlife@Home, with more being collected each field season. Such camera studies are popular in the field of avian ecology as they can reduce researcher impacts on animal behavior and also monitor animals in remote locations [1], [2]. In order to overcome these challenges, Wildlife@Home has been developed to employ both volunteer computing and crowd sourcing to quickly analyze wildlife video, as well as to investigate automated video analysis strategies using computer vision techniques.

Each of these species have different nesting behaviors and users (either volunteers through a crowd sourcing interface, or project scientists through a more advanced web portal) have been tasked with classifying them. Examples of behaviors are *On Nest*, *Off Nest*, *Brooding*, *Flying*, *Foraging*, and *Feeding*. While users are observing the nests, they create a time-series for each video specifying when these events begin and end. Each event in the time-series has a type, start time and end time (see Figure 1). These users have provided a large body

of observations, which has led to Wildlife@Home's first data release of human annotated video¹ [3].

The goal of Wildlife@Home is not simply to crowd source the observation of this video, but rather to use these human observations to train effective computer vision algorithms to automate the analysis of this video. In order for Wildlife@Home's ecological researchers to be able to understand this vast amount of information and evaluate biological hypotheses of interest, these automated techniques are required due to the sheer scale of the data and the fact that human viewers cannot reliably watch it and make observations within it in a reasonable amount of time. Previous work has been done utilizing background subtraction methods to detect areas of interest within these videos [3], [4], however this work was limited in its applicability due to the fact that changing weather conditions and wind can drastically skew results. Further, background subtraction methods only have the ability to determine if activity is occurring, and not automatically detect different types of behavior.

In order to address the limitations of background subtraction, this paper investigates the preliminary use of convolutional neural networks (CNNs) to detect Interior Least Tern within video from Wildlife@Home. In particular, this problem comes with a set of unique challenges which are not addressed in typical CNN literature. First, the videos used have significantly higher resolution than typical input images to CNNs. The videos used in this study were 704x360 pixels, whereas many data sets used in CNN literature much smaller, such as 20x20 in the MNIST dataset [5], 32x32 in the CIFAR 10 and CIFAR 100 datasets [6], and 32x32 in the TinyImage dataset [7]. In other cases, input images are downsampled to something around 32x32 before being fed into the CNN, however this presents problems for this data as the species of interest only take up a fraction of each frame if they are present.

This work presents the use of a striding method to extract positive and negative examples from within video frames and generate fixed 32x32 pixel sized training and testing images, and then has developed new software using OpenCL to run the trained CNNs over entire videos and display which regions within the video contain the species of interest. Finally, as

¹http://csggrid.org/csg/wildlife/data_releases.php

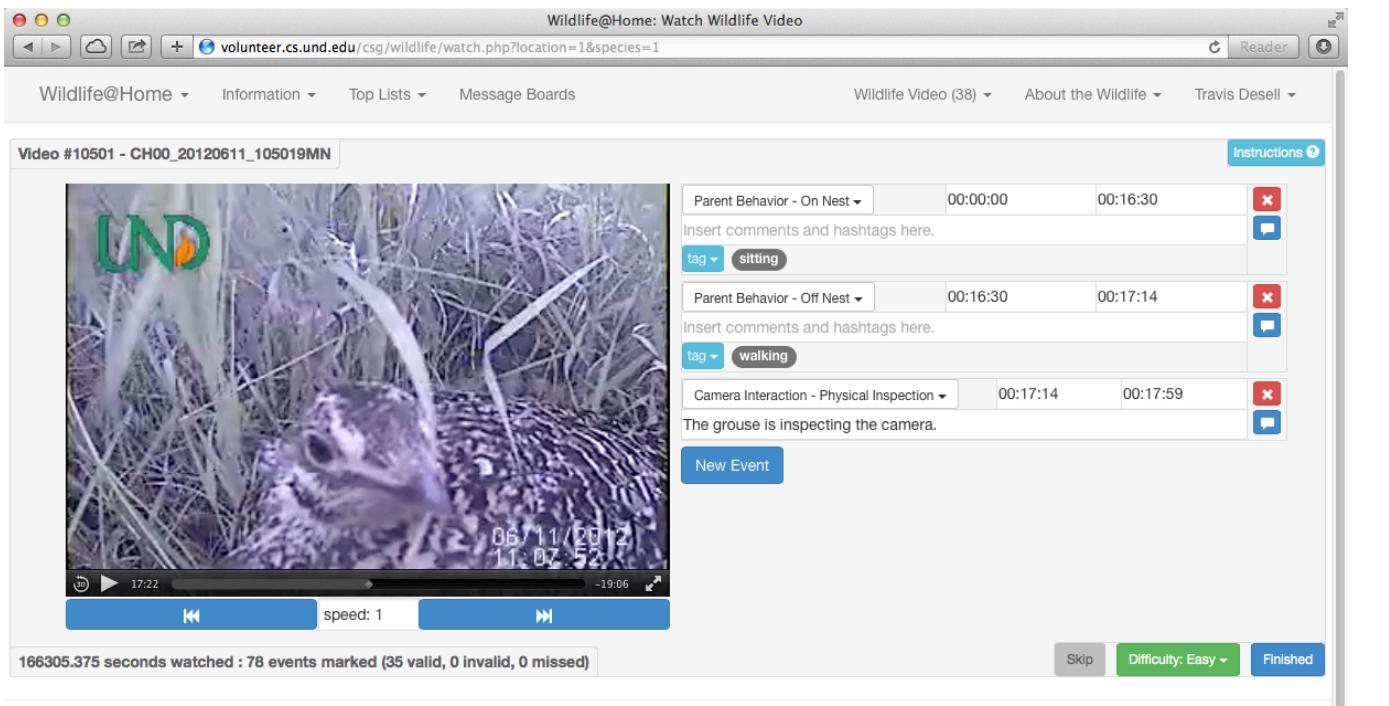


Fig. 1. An example of Wildlife@Home’s video viewing interface. Users are shown 30 minute to 2 hour long nesting videos, and can specify the start and end time for various events of interest, and provide tags and comments for additional detail. Users can also specify how difficult it was to determine events for the video and discuss segments of the video on the project’s message boards.

generating effective training data is an interactive process due to varying background imagery, this work also presents some valuable tricks for other researchers attempting to train CNNs for similar purposes, such as continuing the training of an already trained CNN using new example imagery to further refine its results while saving time from having to completely retrain a new CNN.

II. RELATED WORK

While there is a large body of work detecting humans and other objects within video (for surveys see [8], [9]), there is a growing amount of work in using computer vision to detect animals and animal events. The majority of this work with animals has been done in controlled laboratory settings, which simplifies the task of gathering video and animal detection. A common approach is to subtract a uniform background from the animals, which has been used to track white mice on black backgrounds [10]–[12] or in water [13]–[15]. Tracking and detecting behavior of fruit flies (*Drosophila*) [16]–[18] has been done in similar settings. Detection of particular actions or events has also been studied, such as vomiting of musk shrews [19], [20] using non-rigid body contour matching and various actions of a grasshopper [21] using spectral clustering [22].

Research has also been done in uncontrolled lab settings, without background subtraction or environmental controls. Sequential Monte Carlo methods, or particle filters [23]–[26], have been used to provide tracking with resiliency to unpredictable motion and non-linear measurement models,

and have been mostly used with insects such as ants [27] and bees [28], [29]. Tracking outlines of animals in their stalls using active contours has been used for larger animals like cows [30] and pigs [31]. Using multiple features (image abstractions such as anatomical and cage characteristics) to track rats in reflective and potentially scratched cages [32] and determine mice behaviors [33], [34] has been successfully used as well. Also of note, Jhaung et al. developed a manually annotated video database for training and testing a computer vision system for detecting behavior in mice in cages [35]–[37]. Weinstein has developed MotionMeerkat [38] to alleviate video stream data analysis by extracting frames with motion from videos, using Mixture of Gaussians [39] or Running Gaussian Average [40], [41] for foreground segmentation and then blob detection and thresholding to determine if a foreground object is present.

Considerably less research has been done using video taken in uncontrolled natural settings. Particle filters have been used to track multiple birds in the sky [42]; and data association methods have tracked and counted extremely large numbers of bats in noisy infrared video, taken as the bats leave their caves at night [43]. Face detection has also been used to classify species of African great apes using footage taken from video traps [44]. In settings most similar to this work, BearCam has been used to detect bears in the Arctic Circle during four hour daytime periods [45] by taking low level features such as image gradients and background differences and combining them into a mid-level *motion shapelet* [46]

using AdaBoost [47].

III. METHODOLOGY

The goal of this work was to create and train a CNN that could be used for the recognition of Least Tern in video and images. However, the video and images collected for Wildlife@Home are not of a consistent size due to different camera technology used by different researchers to gather the data. Further, most literature on convolutional neural networks involves training and testing data sets of small resolution, in the general range of 32x32 pixels. The Least Tern videos analyzed in this work are 704 x 480 pixels, of which the Least Tern only take up a small, but larger than 32x32 pixel portion. There may also be multiple Least Tern within a video.

In many ways, this illustrates a disconnect between the test data sets which see widespread use by the computer vision community, and how CNNs can actually be applied by wildlife researchers in the field. The following sections describe the process to bridge this gap, namely how training and testing images were extracted from the video in Section III-A, the architecture of the CNN used and how it was trained in Section III-B, and the software that was developed to run trained CNNs over videos and determine the presence or absence of Least Tern in Sections III-C and III-D.

A. Creating the Training Data

The training data that was available were videos from the first Wildlife@Home data release [3]. The videos used to generate training and testing data for this study had ids 58277, 58287 and 58307. These were videos of varying length that had been watched by volunteers who marked during what times there was an animal in the video and what the animal behavior was at that time (on nest, flying, walking, etc.). From these videos, regions were extracted from various frames that were of both positive and negative examples. The extracted regions were of varying sizes, as the wildlife could be of different sizes depending on how far they were from the camera. For positive examples, these extracted regions were then carefully cropped to minimize the existence of any 32 x 32 sub-images within a positive image that did not contain any pixels of the positive class (see Figure 2). For creating training data for the CNN, 32 x 32 sub-images were made by striding across each of the carefully cropped, variable sized training images. The striding process worked as follows. A 32 pixel x 32 pixel region starting in the upper lefthand corner was extracted. After extraction, the start point was moved right by some stride. Then another 32 x 32 region was extracted with the new start point. This was repeated until no more 32 x 32 regions could be extracted from the row. The start point was then moved back to the lefthand side and moved down by the same stride that was used to move horizontally and the previous steps were repeated to extract all images from the row. This was repeated until no more subimages could be extracted from the image. The striding process was automated with the stride used as a hyperparameter. Not all images were processed with the same stride. The amount of sub-images

TABLE I
CNN ARCHITECTURE

Layer Type	Layer Dims	Filter/ Pool Size	Stride	Number Filters	Padding
Input	32 x 32 x 3				
Convolutional	28 x 28 x 6	5	1	6	0
Max Pooling	14 x 14 x 6	2	2		
Convolutional	14 x 14 x 7	3	1	7	1
Convolutional	12 x 12 x 10	3	1	10	0
Max Pooling	4 x 4 x 10	3	3		
Convolutional	4 x 4 x 5	3	1	5	1
Fully Connected	1 x 1 x 2				

made from each image, which is dependent on the stride used, was determined in such a way to make the size of the classes approximately equal. Some of the images were overlapping, so some duplicate sub-images exist within the training data. In general, this meant reducing the number of negative examples and searching for additional positive examples as there is far more footage of background than the species of interest.

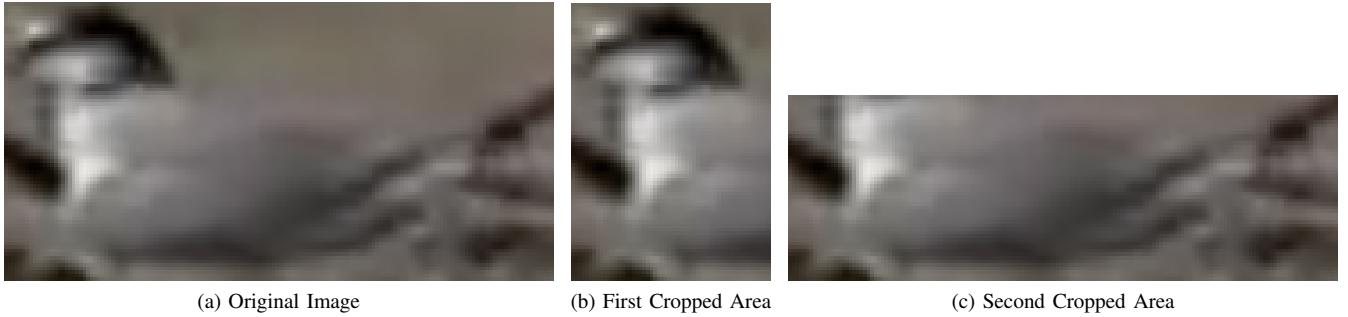
After the network was trained, new images were strided across and a prediction image was created using the output of the CNN (for examples see Figure 5).

B. Creating and Training the CNN

A convolutional neural network was created to learn from the training data. The architecture for the CNN is shown in Figure 3, and Table I provides specifics of how the layers were connected. The input is 32x32x3, 32x32 pixels by RGB. The first layer is a convolutional layer with filters of size 5x5x3 (3 because it is the previous depth), a stride of 1, 6 filters, and no zero padding. The resulting size after the convolution is 28x28x6 (6 because it is the number of filters). The rest of the layers follow the same pattern of the layer dimensions being the resulting size after the operation has completed.

A few considerations went into the design of the architecture for the CNN. Because CNNs are relatively slow and there are many hours of video in the Wildlife@Home data release, fewer filters were used in the convolutional layers to decrease run time. The downside to this is that there are less filters that could be learned. Because of the small number of output classes, it was thought that even with not many filters, the CNN could still learn effectively and have good accuracy. The final output is of size two because this network considers two options, the first being the image is not of an Interior Least Tern, the second being that it is.

The network was implemented and trained using a custom CNN implementation that was developed in C++ and OpenCL [48]. Using OpenCL allows for the network to train and run on most CPUs and GPUs, allowing for decreased training and run time. While there are a number of popular packages for training CNNs, notably Caffe [49] and MatConvNet [50], the endgoal of this research is to run the trained CNNs on the over 100,000 hours of Wildlife@Home videos on volunteered computers using BOINC [51], which precludes the use of Python (Caffe) or Matlab (MatConvNet) and requires



(a) Original Image

(b) First Cropped Area

(c) Second Cropped Area

Fig. 2. Example of cleaning training data. The cropping prevents the top right portion of the original image from becoming false positive training examples when the images are broken down into 32x32 subimages.

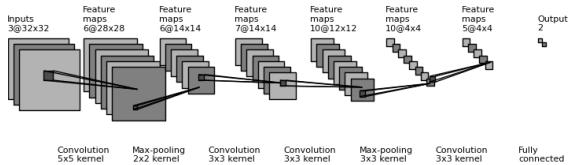


Fig. 3. Visualization of the CNN Architecture used in this work.

code to be modified with C/C++ BOINC library calls to run within BOINC clients.

The activation function used for the Convolutional Layers was Leaky RELU, a variant of the commonly used RELU function [52]. The implementation is a feed-forward convolutional neural network that trains using stochastic gradient descent backpropagation. It utilizes L2 regularization [53] and Nesterov Momentum [54] to decrease training time and help prevent the network converging at a non-optimal point. Weights for the convolutional layers were initialized based on a normal distribution with mean of 0 and standard deviation of $\sqrt{2/n}$ where n is the number of inputs to the neuron². For the final class outputs, a 2-way softmax classifier is used. The learning rate started at 10^{-3} and was cut in half every 5 epochs. The network initially trained for five epochs on the whole set of the training data.

C. Running over Full Images and Videos

After the CNN was trained on the full set of training data, it was run on videos by striding over 32x32 sub images within each frame. The Softmax classifier at the end of the network outputs a number between 0 and 1 for each possible class. This number roughly describes the confidence that the network has that the particular image is of the given class. To determine which parts of a full sized image contain Interior Least Tern and which do not, each pixel in the image has a list with elements describing the confidence that the pixel is of each class, very similar to the Softmax classifier. This will

²In this initialization a weight is not considered to be an input, but a bias is. Therefore the equation for n would be $f^2 + 1$ with f being the size of the filter (for a 3x3 filter, f would be 3).

be referred to as the pixel classifier. For each sub-image that the CNN is run on, every pixel of that sub-image has added to their pixel classifier the elements of the Softmax classifier for the sub-image after it is run through the network. Note that because sub-images may overlap, the elements of the pixel classifier may be larger than 1.

The CNN is strided across the whole image and each Softmax classifier is summed into the pixel classifiers. Once the whole image has been run through the CNN, the values of the pixel classifiers are used to determine the class the pixel belongs to. Red was chosen to represent pixels determined to be Interior Least Tern and blue was chosen to represent everything else. The more red the pixel, the more confident the CNN was that the pixel was of Interior Least Tern. The ratio of the squares of the pixel classifiers were used to determine the color according to these equations:

$$r = 255c_p^2 / \sum_{i=0}^n c_i^2 \quad (1)$$

$$b = 255c_n^2 / \sum_{i=0}^n c_i^2 \quad (2)$$

where r is the red pixel value, b is the blue pixel value, c_p and c_n are the classifier values for pixel for the positive and negative classifications. This assumes there are only two classifications, positive and negative. The green pixel value was always set to zero. For running over the videos, each frame was run through the CNN in the same way as described above for full images. The results from running over each frame were then used to make a new video. Also made from each video was a CSV file tracking the total red pixel value in each frame. To try to minimize counting of pixels that were primarily blue into this total, if a pixel's red pixel count was less than 150 of 255, it was not added to the total. This file was then used to create a line graph plotting the red pixel value against time (e.g., Figures 6, 7).

D. Improving Analysis Performance

After a working version of the video analysis software was created, another version was created that can utilize all available OpenCL devices on the machine that allow for

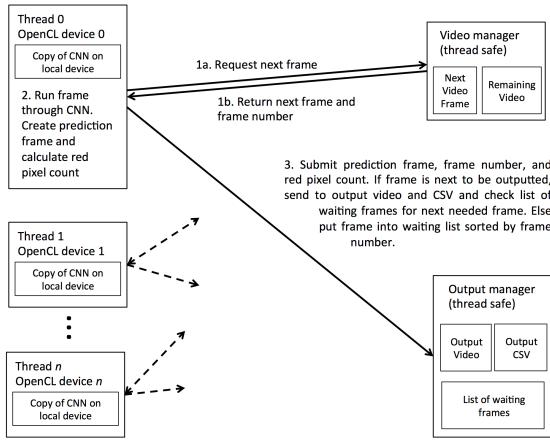


Fig. 4. Overview of the work stealing approach to utilize multiple OpenCL devices to quickly run a trained CNN over a video.

double precision floating point values (a requirement for the CNN). The software utilizes a work stealing strategy (see Figure 4), where each device is managed by a separate thread and has its own version of the CNN with the same weights. Each thread goes through a number of steps. First, the threads request a frame from a thread safe function that manages the input video. Second, it runs the CNN over that frame on its corresponding device. And last, it submits the new prediction and its corresponding frame number to a thread safe function that manages all completed frames. This submission function takes the prediction frame, and if it is the next frame to be put in the output video according to the frame number, it is sent to the output video. If not, it is put in a list sorted by frame number until it is up to be placed in the video. These steps are done by all threads in a loop until all frames have been run through the CNNs. This method of storing out of order frames allows faster devices to run through a greater proportion of frames without waiting for slower devices. This method is akin to using separate CNNs with the same weights on each OpenCL device and each CNN runs independently of the others. Thus adding more OpenCL devices will decrease runtime provided there are sufficient frames in the video and there is not a device slow enough that all other devices can finish all other frames in the time it takes the slow device to finish one frame. The implementation does not yet use MPI or OpenMP and therefore only works on single machines and not clusters.

IV. RESULTS

The network was trained originally for five epochs over a testing data set consisting of 72,951 sub-images (39.05% negative, 69.95% positive) taken from video 58277, which resulted in an accuracy of 95.6% on the training data. This network was then run on a larger set of test data of over 280,000 32 x 32 images taken from videos 58287 and 58307. This test set consisted of mostly novel images, however it

was not possible to perfectly guarantee no duplicates from the training data due to the similarity of the image backgrounds throughout the videos. This network achieved an accuracy of 82% on the test set with 77% of its errors results from false positives.

A. Video and Image Analysis

The initially trained network was tested on subsections of multiple videos with varying results. When run on one of the videos that a large amount of training data came from, it performed fairly well, although it did have a tendency to misclassify trees and ground stubble as tern (see Figure 5b). In light of this, more training data involving trees and ground stubble was acquired. To account for these new negative examples, the already trained weights were trained for an additional 2 epochs on a training set that was comprised of approximately 17,000 images with 69% of them being negative training data and the rest positive training data. In the videos and images that Wildlife@Home has, there is far more negative area than positive area in the frame. This is one reason that the percentage of negative training data was greater than positive in this new training set, and in general will be a problem for ecologists performing automated detection of wildlife.

After training for 2 epochs on the new training data, the results had less false positives, but still contained a fair amount of them (see Figure 5c). The CNN was then trained for another 2 epochs on the data and performed significantly better afterwards (see Figure 5d). As a note, the CNN that has the extra 2 epochs of training achieved an accuracy of 80% on the test data with 62% of its errors resulting from false positives and the CNN that trained for the 4 extra epochs got 78% with only 37% of its errors resulting from false positives. Although the networks with less extra training performed better on the test data, the networks with extra training performed better on the videos. It is believed this is because proportion of false positives on the earlier networks was higher than in the later networks.

To approximate how well the CNN performed over entire videos, the output file containing the total amount of red pixels in each frame of video was analyzed. This file was shown to have some interesting properties that correspond well to events in the video, as shown in Figure 6. This chart was on the 2 minutes 15 seconds of video that the images from Figure 5 came from. From times 0–28 seconds and 112–135 seconds, the tern is sitting on its nest, which corresponds to the raised red pixel count. At about 28 seconds, the bird flies off of its nest and off of the screen. This corresponds to the peak in the red pixel count. There is a peak because the body of the bird in flight is larger than the when it's on the nest (compare Figure 5 with Figure 8). The bird flies through the picture but does not land at around 72 seconds and flies in to land at its nest at about 110 seconds. Between times of about 30–70 seconds and 73–109 seconds, the bird is off of the screen. The amount of red pixels that are seen in those times comes from both noise from small amounts of misclassified areas. Using a

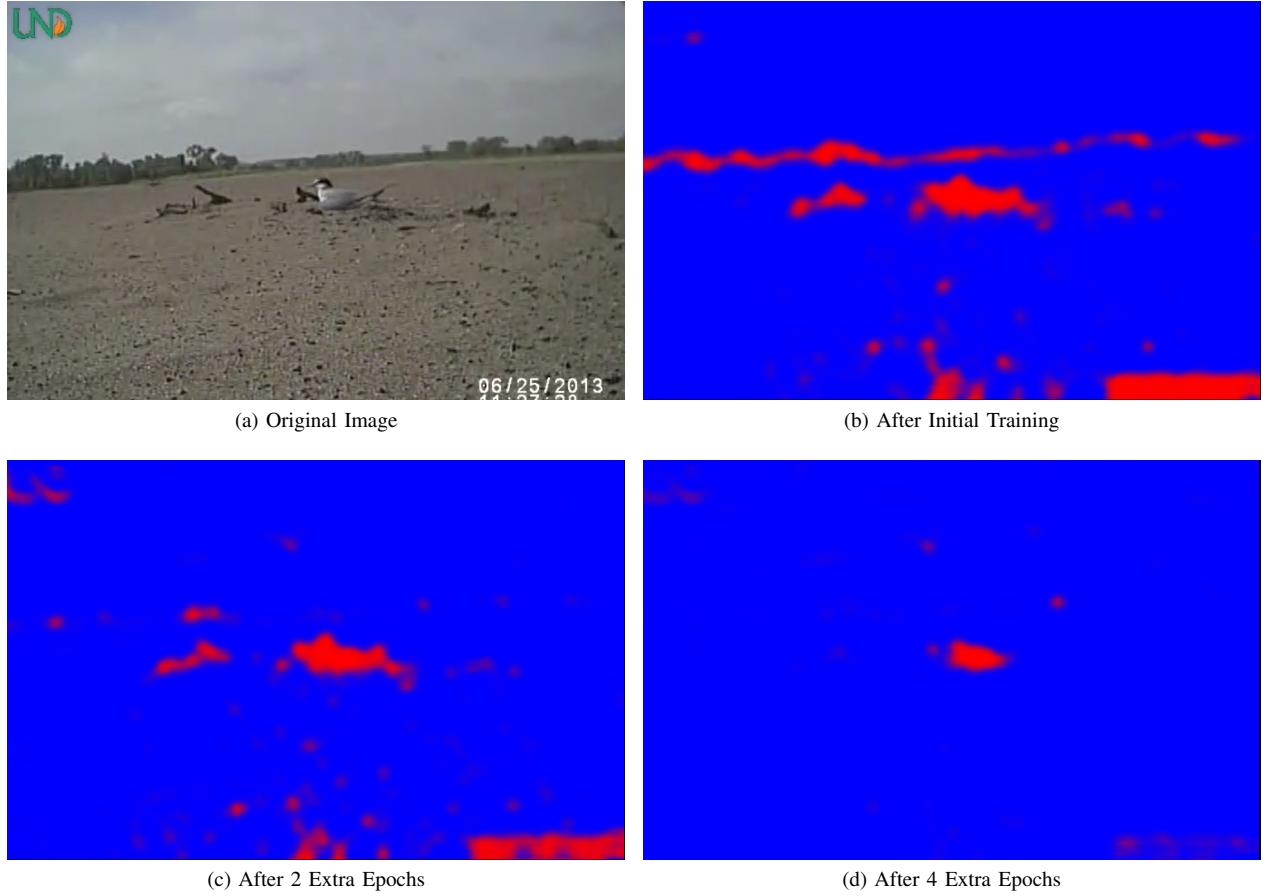


Fig. 5. Predictions after different amounts of training. The stride across image was 3 pixels in both directions.

TABLE II
PERFORMANCE RESULTS ON TWO MACHINES

Computer	Devices	Time (h:mm:ss)	Seconds/Frame
Mac Pro	1 GPU	48:07	5.12
Mac Pro	CPU	32:01	3.41
Mac Pro	2 GPUs	27:34	2.93
Mac Pro	CPU and 1 GPU	20:45	2.21
Mac Pro	CPU and 2 GPUs	17:27	1.86
MacBook Pro	GPU	1:17:02	8.20
MacBook Pro	CPU	35:06	3.73
MacBook Pro	CPU and GPU	26:03	2.77

These results are from running on 56 seconds of video (564 frames) with a stride of 15 in both directions.

method similar to momentum for training CNN weights could be used to smooth out some of the jagged-ness of the chart.

B. Performance

One of the downsides of CNNs is that they are typically quite slow. This was also the case for running the CNNs across the videos. One of the machines used for testing was a 2013 MacBook Pro with a 2.4 GHz Intel i7, 8 GB of RAM, and a NVIDIA GeForce GT 650M graphics card with 1GB VRAM. On this machine, it took the non-parallel version 1 hour, 27 minutes, and 53 seconds to run on 2 minutes and 15 seconds of 704 x 480 video. The video was shot at 10 FPS for a

total of 1,353 frames. The stride size across the video was 15 pixels in both the horizontal and vertical directions. Note that this run was using OpenCL code on the CPU only³. The parallel version was run on the same video and machine and took 1 hour, 5 minutes, and 3 seconds to run, which is almost a 26% speedup. This code ran on both the CPU and GPU in the machine.

Another machine used for testing was a 2013 Mac Pro with a 3.5 GHz 6-Core Intel Xeon E5 and two AMD FirePro D700s with 6144MB of VRAM each. A short test video was run on this machine. The video was 6 seconds long (60 frames). The stride was 15 in both directions. Using only the CPU, it ran it in 3 minutes, 20 seconds, which is 3.33 seconds per frame. Using only one of the GPUs it ran it in 4 minutes, 31 seconds, which is 4.52 seconds per frame. Using the parallel code running the CPU and both GPUs it ran in 1 minute, 37 seconds, which is 1.62 seconds per frame. The parallel code had an performance increase of 64% over one GPU alone and an increase of 51.5% over the CPU alone. The amount of

³Because of the relatively shallow depth of the network and the small number of filters used, the code ran faster on the CPU than the GPU. This is presumably because of the larger ratio of memory transfers to computation for small networks. Networks with a larger amount of filters tested using the same CNN code did run faster on the GPU than the CPU.

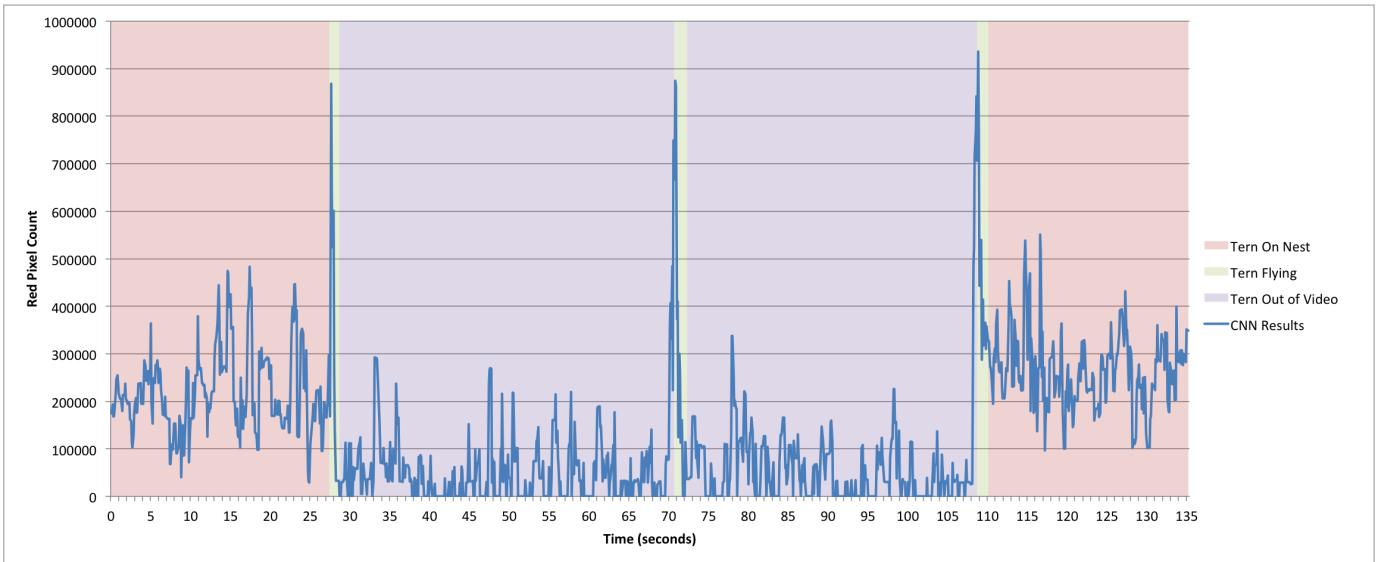


Fig. 6. Presence of red pixels from the pixel classifier, which represent a measure of how much of each frame was an Interior Least Tern. This was run on a portion of Video 58277.

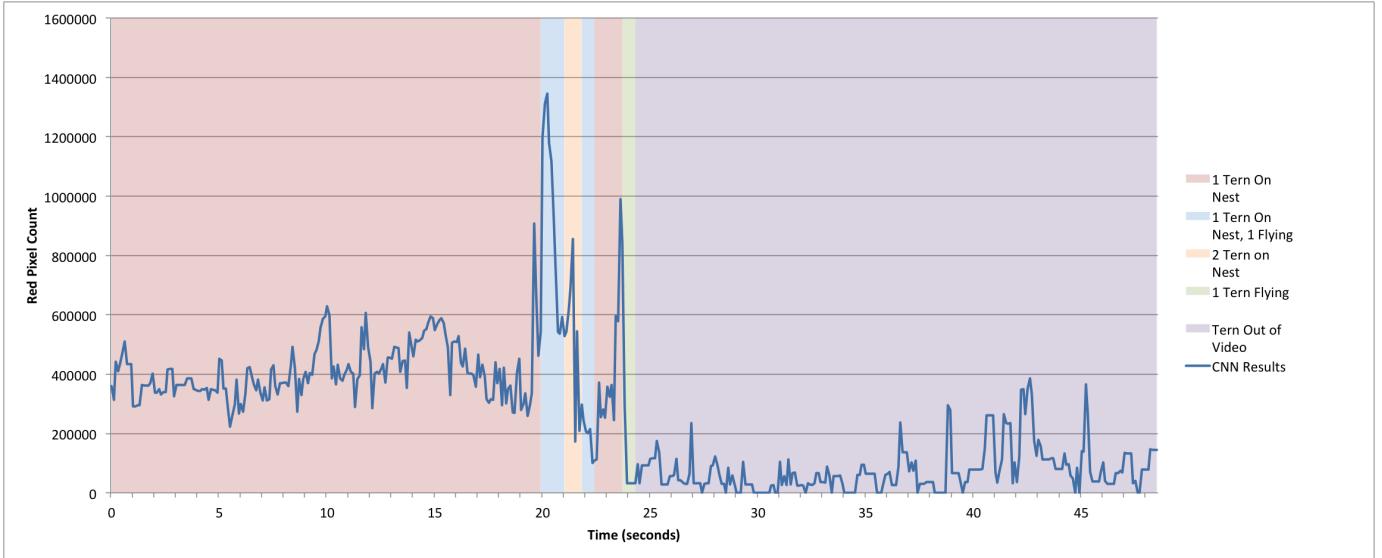


Fig. 7. A chart from a run on a portion of Video 58277 where multiple tern were in video at the same time.

speedup one can expect to get from using the parallel code is dependent on the amount of OpenCL devices in the machine. For additional performance results using differing amounts of OpenCL devices, see Table II.

V. CONCLUSION

To the authors' knowledge, this work presents the first use of convolutional neural networks to detect wildlife within uncontrolled outdoor video. These preliminary results show a strong ability for a well trained CNN to detect Interior Least Terns within video collected by Wildlife@Home. To accomplish this, this work introduced strategies to bridge the gap between video collected by wildlife biologists and the current methodology for training and testing CNNs, by utilizing a

striding methodology to extract positive and negative training examples of a fixed size. In order to efficiently run trained CNNs over full videos, software was developed using OpenCL which was capable of utilizing multiple GPUs and other OpenCL capable compute devices concurrently. It was also shown that an already trained CNN can be further refined by training it further on new imagery, without having to retrain the whole network from scratch, which can save significant time. Further, while the CNNs trained were only for detection of Interior Least Terns, they show promise for actually detecting behavior, as obvious peaks resulted for periods of video when a tern was in flight.

This work lays the groundwork for significant future study,

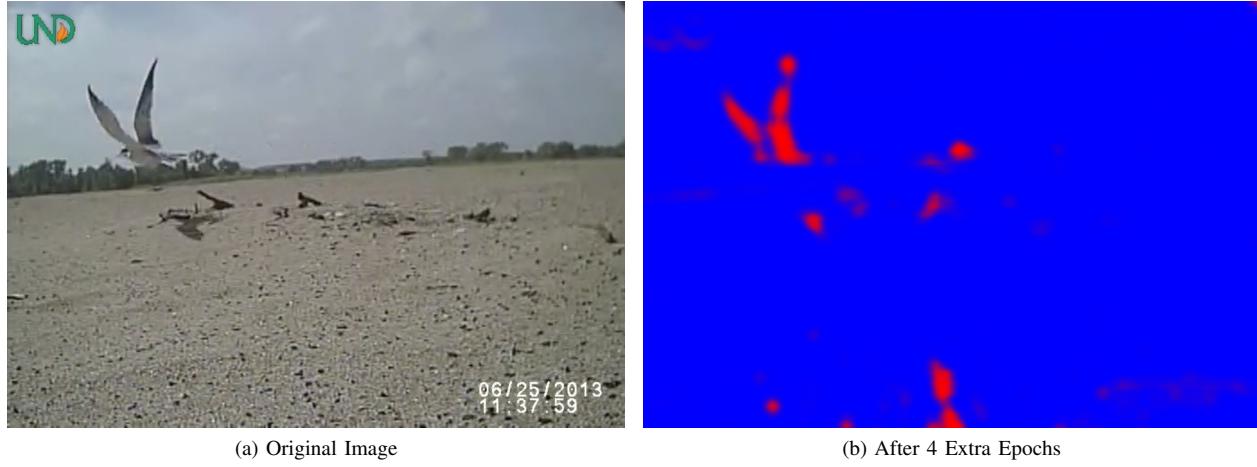


Fig. 8. Predictions on a frame containing a flying Interior Least Tern.

in particular to investigate how generalizable these trained CNNs are over the entire dataset of Wildlife@Home videos. Additional CNNs will need to be trained for the other species involved in the project. With these CNNs trained, it will be possible to utilize the over 3,000 volunteered computers at Wildlife@Home to analyze the entire data set of videos and compare these to volunteer and expert observations. This will provide a chance to further refine and investigate different CNN architectures to determine which are the most capable of determining wildlife behavior in this challenging data set.

ACKNOWLEDGMENTS

We appreciate the support and dedication of the Wildlife@Home citizen scientists who have spent significant amounts of time watching video. This work has been partially supported by the National Science Foundation under Grant Number 1319700. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. Funds to collect data in the field were provided by the U.S. Geological Survey.

REFERENCES

- [1] W. A. Cox, M. S. Pruitt, T. J. Benson, J. C. Scott, and F. R. Thompson, "Development of camera technology for monitoring nests," *Video surveillance of nesting birds. Studies in Avian Biology*, vol. 43, pp. 185–210, 2012.
- [2] S. N. Ellis-Felege and J. P. Carroll, "Gamebirds and nest cameras: present and future," *Video surveillance of nesting birds. Studies in Avian Biology*, vol. 43, pp. 35–44, 2012.
- [3] K. Goehner, T. Desell, R. Eckroad, L. Mohsenian, P. Burr, N. Caswell, A. Andes, and Susan-Ellis-Felege, "A comparison of background subtraction algorithms for detecting avian nesting events in uncontrolled outdoor video," in *In the 11th IEEE International Conference on eScience*, Munich, Germany, August 2015.
- [4] T. Desell, R. Bergman, K. Goehner, R. Marsh, R. VanderClute, and S. Ellis-Felege, "Wildlife@ home: Combining crowd sourcing and volunteer computing to analyze avian nesting video," in *eScience (eScience), 2013 IEEE 9th International Conference on*. IEEE, 2013, pp. 107–115.
- [5] Y. LeCun and C. Cortes, "Mnist handwritten digit database," *AT&T Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist>, 2010.
- [6] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Computer Science Department, University of Toronto, Tech. Rep*, 2009.
- [7] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 11, pp. 1958–1970, 2008.
- [8] D. Weinland, R. Ronfard, and E. Boyer, "A survey of vision-based methods for action representation, segmentation and recognition," *Computer Vision and Image Understanding*, vol. 115, no. 2, pp. 224 – 241, 2011. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1077314210002171>
- [9] T. B. Moeslund, A. Hilton, and V. Kruger, "A survey of advances in vision-based human motion capture and analysis," *Computer Vision and Image Understanding*, vol. 104, no. 23, pp. 90 – 126, 2006. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1077314206001263>
- [10] H. Ishii, M. Ogura, S. Kurisu, A. Komura, A. Takanishi, N. Iida, and H. Kimura, "Development of autonomous experimental setup for behavior analysis of rats," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 29 2007-nov. 2 2007, pp. 4152 –4157.
- [11] W. Goncalves, J. Monteiro, J. de Andrade Silva, B. Machado, H. Pistori, and V. Odakura, "Multiple mice tracking using a combination of particle filter and k-means," in *Computer Graphics and Image Processing, 2007. SIBGRAPI 2007. XX Brazilian Symposium on*, oct. 2007, pp. 173 –178.
- [12] H. Pistori, V. V. V. A. Odakura, J. B. O. Monteiro, W. N. Goncalves, A. R. Roel, J. de Andrade Silva, and B. B. Machado, "Mice and larvae tracking using a particle filter with an auto-adjustable observation model," *Pattern Recognition Letters*, vol. 31, no. 4, pp. 337 – 346, 2010. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0167865509001330>
- [13] Y. Nie, I. Ishii, K. Yamamoto, K. Orito, and H. Matsuda, "Real-time scratching behavior quantification system for laboratory mice using high-speed vision," *Journal of Real-Time Image Processing*, vol. 4, pp. 181–190, 2009, 10.1007/s11554-009-0111-7. [Online]. Available: <http://dx.doi.org/10.1007/s11554-009-0111-7>
- [14] Y. Nie, I. Ishii, K. Yamamoto, T. Takaki, K. Orito, and H. Matsuda, "High-speed video analysis of laboratory rats behaviors in forced swim test," in *Automation Science and Engineering, 2008. CASE 2008. IEEE International Conference on*, aug. 2008, pp. 206 –211.
- [15] T. Mukhina, S. Bachurin, N. Lermontova, and N. Zefirov, "Versatile computerized system for tracking and analysis of water maze tests," *Behavior Research Methods*, vol. 33, pp. 371–380, 2001, 10.3758/BF03195391. [Online]. Available: <http://dx.doi.org/10.3758/BF03195391>
- [16] S. N. Fry, N. Rohrseitz, A. D. Straw, and M. H. Dickinson, "Trackfly: Virtual reality for a behavioral system analysis in free-flying fruit flies," *Journal of Neuroscience Methods*,

- vol. 171, no. 1, pp. 110 – 117, 2008. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0165027008001210>
- [17] H. Dankert, L. Wang, E. D. Hooper, D. J. Anderson, and P. Perona, “Automated monitoring and analysis of social behavior in drosophila,” *Nature Methods*, 2009.
- [18] K. Branson, A. Robie, J. Bender, P. Perona, and M. Dickinson, “High-throughput ethomics in large groups of drosophila,” *Nature Methods*, 2009.
- [19] D. Huang, K. Meyers, S. Henry, F. De la Torre, and C. Horn, “Non-rigid tracking of musk shrews in video for detection of emetic episodes,” in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, june 2011, pp. 17 –23.
- [20] D. Huang, K. Meyers, S. Henry, F. D. la Torre, and C. C. Horn, “Computerized detection and analysis of cancer chemotherapy-induced emesis in a small animal model, musk shrew,” *Journal of Neuroscience Methods*, vol. 197, no. 2, pp. 249 – 258, 2011. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0165027011001270>
- [21] M. M. Naeini, G. Dutton, K. Rothley, and G. Mori, “Action recognition of insects using spectral clustering,” in *In IAPR Conference on Machine Vision Applications*, 2007.
- [22] J. Shi and J. Malik, “Normalized cuts and image segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 888–905, 2000.
- [23] J. Carpenter, P. Clifford, and P. Fearnhead, “Improved particle filter for nonlinear problems,” *Radar, Sonar and Navigation, IEE Proceedings -*, vol. 146, no. 1, pp. 2 –7, feb 1999.
- [24] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, “Monte carlo localization for mobile robots,” in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2, 1999, pp. 1322 –1328 vol.2.
- [25] N. Gordon, D. Salmond, and A. Smith, “Novel approach to nonlinear/non-gaussian bayesian state estimation,” *Radar and Signal Processing, IEE Proceedings F*, vol. 140, no. 2, pp. 107 –113, apr 1993.
- [26] M. Isard and A. Blake, “Contour tracking by stochastic propagation of conditional density,” in *Proceedings of the 4th European Conference on Computer Vision-Volume 1 - Volume 1*, ser. ECCV ’96. London, UK, UK: Springer-Verlag, 1996, pp. 343–356. [Online]. Available: <http://dl.acm.org/citation.cfm?id=645309.648900>
- [27] Z. Khan, T. Balch, and F. Dellaert, “Mcmc-based particle filtering for tracking a variable number of interacting targets,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 11, pp. 1805 –1819, nov. 2005.
- [28] A. Veeraraghavan, R. Chellappa, and M. Srinivasan, “Shape-and-behavior encoded tracking of bee dances,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 3, pp. 463 –476, march 2008.
- [29] Z. Khan, T. Balch, and F. Dellaert, “A rao-blackwellized particle filter for eigentracking,” in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 2, june-2 July 2004, pp. II-980 – II-986 Vol.2.
- [30] A. Cangar, T. Leroy, M. Guarino, E. Vranken, R. Fallon, J. Lenehan, J. Mee, and D. Berckmans, “Automatic real-time monitoring of locomotion and posture behaviour of pregnant cows prior to calving using online image analysis,” *Computers and Electronics in Agriculture*, vol. 64, no. 1, pp. 53 – 60, 2008. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0168169908001373>
- [31] R. Tillett, C. Onyango, and J. Marchant, “Using model-based image processing to track animal movements,” *Computers and Electronics in Agriculture*, vol. 17, no. 2, pp. 249 – 261, 1997. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0168169996013087>
- [32] R. Farah, J. Langlois, and G. Bilodeau, “Rat: Robust animal tracking,” in *Robotic and Sensors Environments (ROSE), 2011 IEEE International Symposium on*, sept. 2011, pp. 65 –70.
- [33] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, “Behavior recognition via sparse spatio-temporal features,” in *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, oct. 2005, pp. 65 – 72.
- [34] S. Belongie, K. Branson, P. Dollár, and V. Rabaud, “Monitoring animal behavior in the smart vivarium,” in *Measuring Behavior*, Wageningen, NL, 2005, pp. 70–72.
- [35] H. Jhuang, E. Garrote, N. Edelman, T. Poggio, A. Steele, and T. Serre, “Trainable, vision-based automated home cage behavioral phenotyping,” in *Proceedings of the 7th International Conference on Methods and Techniques in Behavioral Research*, ser. MB ’10. New York, NY, USA: ACM, 2010, pp. 33:1–33:4. [Online]. Available: <http://doi.acm.org/10.1145/1931344.1931377>
- [36] ———, “Vision-based automated recognition of mice home-cage behaviors.” in *Workshop: Visual Observation and Analysis of Animal and Insect Behavior in conjunction with International Conference on Pattern Recognition (ICPR)*, 2010.
- [37] H. Jhuang, E. Garrote, X. Yu, V. Khilnani, T. Poggio, A. D. Steele, and T. Serre, “Automated home-cage behavioural phenotyping of mice.” *Nature communications*, vol. 1, no. 6, p. 68, 2010. [Online]. Available: <http://www.nature.com/doifinder/10.1038/ncomms1064>
- [38] B. G. Weinstein, “Motionmeerkat: integrating motion video detection and ecological monitoring.” *Methods in Ecology and Evolution*, 2014.
- [39] P. W. Power and J. A. Schoonees, “Understanding background mixture models for foreground segmentation,” in *Proceedings image and vision computing New Zealand*, vol. 2002, 2002, pp. 10–11.
- [40] A. M. McIvor, “Background subtraction techniques,” *Proc. of Image and Vision Computing*, vol. 4, pp. 3099–3104, 2000.
- [41] M. Piccardi, “Background subtraction techniques: a review,” in *Systems, man and cybernetics, 2004 IEEE international conference on*, vol. 4. IEEE, 2004, pp. 3099–3104.
- [42] D. Tweed and A. Calway, “Tracking many objects using subordinated condensation,” in *Proceedings of the British Machine Vision Conference*, P. Rosin and D. Marshall, Eds. BMVA Press, October 2002, pp. 283–292. [Online]. Available: <http://www.cs.bris.ac.uk/Publications/Papers/1000674.pdf>
- [43] M. Betke, D. Hirsh, A. Bagchi, N. Hristov, N. Makris, and T. Kunz, “Tracking large variable numbers of objects in clutter,” in *Computer Vision and Pattern Recognition, 2007. CVPR ’07. IEEE Conference on*, june 2007, pp. 1 –8.
- [44] A. Ernst and C. Kublbeck, “Fast face detection and species classification of african great apes,” in *Advanced Video and Signal-Based Surveillance (AVSS), 2011 8th IEEE International Conference on*, 30 2011-sept. 2 2011, pp. 279 –284.
- [45] J. Wawerla, S. Marshall, G. Mori, K. Rothley, and P. Sabzmeydani, “Bearcam: automated wildlife monitoring at the arctic circle,” *Mach. Vision Appl.*, vol. 20, no. 5, pp. 303–317, Jun. 2009. [Online]. Available: <http://dx.doi.org/10.1007/s00138-008-0128-0>
- [46] P. Sabzmeydani and G. Mori, “Detecting pedestrians by learning shapelet features,” in *Computer Vision and Pattern Recognition, 2007. CVPR ’07. IEEE Conference on*, june 2007, pp. 1 –8.
- [47] P. Viola and M. Jones, “Robust real-time object detection,” in *International Journal of Computer Vision*, 2001.
- [48] J. Stone, D. Gohara, and G. Shi, “Opencl: A parallel programming standard for heterogeneous computing systems,” *Computing in science & engineering*, vol. 12, no. 3, p. 66, 2010.
- [49] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” in *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014, pp. 675–678.
- [50] A. Vedaldi and K. Lenc, “Matconvnet: Convolutional neural networks for matlab,” in *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference*. ACM, 2015, pp. 689–692.
- [51] D. P. Anderson, E. Korpela, and R. Walton, “High-performance task distribution for volunteer computing.” in *e-Science*. IEEE Computer Society, 2005, pp. 196–203.
- [52] A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier nonlinearity improves neural network acoustic models,” in *Proc. ICML*, vol. 30, 2013, p. 1.
- [53] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, “Efficient backprop,” in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.
- [54] Y. Nesterov, “A method of solving a convex programming problem with convergence rate $O(1/k^2)$,” in *Soviet Mathematics Doklady*, vol. 27, no. 2, 1983, pp. 372–376.

Using LSTM Recurrent Neural Networks to Predict Excess Vibration Events in Aircraft Engines

AbdElRahman ElSaid*, Brandon Wild†, James Higgins†, Travis Desell*

Department of Computer Science*

Department of Aviation†

University of North Dakota

Grand Forks, North Dakota 58202

Email: abdelrahman.elsaid@und.edu, bwild@aero.und.edu, jhiggins@aero.und.edu, tdesell@cs.und.edu

Abstract—This paper examines building viable Recurrent Neural Networks (RNN) using Long Short Term Memory (LSTM) neurons to predict aircraft engine vibrations. The model is trained on a large database of flight data records obtained from an airline containing flights that suffered from excessive vibration. RNNs can provide a more generalizable and robust method for prediction over analytical calculations of engine vibration, as analytical calculations must be solved iteratively based on specific empirical engine parameters, and this database contains multiple types of engines. Further, LSTM RNNs provide a “memory” of the contribution of previous time series data which can further improve predictions of future vibration values. LSTM RNNs were used over traditional RNNs, as those suffer from vanishing/exploding gradients when trained with back propagation. The study managed to predict vibration values for 5, 10 and 20 seconds in the future, with 3.3%, 5.51% and 10.19% mean absolute error, respectively. These neural networks provide a promising means for the future development of warning systems so that suitable actions can be taken before the occurrence of excess vibration to avoid unfavorable situations during flight.

I. INTRODUCTION

Aircraft Engine vibration is a critical aspect of the aviation industry, and accurate predictions of excessive engine vibration have the potential to save time, effort, money as well as human lives in the aviation industry. An aircraft engine, as turbomachinery, should normally vibrate as it has many dynamic parts. However, it is not supposed to exceed resonance limits so not to destroy the Engine [1].

Reference [1] is discussing vibrations generated from engine blades’ fluttering. Engine blades are the engine rotating components that have the largest dimensions among other components. While their rotation at high speeds, they will withstand high centrifugal forces that would logically give the highest contribution to engine vibrations.

Engine vibrations are not that simple to calculate or predict analytically because of the fact that various parameters contribute to their occurrence. This fact is always a problem for aviation performance monitors, especially that engines vary in design, size, operation conditions, service life span, the aircraft they are mounted on, and many other parameters. Most of these parameters’ contributions can be translated in some key parameters measured and recorded on the flight data recorder. Nonetheless, vibrations are likely to be a result of a mixture

of these contributions, making it very hard to predict the real cause behind the excess in vibrations.

This paper presents a means to make these predictions viable in the aviation industry within a reasonable time window. The problem is approached using LSTM RNNs, which have seen widespread recent use with strong results in image [2], speech [3] and, language prediction [4]. LSTM RNNs were chosen for this work in particular due to their generalizability and predictive power due to having a memory for the contribution of the previous time series data to predict the future values of vibration. This study provides another dimension for the use of this promising type of recurrent neural network.

II. RELATED WORK

A. Aircraft Engine Vibration

According to Reference [1]: “*The most common types of vibration problems that concern the designer of jet engines include (a) resonant vibration occurring at an integral order, i.e. multiple of rotation speed, and (b) flutter, an aeroelastic instability occurring generally as a non-integral order vibration, having the potential to escalate, unless checked by any means available to the operator, into larger and larger stresses resulting in serious damage to the machine. The associated failures of engine blades are referred to as high cycle fatigue failures*”. The means available to the operator in practical aviation operations are mainly: *i*) maintenance engine checks scheduled in maintenance programs based on engine reliability observations, and *ii*) engine vibration monitoring for forecasting the excess vibration occurrence based on statistical and analytical methods which consider empirical factors of safety. Some effort had been done using neural networks to classify engine abnormalities without doing analytical computation, *e.g.*, Alexandre Nairac *et al.* [5] worked on this aspect to detect abnormalities in engine vibrations based on recorded data.

David A. Clifton *et al.* [6] presented work for predicting abnormalities in engine vibration based on statistical analysis of vibration signatures. The paper presents two modes of prediction. One is ground-based (off-line), where prediction is done by run-by-run analysis to predict abnormalities based on previous engine runs. The success in this approach was

predicting abnormalities two runs ahead. The other mode is an flight based-mode (online) in which detection is done either by sending reduced data to the ground-base or onboard the aircraft. The paper mentions that they had future 2.5 hours successful prediction. However, this prediction is done after half an hour of flight data collection, which might be a critical time as well, as excess vibration may occur during this data collection time. The paper did not mention how much data was required to have a sound prediction.

B. LSTM RNN

LSTM RNN was first introduced by S. Hochreiter & J. Schmidhuber [7]. The paper introduced a solution for this problem: "*Learning to store information over extended period of time intervals via recurrent backpropagation takes a very long time, mostly due to insufficient, decaying error back flow*". It was a solution for the exploding/vanishing gradients in backpropagation to modify the weights of the network. This study paved the way for many interesting projects. LSTM RNN's have been used with strong performance in image recognition [2], audio visual emotion recognition [3], music composition [8] and other areas.

III. EXPERIMENTAL DATA

The data used consists of 76 different parameters recorded on the aircraft Flight Data Recorder (FDR) as well as the vibration parameter. A subset of these parameters were chosen based on the likelihood of their contribution to the vibration based on aerodynamics/turbo-machinery background. Some parameters, such as Inlet Guide Vans Configuration, Fuel Flow, Spoilers Configuration (this was preliminary considered because of the special position of the engine mount), High Pressure Valve Configuration and Static Air Temperature were excluded because it was found that they generated noise more than positively contributing in the vibration prediction.

The finally chosen parameters were:

- 1) Altitude
- 2) Angle of Attack
- 3) Bleed Pressure
- 4) Turbine Inlet Temperature
- 5) Mach Number
- 6) Primary Rotor/Shft Rotation Speed
- 7) Secondary Rotor/Shft Rotation Speed
- 8) Engine Oil pressure
- 9) Engine Oil Quantity
- 10) Engine Oil Temperature
- 11) Aircraft Roll
- 12) Total Air Temperature
- 13) Wind Direction
- 14) Wind Speed
- 15) Engine Vibration

IV. METHODOLOGY

Three LSTM RNN architectures were designed to predict engine vibration 5 seconds, 10 seconds, and 20 seconds in the future. Each of the 15 selected FDR parameters is represented

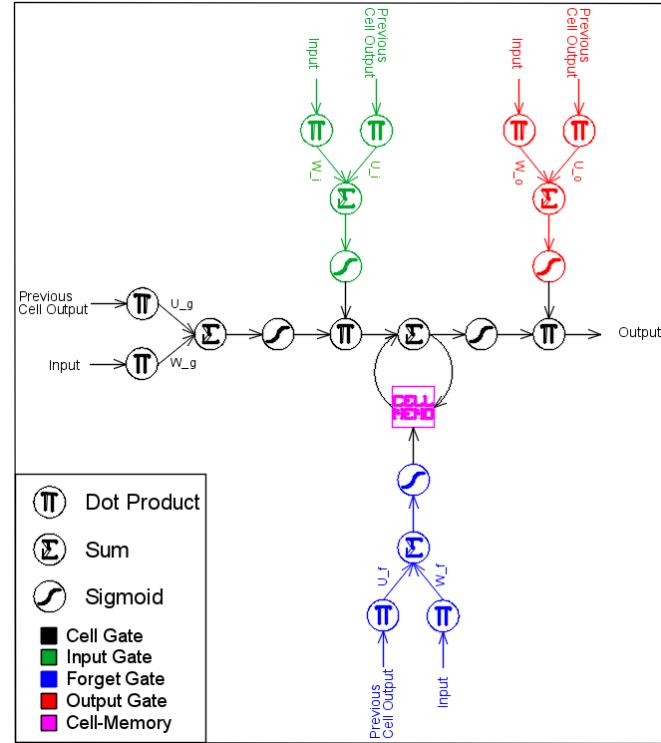


Fig. 1. LSTM cell design

by a node in the inputs of the neural network and an additional node is used for a bias. Each neural network in the three designs consists of LSTM cells that receive both an initial input and the output of the previous cell, as inputs. Each cell has three gates to control the flow of information through the cell and accordingly, the output of the cell. Each cell also has a cell-memory which is the core of the LSTM RNN design. The cell-memory allows the flow of information from the previous states into current predictions.

The gates that control the flow are shown in Figure 1. They are: *i*) the *input gate*, which controls how much information will flow from the inputs of the cell, *ii*) the *forget gate*, which controls how much information will flow from the cell-memory, and *iii*) the *output gate*, which controls how much information will flow out of the cell. This design allows the network to learn not only about the target values, but also about how to tune its controls to reach the target values.

All the utilized architectures have a common LSTM cell design shown in Figure 1. However, there are two variations of this common design used in the utilized architectures, shown in Figures 2 and 3, with the difference being the number of inputs from the previous. Cells that take initial inputs from more input nodes are denoted by '*M1*' cells. As input nodes are needed to be reduced through the neural network, the design of the cell will be different and it is denoted by '*M2*' cells.

The equations used in the forward propagation through the neural network are:

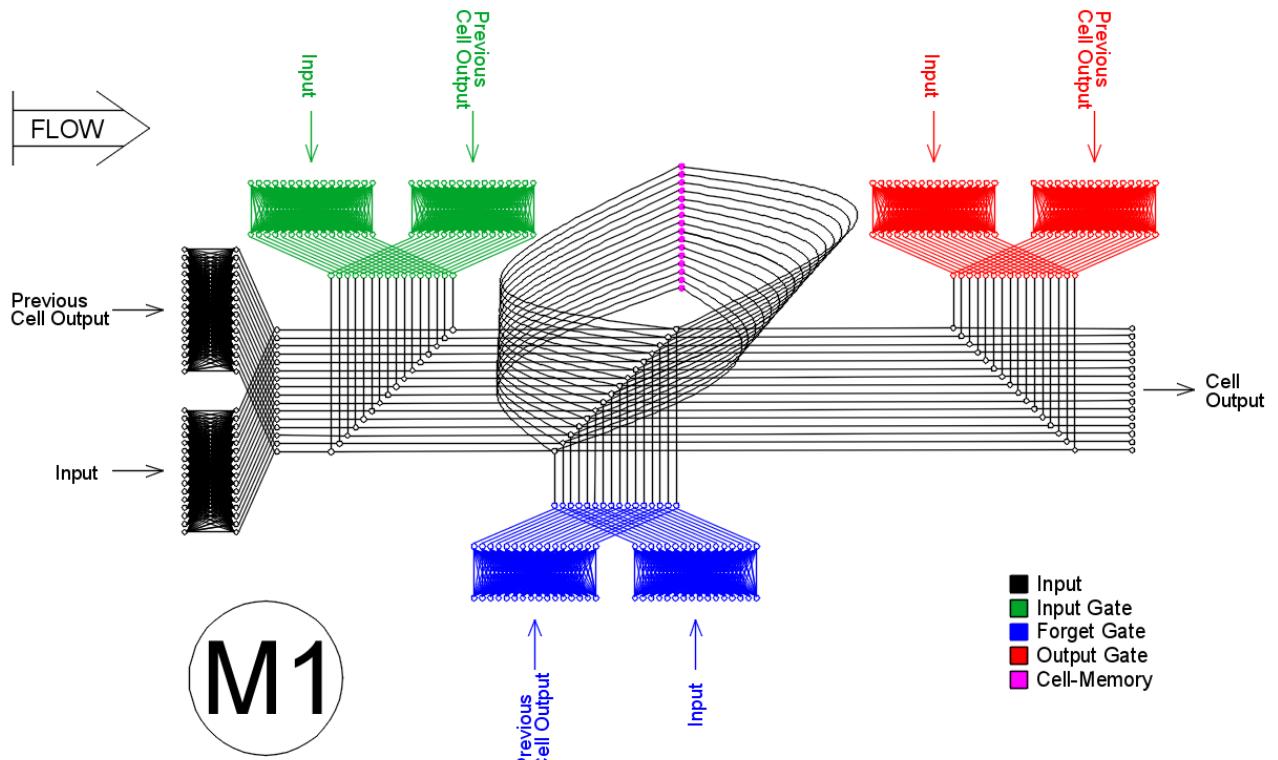


Fig. 2. Level 1 LSTM cell design

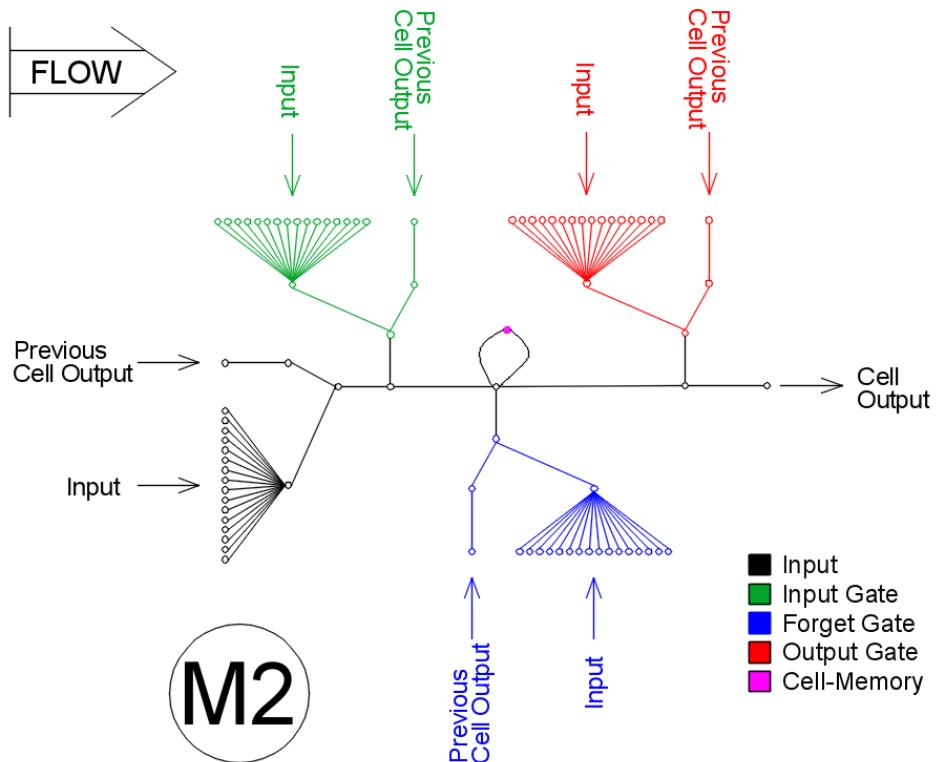


Fig. 3. Level 2 LSTM cell design

TABLE I
ARCHITECTURES WEIGHTS-MATRICES DIMENSIONS

$$i_t = \text{Sigmoid}(w_i \bullet x_t + u_i \bullet a_{t-1} + \text{bias}_i) \quad (1)$$

$$f_t = \text{Sigmoid}(w_f \bullet x_t + u_f \bullet a_{t-1} + \text{bias}_f) \quad (2)$$

$$o_t = \text{Sigmoid}(w_o \bullet x_t + u_o \bullet a_{t-1} + \text{bias}_o) \quad (3)$$

$$g_t = \text{Sigmoid}(w_g \bullet x_t + u_g \bullet a_{t-1} + \text{bias}_g) \quad (4)$$

$$c_t = f_t \bullet c_{t-1} + i_t \bullet g_t \quad (5)$$

$$a_t = o_t \bullet \text{Sigmoid}(c_t) \quad (6)$$

where (see Figure 1):

i_t : input-gate output

f_t : forget-gate output

o_t : output-gate output

g_t : input's sigmoid

c_t : cell-memory output

w_i : weights associated with input and input-gate

u_i : weights associated with previous output and input-

gate

w_f : weights associated with input and forget-gate

u_f : weights associated with previous output and forget-

gate

w_o : weights associated with input and output-gate

u_o : weights associated with previous output and the output-gate

w_g : weights associated with the cell input

u_g : weights associated with previous output and the cell input

V. LSTM RNN ARCHITECTURES

The three architectures are as follows, with the dimensions of the weights of these architectures shown in Table I and the total number of weights shown in Table II:

A. Architecture I

As shown in Figure 4, this architecture takes inputs from ten time series (the current time instant and the past nine). It feeds the second level of the neural network with its output. The output of the first level of the neural network is considered the first hidden layer. The second level of the neural network then reduces the number of nodes fed to it from 16 nodes (15 input nodes + bias) per cell to only one node per cell. The output of the second level of the neural network is considered the second hidden layer. Finally, the output of the second level of the neural network would be only 10 nodes, a node from each cell. These nodes are fed to a final neuron in the third level to compute the output of the whole network.

Architecture I								
	w_i	u_i	w_f	u_f	w_o	u_o	w_g	u_g
Level 1	16×16	16×16	16×16	16×16	16×16	16×16	16×16	16×16
Level 2	16×1	1×1	16×1	1×1	16×1	1×1	16×1	1×1
Level 3							16×1	
Architecture II								
	w_i	u_i	w_f	u_f	w_o	u_o	w_g	u_g
Level 1	16×16	16×16	16×16	16×16	16×16	16×16	16×16	16×16
Level 2	16×1	1×1	16×1	1×1	16×1	1×1	16×1	1×1
Level 4							16×1	
Architecture III								
	w_i	u_i	w_f	u_f	w_o	u_o	w_g	u_g
Level 1	16×16	16×16	16×16	16×16	16×16	16×16	16×16	16×16
Level 2	16×16	16×16	16×16	16×16	16×16	16×16	16×16	16×16
Level 3	16×1	1×1	16×1	1×1	16×1	1×1	16×1	1×1
Level 4							16×1	

TABLE II
ARCHITECTURES WEIGHTS MATRICES' TOTAL ELEMENTS

Art I	Art II	Art III
21,170	21,160	83,290

B. Architecture II

As shown in Figure 5, this architecture is almost the same as the previous one except that it does not have the third level. Instead, the output of the second level is averaged to compute the output of the whole network.

C. Architecture III

Figure 6 presents a deeper neural network architecture. In this design, the neural network takes inputs from twenty time series (the current time instant and the past nineteen). It feeds the second level of the neural network with its output. Second level does the same procedure as first level giving a chance for more abstract decision making. The output of the second level of the neural network is considered the first hidden layer and the output of the second level is considered the second hidden layer. The third level of the neural network then reduces the number of nodes fed to it from 16 nodes (15 input nodes + bias) per cell to only one node per cell. The output of the third level of the neural network is considered the third hidden layer. Finally, the output of the third level of the neural network is twenty nodes, a node from each cell. These nodes are fed to a final neuron in the fourth level to compute the output of the whole network.

D. Forward Propagation

The following is a general description for the forward propagation path. This example uses Architecture I as an example but similar steps are taken in the other architectures with minor changes apparent in their diagrams. With Figure 4

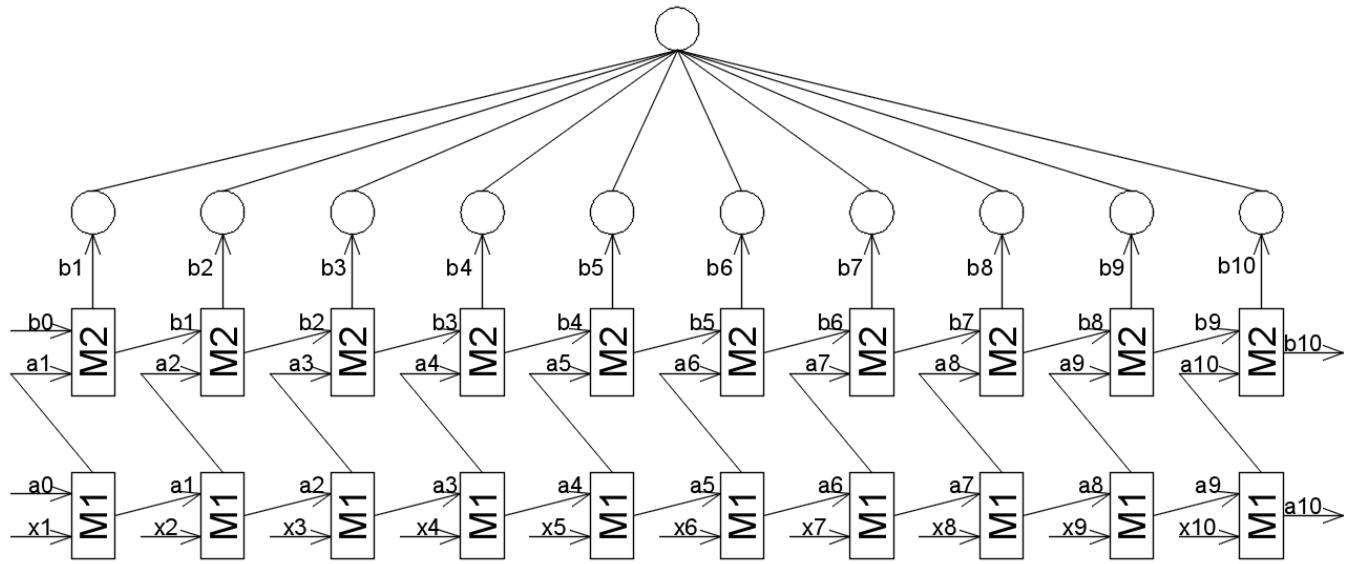


Fig. 4. Architecture I

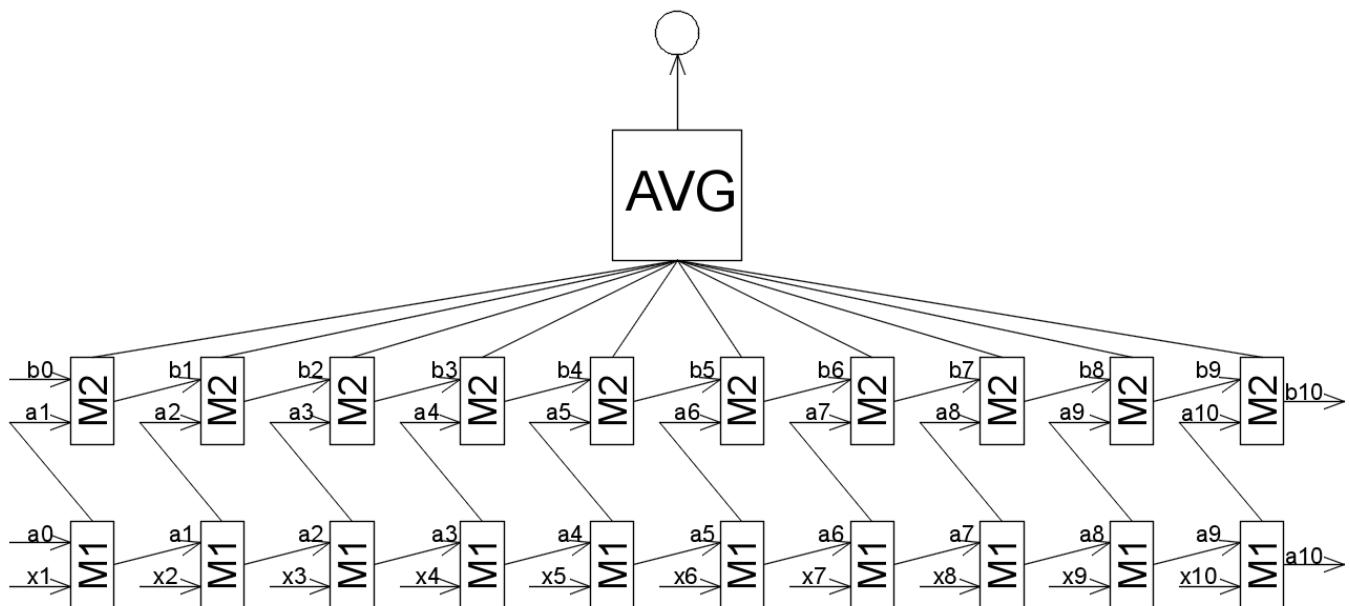


Fig. 5. Architecture II

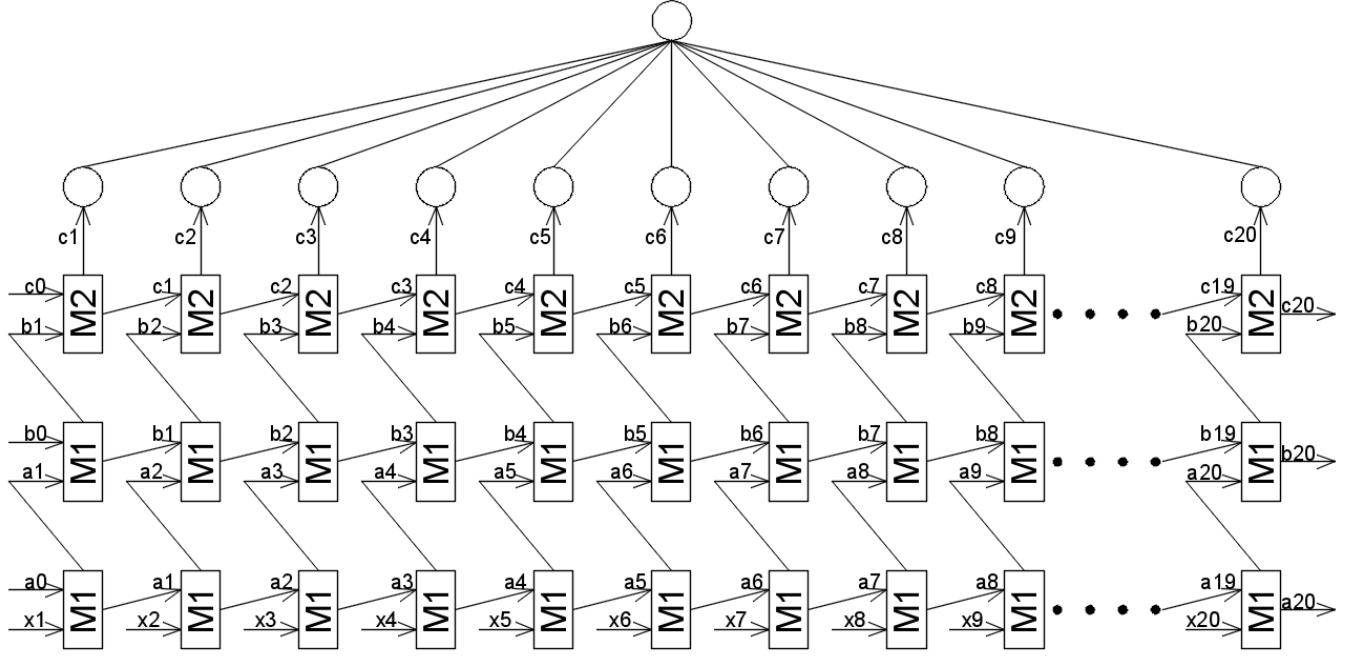


Fig. 6. Architecture III

presenting an overview of the structure of the whole network and considering Figure 2 as an overview of the structure of the cells in *Level 1* and in *Level 2* – the input at each iteration consists of 10 seconds of time series data of the 15 input parameters and 1 bias (*Input* in Figure 2) in one vector (x_t in Figure 4) and the output of the previous cell (*Previous Cell Output* in Figure 2) in another vector (a_{t-1} in Figure 4). Each second of time series input is fed to the corresponding cell (*i.e.*, the first seconds' 15 parameters and 1 bias are fed to first cell, the second seconds' 15 parameters and 1 bias are fed to second cell, ...) into the *cell gate* (shown in black color), *input gate* (shown in green color), *forget gate* (shown in blue color) and the *output gate* (shown in red color). If the gates (*input gate*, *forget gate* and, *output gate*) are seen as valves that control how much of the data flow through it, the outputs of these gates (i_t , f_t and, o_t) are considered as how much these valves are opened or closed.

First, at the *cell gate*, x_t is dot multiplied by its weights matrix w_g and a_{t-1} is dot multiplied by its weights matrix u_g . The output vectors are summed and an activation function is applied to it as in Equation 4. The output is called g_t .

Second, at the *input gate*, x_t is dot multiplied by its weights matrix w_i and a_{t-1} is dot multiplied by its weights matrix u_i . The output vectors are summed and an activation function is applied to it as in Equation 1. The output is called i_t .

Third, at the *forget gate*, x_t is dot multiplied by its weights matrix w_f and a_{t-1} is dot multiplied by its weights matrix

u_f . The output vectors are summed and an activation function is applied to it as in Equation 2. It controls how much of the *cell memory* Figure 4 (saved from previous time step) should pass. The output is called f_t .

Fourth, at the *output gate*, x_t is dot multiplied by its weights matrix w_o and a_{t-1} is dot multiplied by its weights matrix u_o . The output vectors are summed and an activation function is applied to it as in Equation 3. The output is called o_t .

Fifth, the contribution of the cell input *Input* g_t and *cell memory* c_{t-1} is decided in Equation 5 by dot multiplying them by f_t and i_t respectively. The output of this step is the new *cell memory* c_t .

Sixth, cell output is also regulated by the output gate (valve). This is done by applying the sigmoid function to the *cell memory* c_t and dot multiplying it by o_t as shown in Equation 6. The output of this step is the final output of the cell at the current time step a_t . a_t is fed to the next cell in the same level and also fed to the cell in the above level as an *Input* a_t .

The same procedure is applied at *Level 2* but with different weight vectors and different dimensions. Weights at *Level 2* have smaller dimensions to reduce their input dimensions from vectors with 16 dimensions to vectors with one dimension. The output from *Level 2* a one dimensional vector from each cell of the 10 cells in *Level 2*. These vectors are fed as one 10 dimensional vector to a simple neuron shown in Figure 4 at *Level 3* to be dot multiplied by a weight vector to reduce the vector to a single scalar value: the final output of the network

TABLE III
TRAINING RESULTS

	Error at 5 seconds	Error at 10 seconds	Error at 20 seconds
Architecture I	0.000398	0.000972	0.001843
Architecture II	0.001516	0.001962	0.002870
Architecture III	0.000409	0.000979	0.001717

at the time step.

VI. IMPLEMENTATION

A. Programming Langauge

Python's Theano Library [9] was used to implement the neural networks. It has four major advantages: *i*) it will compile the most, if not all, of functions coded using it to C and CUDA giving fast performance, *ii*) it will perform the weights updates for back propagation with minimal overhead, *iii*) Theano can compute the gradients of the error (cost function output) with respect to the weights saving significant effort and time needed to manually derive the gradients, coding and debugging them, and finally, *iv*) it can utilize GPU's for further increased performance.

B. Data Processing

The flight data parameters used were normalized between 0 and 1. The sigmoid function is used as an activation function over all the gates and inputs/outputs. The ArcTan activation function was tested on the data, however it gave distorted results and sigmoid function provided significantly better performance.

C. Machine Specifications

Each of the examined architectures runs on a hyperthreaded 3.5 GHz core and is considered capable of real-time processing. Results were collected using a Mac Pro with 12 logical cores, with each different architecture being trained for 575 epochs. Run times for training are shown in Table IV. Some unexpected variance might be realized in these run-times, due to CPU interruptions which may have occurred over the course of the experiments. In general, the first two architectures took similar amounts of time (approximately 8.5-9 hours) for each time prediction (5, 10 and 20 seconds), and the third took a bit more than twice as long, at approximately 20 hours for each time prediction.

VII. RESULTS

The neural networks were run against flights that suffered from the excessive vibration in a training phase. They were then run against different set of flights, which also suffered from the same problem, in a testing phase. There were 28 flights in the training set, with a total of 41,431 seconds of data. There were 57 flights in the testing set, with a total of 38,126 seconds of data. The networks were allowed to train for 575 epochs to learn and for the cost function output curve to flatten.

TABLE IV
RUN TIME (HOURS)

	05	10	20
Architecture I	9	8.98	8.85
Architecture II	8.44	8.41	8.4
Architecture III	21.6	19.7	18.5

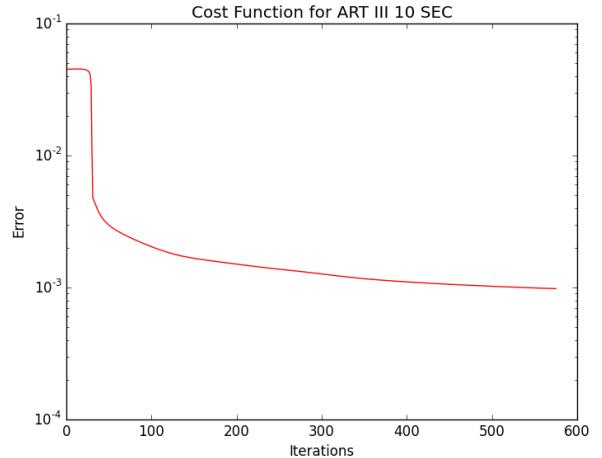


Fig. 7. Cost function plot for ART III predicting vibration in 10 future sec

A. Cost Function

Mean squared error was used to train the neural networks as it provides a smoother optimization surface for backpropagation. A sample of the cost function output can be seen in Figure 7. The Figure is a logarithmic plot for architecture III, for predicting vibrations 10 seconds in the future.

B. Architecture Results

Mean Squared Error (MSE) (shown in Equation 7) was used as an error measure to train the three architectures, which resulted in values shown in Table V. Mean Absolute Error (MAE) (shown in Equation 8) is used as a final measure of accuracy for the three architectures, with results shown in Table VI. As the parameters were normalized between 0 and 1, the MAE is also the percentage error.

$$Error = \frac{0.5 \times \sum(Actual Vib - Predicted Vib)^2}{Testing Seconds} \quad (7)$$

$$Error = \frac{\sum[ABS(Actual Vib - Predicted Vib)]}{Testing Seconds} \quad (8)$$

Figures 9, Figures 10, and Figures 11 present the predictions for all the test flights condensed on the same plot. Time shown on the x-axis is the total time for all the test flights. Each flight ends when the vibration reaches the max critical value (normalized to 1) and then the next flight in the test set begins. Figure 8 provides an uncompressed example of Architecture

TABLE V
TESTING RESULTS

	Error at 5 seconds	Error at 10 seconds	Error at 20 seconds
Architecture I	0.001165	0.002926	0.010427
Architecture II	0.009708	0.009056	0.012560
Architecture III	0.002386	0.004780	0.041417

TABLE VI
NEW TESTING RESULTS

	Error at 5 seconds	Error at 10 seconds	Error at 20 seconds
Architecture I	0.033048	0.055124	0.101991
Architecture II	0.097588	0.096054	0.112320
Architecture III	0.048056	0.070360	0.202609

I predicting vibration 10 seconds in the future over a single flight from the testing data.

1) *Results of Architecture I:* The results of this architecture, shown in Table V, came out to be the best results regarding the overall accuracy of the vibration prediction. While there is more misalignment between the actual and calculated vibration values as predictions are made further in the future, as shown in Figure 9, this is to be expected. Also, it can be seen that the prediction of higher peaks is more accurate than the lower peaks prediction as if the neural network is tending to learn more about the max critical vibration value, which is favorable for this project.

2) *Results of Architecture II:* The results of this architecture in Table V came out to be the least successful in vibration prediction. While it managed to predict much of the vibration, its performance was weak at the peaks (either low or high) compared to the other architectures, as shown in Figure 10. It is also worth mentioning that somehow the lower peaks were better at some positions on the curve of this architecture, compared to the other architectures.

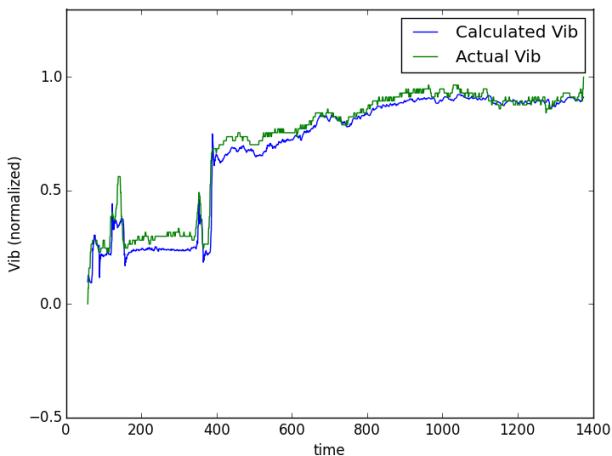
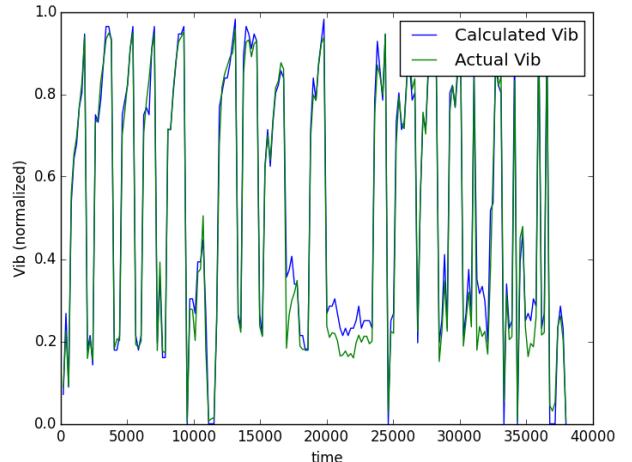
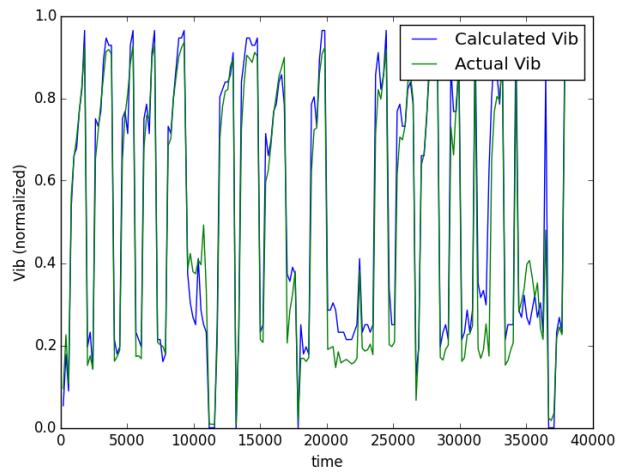


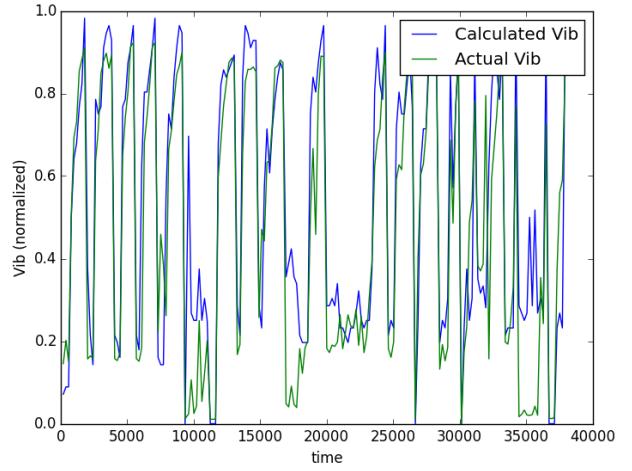
Fig. 8. Architecture I predicting vibration 10 seconds in the future.



(a) ART I Results Plot @ 05 SEC

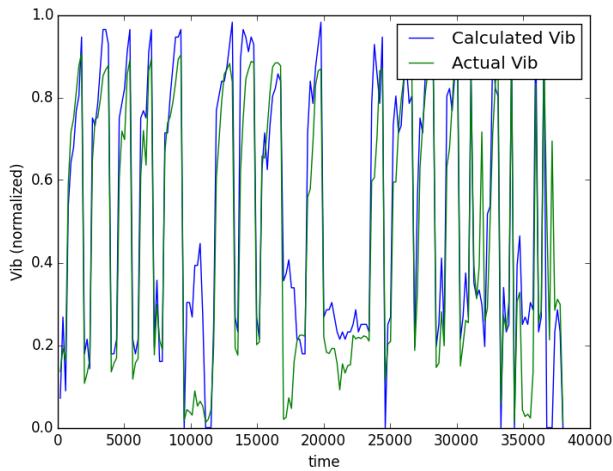


(b) ART I Results Plot @ 10 SEC

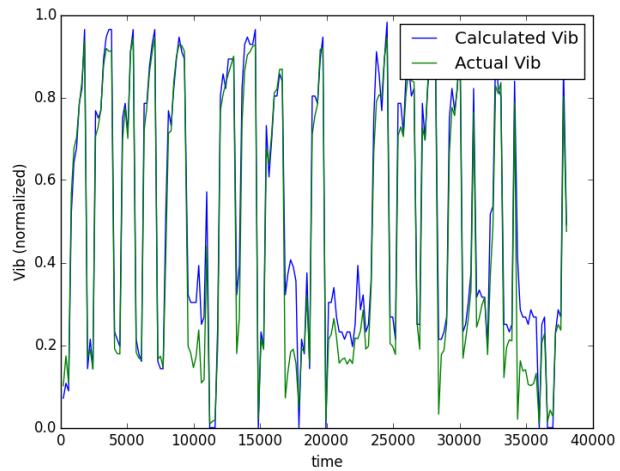


(c) ART I Results Plot @ 20 SEC

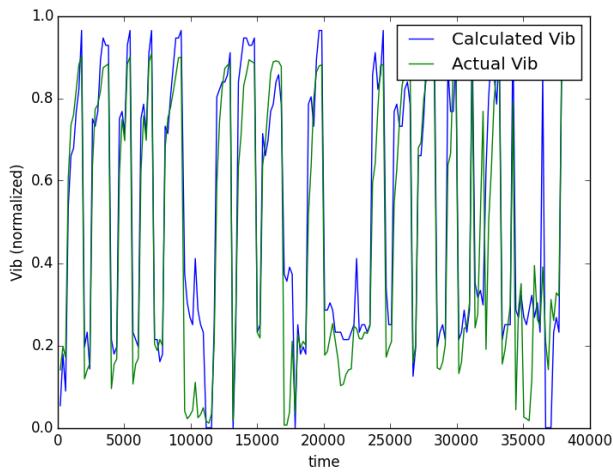
Fig. 9. Plotted results for Architecture I for the three scenarios.



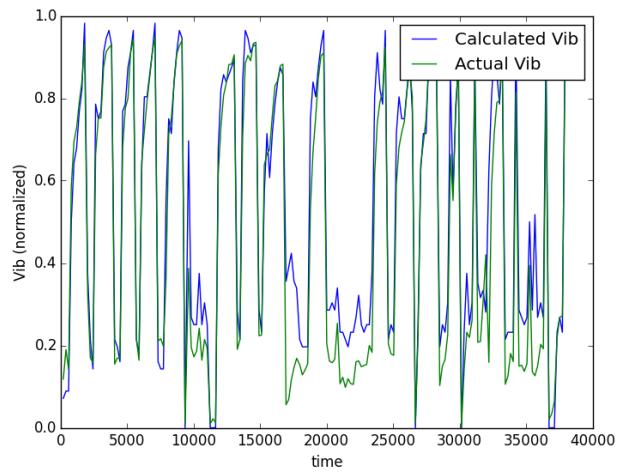
(a) ART II Results Plot @ 05 SEC



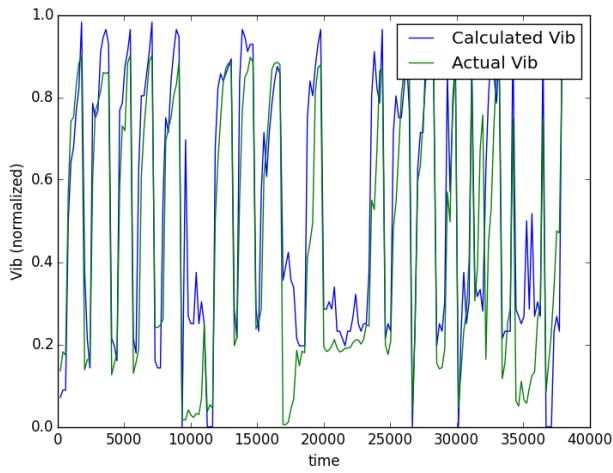
(a) ART III Results Plot @ 05 SEC



(b) ART II Results Plot @ 10 SEC

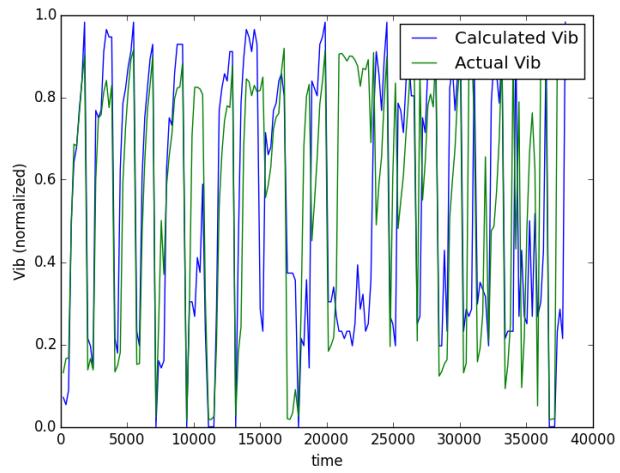


(b) ART III Results Plot @ 10 SEC



(c) ART II Results Plot @ 20 SEC

Fig. 10. Plotted results for Architecture II for the three scenarios.



(c) ART III Results Plot @ 20 SEC

Fig. 11. Plotted results for Architecture I for the three scenarios.

3) Results of Architecture III: Although it was the most computationally expensive and had a chance for deeper learning, its results were not as good as expected, as shown in Figure 11. The results of this architecture in Table V show that the prediction accuracy for this architecture was less than the more simple Architecture I. As this came counter to the predictions for deeper learning, this opens door for investigating about the deeper learning for this problem; this LSTM RNN was one layers deeper and also had 20 seconds memory from the past which was not available for the other two LSTM RNN's used. It is also realized that the overall error in Table V for the prediction at 20 future seconds came relatively high. Looking at Figure 11c between time 10,000-15,0000, 20,000-25,000 and 35,000-40,000, it can be seen that the calculated curve got very much higher than the actual vibration curve. This strange behaviour is unique as it can be seen that the calculated vibration would rarely exceed the actual vibration for all the curves plotted for all the architectures at all scenarios, and it would be for relatively small value if occurred. This network could potentially gain further improvement if trained for more epochs over the other simpler architectures.

VIII. CONCLUSIONS AND FUTURE WORK

This paper presents early work for utilizing long short term memory (LSTM) recurrent neural networks (RNNs) of different types to predict engine vibrations and other critical aviation parameters. The results obtained from this study are very encouraging, given the accuracy of the predictions rather far in the future – 3.3% error for 5 second predictions, 5.51% error for 10 second predictions, and 10.19% error for 20 second predictions. This work opens up many avenues for future work, such as fine tuning the neural network designs and their hyper parameters, changing the design of the layers and/or combine different types of RNNs to further refine the results. Selecting flight parameters also had a great influence on the results. This work could be extended by further investigating the flight parameters and their contributions to the prediction process. This could be achieved by either statistical means or going deeper in the analytical and empirical theories and equations

to provide a deeper understanding of the relations between parameters, and thus, more precise future predictions.

Also the use of accelerator cards such as GPUs could be used in this research to further improve training times, and allow the neural networks to be trained longer (which could potentially improve the performance of Architecture III). This can save time if well implemented; as data weights vectors and matrices for all gates (inputs, input gates, forget gates, and output gates) can be grouped together in one matrix/vector saved in one global memory variable to be transferred as one group to the GPU. This would reduce the penalty of data transfer between the CPU and GPU. Similar measures can be followed for processing the data for the several files instead of doing one data file (FDR reading) at a time. Subsequently, processing the data for several future vibration predictions (ex. at 5 sec, 10 sec, 20 sec, ...) could be performed together at the same time, reducing data transfer between CPU and GPU.

Overall, this work provides promising initial work in engine vibration prediction that could be integrated into future warning systems so that pilots can act to prevent excessive vibration events before unfavorable situations happen during flight.

REFERENCES

- [1] A. V. Srinivasan, “FLUTTER AND RESONANT VIBRATION CHARACTERISTICS OF ENGINE BLADES,” 1997. [Online]. Available: <http://www.energy.kth.se/compedu/webcompedu/WebHelp>
- [2] Donahue, Jeffrey and Anne Hendricks, Lisa and Guadarrama, Sergio and Rohrbach, Marcus and Venugopalan, Subhashini and Saenko, Kate and Darrell, Trevor, “Long-term recurrent convolutional networks for visual recognition and description,” June 2015.
- [3] Linlin Chao et al, “Audio visual emotion recognition with temporal alignment and perception attention,” Mar 2016.
- [4] Ilya Sutskever, Oriol Vinyals, Quoc V. Le, “Sequence to Sequence Learning with Neural Networks,” Dec 2014.
- [5] A. NAIRAC et al, “A System for the Analysis of Jet Engine Vibration Data,” 1999.
- [6] David A. Clifton et al, “A Framework for Novelty Detection in Jet Engine Vibration Data,” 2007.
- [7] S. Hochreiter & J. Schmidhuber, “Long Short Term Memory.”
- [8] D. Eck & J. Schmidhuber, “A First Look at Music Composition using LSTM Recurrent Neural Network.”
- [9] Theano Development Team, “Theano: A Python framework for fast computation of mathematical expressions,” *arXiv e-prints*, vol. abs/1605.02688, May 2016. [Online]. Available: <http://arxiv.org/abs/1605.02688>

SimP: Secure Interoperable Multi-Granular Provenance Framework

Amani Abu Jabal, Elisa Bertino

Dept. of Computer Science, Purdue University, West Lafayette, USA

{aabujaba,bertino}@purdue.edu

Abstract— We propose a provenance framework which includes an expressive provenance model able to represent the provenance of any data object captured with various granularities. The model is represented according to relational and graph specifications. The framework is interoperable with two standard provenance models: OPM and PROV. In addition, the framework captures access control policies for data objects and secures the provenance storage itself. We have integrated our framework with CRIS - a real world system for managing scientific data.

Keywords—Provenance Model, Provenance Framework, Relational, Graph

I. INTRODUCTION

Data provenance, one kind of metadata, pertains to the derivation history of a data object starting from its original sources [3]. The term data object refers to data in any format (e.g., files, database records, or workflow templates). One example of provenance model for scientific applications is Chimera [1] that documents workflows generating data. Data provenance is critical for many purposes including: assessing data quality based on its ancestral data and derivations, detecting sources of errors and anomalies, providing an audit trail for regulatory purposes, and assessing data trustworthiness.

Building a comprehensive provenance infrastructure involves addressing the following requirements:

1. **multi-granular provenance model**: the provenance infrastructure should provide a rich provenance model capable to represent the provenance of data objects with different granularities (e.g., file, database record, data in a workflow);
2. **provenance queries**: the infrastructure should support queries for inspecting various aspects of the provenance;
3. **security**: the framework should be able to capture the access control policies, according to which users had been granted permissions to the data, at the time of data access; the infrastructure should also control access to provenance data storage as provenance may be sensitive;
4. **interoperability services**: the framework should provide services supporting provenance integration across different systems.

Despite a large number of research efforts devoted to provenance management, only a few provenance infrastructures have been proposed. Chimera [1], myGrid [2], and Karma [7] are examples of provenance systems. However, the provenance models of these systems are tailored to their

specific applications and therefore are not general enough. PASS [4] is a provenance management system for file systems; it provides a custom query tool but it does not support security and different granularity levels for provenance metadata. In addition, there are two standard provenance models: Open Provenance model (OPM) [5] and PROV [11]. These two models are interoperable and generic so that they are able to represent provenance for different systems and applications. However, their major limitation is that they are not able to represent metadata about access control policies. Ni et al. [8] has proposed a provenance model that focuses on access control policies for provenance. However, Ni's model is not able to support different granularity levels. Sultana and Bertino [14] have designed an initial comprehensive provenance infrastructure. However, this infrastructure has several limitations, including the lack of a query language and lack of interoperability services for provenance. In addition, such framework has not been implemented nor integrated with an actual data management system.

In this paper, we thus design and implement the first comprehensive provenance infrastructure addressing the four requirements discussed earlier. Our contributions include:

- A data provenance model extended from the provenance model proposed by Sultana and Bertino [14]. We provide specifications of this model according to a relational and graph model.
- A mapping ontology to support interoperation of our provenance model with both OPM and PROV.
- The integration of our provenance framework with the *Computational Research Infrastructure for Science* (CRIS) [13]. CRIS is widely used at Purdue University for managing scientific data from many different research areas, including biology, bio-chemistry, water management, and social sciences.

The rest of the paper is organized as follows: Section II introduces our provenance framework including the provenance model, mapping ontology. Section III discusses related work and Section IV outlines conclusions and future work.

II. PROVENANCE FRAMEWORK

Our provenance framework as shown in Fig. 1 is composed of several components that we discuss in what follows.

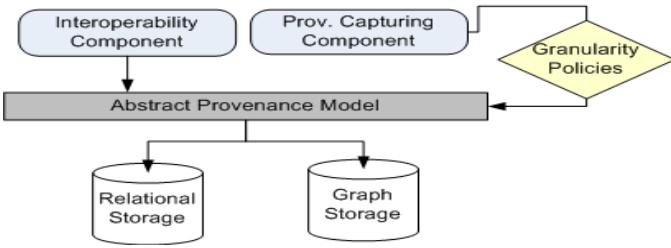


Figure 1: SimP Framework

A. Provenance Model

In our provenance model, called SimP, the provenance of a data object records the history of its input *data*, *processes*, *operations*, *communications*, *actors*, *environments*, and *access controls*.

A *process* manipulates input data objects by performing a sequence of operations to generate other data objects. A process might be a service in a user application (e.g., a workflow in a scientific experiment application) or an operating system level process (e.g., executing a script by shell command in UNIX). At a more detailed level, an *operation* executes a sub-task which is part of a process execution. The operations may generate/modify persistent data or intermediate results. In provenance, it is crucial to capture the origin of the generated data. Such information is captured by a *data lineage* entity. In data lineage, a data item is described by the source data and the operations used to derive the data item from source data. Lineage helps in producing the data dependency graph of a data object and implicitly describing the process dependency.

The operations in the same process or in two different processes interact by real or virtual messages. In our model, such interaction is referred to as *communication*. There are two types of communication between operations. The first type refers to the completion of an operation followed by the start of another operation. The second type refers to an operation executed (i.e., started and completed) within the execution of another operation. We refer to the first type of communication as *sequential* and to the second as *composition*. A communication may involve data passing if the operation which initiated the communication generates data. Examples of communications include data flow and copy-paste in UNIX. On the other hand, a process may initialize another process to be executed. The process which invokes the initialization is the parent process and the newly created process is the child process.

Processes (including their operations) and data are manipulated by *actors* which can be human subjects or workflows. Capturing information about actors, who actuate the activities changing data objects, helps in detecting intrusion, data misuse, or system changes. A user may authorize other users to perform certain activities on his behalf. In this model, *data* objects are attributed to actors to identify users who inserted input data or generated output by executing a process. Processes also have a context that affects their execution and output. Such context is represented by the

environment which refers to a set of parameters and system configurations. Environment information helps in understanding the system context in which processes were executed and data output were generated.

Our model is security-aware. Following the model by Ni et al. [8], our model is able to represent the access control policies in place at the time of data manipulation by the actors. Information about access control policies include which actors are authorized to utilize which processes and operations on which data. Such information is modeled by the *Access Control Policy* entity which includes actor and policy information. The policy object might refer to processes, or operations specified by the policy subject.

All fundamental provenance entities contain a domain attribute which might be used to specify the scope of provenance information (e.g., where processes and data manipulations executed) especially when providing a provenance storage for different systems. In addition, the domain value might include more detailed scope information (e.g., a particular application or workflow). The domain attribute is essential for providing an abstract domain view of the provenance graph.

Our framework supports the specification of the provenance model according to two representations: relational and graph.

1) Relational Representation

The relational model representation of SimP is shown in Fig. 2. Based on the abstract description of our model, the fundamental entities are stored in six fundamental tables (i.e., *Data*, *Processes*, *Operations*, *Actors*, *Environments*, and *Access Control Policies*). In addition, there are tables maintaining many-to-many relationships among the fundamental tables (i.e., *Lineages*, *Communications*, *Process Input Data*, *Process Output Data*, *Operation Input Data*, *Operation Output Data*, and *Delegations*). These tables are connected through a set of referencing relations (i.e., foreign key constraints). Each table has a unique identifier and consists of several attributes.

Each data record contains description, value, and actor ID. An example of a data object is a file. The actual data object identifier is different from data record identifier in the provenance storage. Suppose that the data object is a script file named *f_i*. This file might be edited by different users at different times. Thus, the provenance storage contains multiple records for *f_i* and each record is identified by different data ID but the description attributes of these records are similar (i.e. *f_i*). In our model, every data object is attributed to an actor to trace who created the object or generated it by executing a process.

Each process record has a unique ID and is executed by an actor in a certain environment. A process manipulates certain data and may generate other data so process's input and output data are recorded in the Process Input Data and Process Output Data tables. Processes are categorized into two types: application process or system process. If the process belongs to an application (e.g., a scientific workflow), it contains a workflow ID. Otherwise, it is a system process. In the case of a

system process, the workflow ID attribute refers to the workflow hosting the application process which encapsulates the system process. Each process has a description which is the actual identifier of the process (e.g., executing the script file (*f*)). A process might be executed multiple times. Therefore, in order to capture provenance for the same process multiple times we have an automatic generated ID to distinguish process records. A process might be initialized (i.e., forked) by another process so we store such information in the parent process attribute.

Operation record attributes include ID, description, and process ID. Depending on applications, the operation description attribute might contain different definitions (e.g., a function name or a block of source code statements). The output of an operation might be intermediate or persistent. The output is intermediate if it is used as input for another operation while a persistent output is final. The persistent output of operations is considered also as the output for the container process. On the other hand, all input data of a process can be input data for all its contained operations.

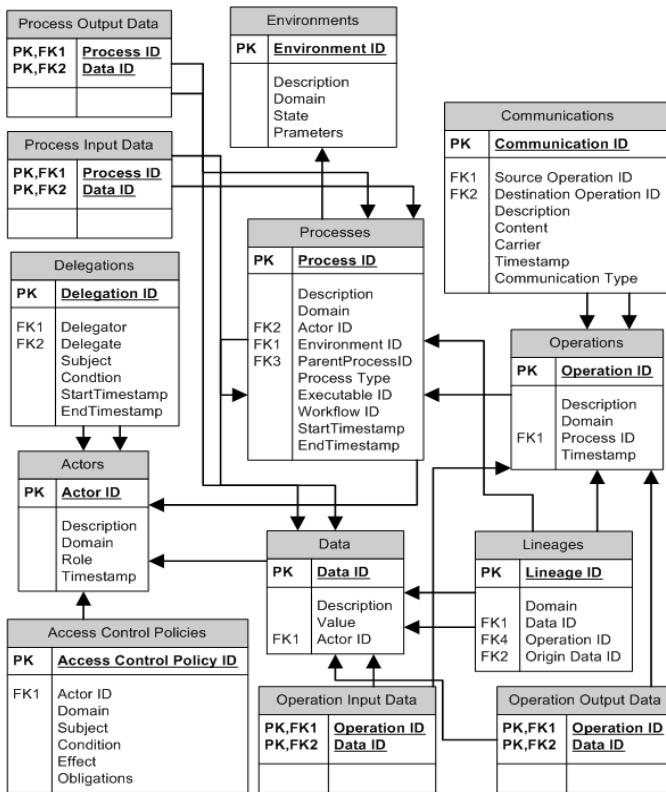


Figure 2: ER-Diagram of the SimP Model

Communication record attributes include description, carrier, the source operation ID, the destination operation ID, and type of communication. The detail level of the communication description depends on the applications. The communication channel (e.g., HTTP or SOAP) is described by the carrier attribute. The type of communication between two operations might be *sequential* or *composition*.

Lineage record attributes include lineage ID, data ID, and the ID of the operation that produced this data operating on the

input data identified by the data ID. A data item has multiple lineage records if it is generated by an operation which received multiple input data.

The attributes of an actor record include actor ID, description, and role. A role is a job of the actor. For an actor, the most recent record contains its current job while its previous records might include its previous jobs. An actor can delegate the execution of a process to another actor. Such information is stored in a Delegation record which contains delegator, delegate, subject, condition, start timestamp, and end timestamp. The subject might refer to processes or operations while the condition refers to the identifier of the subject.

The content of an environment record includes domain, state, and parameters attributes. In our model, each process belongs to one environment while an environment contains multiple processes. An environment timestamp is determined by its parent process record (i.e., the first process executed within the environment).

The attributes of an access control policy record include access policy ID, actor ID, subject, condition, effect, obligations. Such records capture the access control enforced at the time when the data of interest has been manipulated. The actor ID records the actor privileged to perform certain operations or processes. The policy subject might refer to processes, operations, or communications while the condition refers to the identifier (e.g., process or operation description) of the subject.

2) Graph Representation

The relational model is suitable for storing provenance in a relational database. However, as the standard provenance models (i.e., OPM and PROV) are modelled according to a graph-based format, developing an interoperability ontology mapping from these models onto our model requires a corresponding graph representation of our model. Thus, we also introduce a graph model specification of SimP.

Our graph model consists of six nodes and twelve types of edges. The graph nodes include *Data*, *Process*, *Operation*, *Actor*, *Environment*, and *Access Control policy*. Each graph node has a set of attributes similar to its corresponding entity table described in the relational model. Furthermore, an edge represents a relationship between the source of the edge and the destination of the edge. The kinds of the relationship (and thus the meaning of the edges) are specified by the types of the nodes that are the end-points of the edges. These types of relations are defined as follows:

- *used*: an edge that connects a source Process or Operation to a destination Data. It indicates that the process/operation required the availability of certain data to be able to complete its execution.
- *wasGeneratedBy*: an edge that connects a source Data to a destination Process or Operation. It indicates that the process/operation was required to initiate its execution for the data to have been generated.
- *wasDerivedFrom*: an edge that connects a source Data to a destination Data. It indicates that the destination

data needs to have been generated for the source data to be generated.

- *wasExecutedBy*: an edge that connects a source Process to a destination Actor. It indicates that a process with all its operations was executed by the actor.
- *wasInformedBy*: an edge that connects a source Operation to a destination Operation. It indicates that the execution of the destination operation was followed by the execution of the source operation.
- *wasEncapsulatedBy*: an edge that connects a source Operation to a destination Operation. It indicates that the execution of the source operation is part of the execution of the destination operation.
- *wasPartOf*: an edge that connects a source Operation to a destination Process. It indicates that the execution of the source operation is part of the execution of the destination process.
- *wasForkedBy*: an edge that connects a source Process to a destination Process. It indicates that the execution of the source process is initiated by the destination process.
- *wasInContext*: an edge that connects a source Process to a destination Environment. It indicates that the execution of the source process was in the context of the destination Environment.
- *wasGrantedTo*: an edge that connects a source Access Control Policy to a destination Actor. It indicates that the source access control policy was enforced on the destination actor at provenance capture time.
- *actedOnBehalfOf*: an edge that connects a source Actor to a destination Actor. It indicates that the execution of the action performed by the source actor was delegated to the source actor by the destination actor.
- *wasAttributedTo*: an edge that connects a source Data to a destination Actor. It indicates that the source data was manipulated by the destination actor.

Based on the proposed provenance specifications for relational and graph-based representations, our framework supports two types of storage: relational database (i.e., MySQL) or graph database (i.e., Neo4J) for storing provenance metadata. For this sake, our framework has an abstract storage interface. This abstract interface communicates with either MySQL adapter or Neo4J adaptor. The default storage is Neo4J and the framework can be configured to change to the second database. The administrator is able to change the target storage type at any time but he should first import the database from Neo4J to MySQL (or vice versa).

B. Interoperability With Other Provenance Models

Provenance interoperability refers to the ability to integrate and convert provenance information represented by different provenance models in order to facilitate provenance data interchange. Our model supports interoperability with two well-known standard provenance models: OPM [5] and PROV

[11]. Below we provide some background about OPM and PROV model ontologies and address the interoperation challenge by defining a mapping ontology that maps provenance information expressed in each of those two standard models into provenance information expressed in SimP.

1) Mapping from OPM Standard Model

OPM [5] represents provenance information as a directed graph which consists of nodes and edges. The nodes comprise three types of entities: *Artifacts* (i.e., data) which represent resources, *Processes* which represent actions or steps of action performed on artifacts, and *Agents* (i.e., actuators) which control the processes. The edges represent the dependencies and relations among the entities. There are five types of edges: *used*, *wasControlledBy*, *wasGeneratedBy*, *wasDerivedFrom*, and *wasTriggeredBy*. Each edge is distinguished by its source and destination: a process used an artifact, a process was controlled by an agent, an artifact was generated by a process, an artifact was derived from another artifact, and a process was triggered by another process. OPM also introduced the concept of *account* to represent a particular view of the provenance.

The conversion from OPM to our provenance model is straightforward because of the rich vocabularies in our model. Table 1 shows the mapping from the OPM graph to our provenance model graph. Besides the intuitive mapping provided in Table 1, we need to consider the following:

- OPM does not have a finer granularity level of description for a process as a set of operations. When converting a process from OPM to SimP, we create a process and an operation and link them by a *wasPartOf* edge. Mapping an OPM process into a SimP operation assumes that the relations connected with the OPM process are mapped onto relations with SimP operation. So the purpose of creating a dummy SimP process is only to host the created SimP operation.
- In OPM, the intercommunication between a process and another process is identified by one edge type, that is, *wasTriggeredBy*, which does not capture the exact meaning of process-process relations (i.e., communication relation or parent-child relation). In our model, these relations are represented by three types of edges (*wasForkedBy*, *wasEncapsulatedBy*, and *wasInformedBy*). *WasForkedBy* represents the parent-child relation between two processes. On the other hand, *wasInformedBy* and *wasEncapsulatedBy* represent communications between two operations (in the same process or different processes). More specifically, a *wasInformedBy* represents a sequential communication and a *wasEncapsulatedBy* represents a composition communication. When performing a conversion, we map the OPM *wasTriggeredBy* edge to *wasInformedBy* in our model under the assumption that all OPM processes are related with sequential communication.
- OPM supports a particular view of provenance by including the “*account*” attribute of the graph nodes. By contrast, in our model, we support such view of provenance using the environment node. So each node in OPM that has an

account attribute is mapped onto a *wasInContext* edge connecting the mapped node with the environment node.

TABLE 1: MAPPING FROM OPM TO SIMP

	OPM	SimP
Nodes	Process	Process, Operation, WasPartOf
	Artifact	Data
	Agent	Actor
Edges	Used	Used
	WasGeneratedBy	WasGeneratedBy
	WasDerivedFrom	WasDerivedFrom
	WasControlledBy	WasExecutedBy
	WasTriggeredBy	WasInformedBy

2) Mapping from PROV Standard Model

A W3C provenance standard model called PROV [11] was published in 2013 based on a revision of OPM. A PROV graph contains three types of nodes: *Entities* (i.e., data) to represent resources, *Activities* to represent actions performed on entities and *Agents* to model parties responsible for activities. Additionally, PROV includes seven types of edges: *used* (some entity was used by an activity), *wasAssociatedWith* (an agent was engaged in some activity), *wasGeneratedBy* (an entity was generated by an activity), *wasDerivedFrom* (an entity derived another entity), *wasAttributedTo* (an agent used an entity), *actedOnBehalfOf* (an agent acted on behalf of another agent) and *wasInformedBy* (an activity sent its result data to another activity).

Table 2 shows the mapping from a PROV graph to a SimP graph. In addition, we address the same issues discussed for the mapping from OPM.

TABLE 2: MAPPING FROM PROV TO SIMP

	PROV	SimP
Nodes	Activity	Process, Operation, WasPartOf
	Entity	Data
	Agent	Actor
Edges	Used	Used
	WasGeneratedBy	WasGeneratedBy
	WasDerivedFrom	WasDerivedFrom
	WasAssociatedWith	WasExecutedBy
	WasInformedBy	WasInformedBy
	WasAttributedTo	WasAttributedTo
	ActedOnBehalfOf	ActedOnBehalfOf

By following the mapping ontology described earlier, we implemented a conversion tool that facilitates the conversion from the standard models (OPM, and PROV) to SimP model. The input of the tool is an XML -formatted file containing data provenance encoded according to the OPM model or the PROV model. The conversion tool stores the converted provenance data into our provenance storage, that is, MySQL or Neo4J.

C. Integration with the CRIS System

We integrated our provenance model in the *Computational Research Infrastructure for Science* (CRIS) [13]. CRIS is a

scientific data management workflow cyberinfrastructure for scientists lacking extensive computational expertise. The application is currently used by a community of users in Agronomy, Biochemistry, Bioinformatics, and Health Care Engineering at Purdue University. Previously, CRIS had its own provenance model mainly based on versioning mechanism. Such mechanism is not able to capture provenance at different granularities since it is data centric. Our model is data and operational centric in that we maintain metadata about the derivation of every data object (i.e., what is the origin data object and what is the deriving operation).

Within the CRIS system, we have a provenance logging component based on aspect oriented programming to instrument the application code. The logging component collects a set of appropriate provenance logs while the CRIS is running. The logs use the XML-based representation of provenance records to facilitate parsing them into the SimP model. Periodically, the CRIS provenance logs are fetched and converted into another XML-format file. The new XML file contains a data dependency provenance graph following the SimP representation ontology.

D. Security

Due to the sensitivity of data provenance information collected and stored in provenance storage, security is an essential factor and requirement. Hence, we incorporated an additional entity to specify the privileges granted to actors for accessing the provenance storage. Such entity, referred to as Provenance Query Authorization, records information about which actors has which authorizations. An authorization is expressed as subject, condition, and effect fields. The subject refers to the type of provenance storage entity (e.g., Processes entity) while the condition field identifies the provenance entity (e.g. Process Description value). The effect field indicates the authorization status (e.g., granted or revoked).

The security requirement in our framework is thus addressed by the following features: a) the access control policy which is a main entity in the SimP model; and b) the provenance query authorization to secure the access to the sensitive information in the provenance storage.

E. Granularity

Another key feature of our provenance framework is the ability to specify and modify the granularity level needed to capture provenance for specific records or entities. For this purpose, in our provenance database, we include an additional entity, referred to as *Granularity Policy*, which is not part of the provenance model but it is part of our provenance framework. A granularity policy enables users to specify the desired level of provenance details to be captured and stored (e.g., capturing provenance at the activity level in scientific provenance workflow, or the operating system level to capture system configurations). A Granularity Policy record includes the following fields: granularity policy ID, actor ID, subject, condition, granularity type. The subject may refer to the targeted process, operations, or communications, whereas the condition refers to the identifier of the subject. The granularity type is the detail level of the required provenance.

The multi-granularity requirement in our framework is thus addressed by the following features: a) a provenance model able to represent provenance metadata for data objects at various granularity levels; b) support for the integration with systems utilizing different provenance capturing mechanisms (i.e., provenance for workflow data or file system); and c) support for granularity preferences based on granularity policies.

III. RELATED WORK

Chimera [1], myGrid [2], and Karma [7] are examples of workflow-based provenance systems in which the captured provenance is about data-centric workflows. In Chimera, workflow data is represented by a special language, referred to as Virtual Data Language (VDL) which describes provenance as relationships among datasets, procedures, invocations of procedures or tasks. In myGrid, the provenance model represents service invocations and their information (i.e., parameters, start and end times, data products used and derived). Karma collects provenance at three levels (i.e., Workflow, Service, and Application) and it uses the concept of activities (e.g., Workflow-Started or Workflow-Finished) that take place at these three levels to collect provenance.

On the other hand, PreServ [12] is an example of process-based provenance systems. A process-based provenance model represents the relationships between services and data. These relationships have three categories: service-service (i.e., source and sink service), data-data (i.e., data derivation), service-data (a service consumes data or a service produces data).

An example of operating system based provenance system is PASS [4]. PASS works at the level of shared storage system to capture provenance about executed programs, their inputs, and outputs. Another example is ES3 [6] which records provenance metadata including input/output data objects, and domain names.

As already mentioned, models of existing provenance systems apply only to specific applications/domains and do not support security. OPM and PROV provide standard provenance representations. However, they are not able to represent information on access control policies. OPM Toolbox [9] allows one to create OPM graphs while ProvToolbox [15] allows one to create PROV graphs and convert between PROV data model representations (e.g., XML or JSON).

The provenance model by Ni et al. [8] (extended by Cadenhead et al. [10]) is a general model supporting security. This model represents provenance data at a granularity of operation and thus is unable to distinguish between different granularity levels. Sultana and Bertino [14] provide an initial comprehensive provenance infrastructure which we extend in this paper.

IV. CONCLUSION AND FUTURE WORK

In this paper, we introduce SimP - a comprehensive provenance framework integrated with the scientific data

management system CRIS [13]. The framework includes a comprehensive provenance model provided with relational and graph specifications. Our provenance model is interoperable with the OPM and PROV provenance models.

As future work, we plan to design and implement specialized query languages for our framework and investigate efficient compression techniques for our provenance model.

ACKNOWLEDGMENT

This work is supported in part by NSF under award CICI-1547358. The opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any of the sponsors.

REFERENCES

- [1] Foster, I., Vöckler, J., Wilde, M. and Zhao, Y., 2002. Chimera: A virtual data system for representing, querying, and automating data derivation. In Scientific and Statistical Database Management, 2002. Proceedings. 14th International Conference on (pp. 37-46). IEEE.
- [2] Zhao, J., Goble, C., Stevens, R. and Bechhofer, S., 2004. Semantically linking and browsing provenance logs for e-science. In Semantics of a Networked World. Semantics for Grid Databases (pp. 158-176). Springer Berlin Heidelberg.
- [3] Simmhan, Y.L., Plale, B. and Gannon, D., 2005. A survey of data provenance in e-science. ACM Sigmod Record, 34(3), pp.31-36.
- [4] Muniswamy-Reddy, K.K., Holland, D.A., Braun, U. and Seltzer, M.I., 2006, June. Provenance-Aware Storage Systems. In USENIX Annual Technical Conference, General Track (pp. 43-56).
- [5] Moreau, L., Freire, J., JFutrelle, J., McGrath, R.E., Myers, J., aulson, P. The Open Provenance Model (v1.00), Tech. Rep., University of Southampton, URL <http://eprints.ecs.soton.ac.uk/14979/1/opm.pdf>, 2007.
- [6] Bowers, S., McPhillips, T., Riddle, S., Anand, M.K. and Ludäscher, B., 2008. Kepler/pPOD: Scientific workflow and provenance support for assembling the tree of life. In Provenance and Annotation of Data and Processes (pp. 70-77). Springer Berlin Heidelberg.
- [7] Simmhan, Y.L., Plale, B. and Gannon, D., 2008. Query capabilities of the Karma provenance framework. Concurrency and Computation: Practice and Experience, 20(5), pp.441-451.
- [8] Ni, Q., Xu, S., Bertino, E., Sandhu, R. and Han, W., 2009. An access control language for a general provenance model. In Secure Data Management (pp. 68-88). Springer Berlin Heidelberg.
- [9] OPM Toolbox, <https://github.com/lucmoreau/OpenProvenanceModel>
- [10] Cadenhead, T., V. Khadilkar, M. Kantarcioglu, and B. Thuraisingham. "A language for provenance access control." In CODASPY, pp. 133-144. ACM, 2011.
- [11] PROV-Overview: <http://www.w3.org/TR/2013/NOTE-prov-overview-20130430/>
- [12] Hoekstra, R. and Groth, P., 2013. Linkitup: link discovery for research data. In AAAI Fall Symposium Series Technical Reports (No. FS-13-01, pp. 28-35). AAAI Publications.
- [13] Dragut, Eduard C., et al. "CRIS—Computational research infrastructure for science." Information Reuse and Integration (IRI), 2013 IEEE 14th International Conference on. IEEE, 2013.
- [14] Sultana, S. and Bertino, E., 2015. A Distributed System for The Management of Fine-grained Provenance. Journal of Database Management (JDM), 26(2), pp.32-47.
- [15] ProvToolbox, <https://github.com/lucmoreau/ProvToolbox>

A Comprehensive Scenario Agnostic Data LifeCycle Model for an Efficient Data Complexity Management

Amir Sinaeepourfard, Jordi Garcia, Xavier Masip-Bruin, Eva Marín-Tordera

Advanced Network Architectures Lab (CRAAX),
 Universitat Politècnica de Catalunya (UPC, BarcelonaTech),
 Barcelona, Spain
 {amirs, jordig, xmasip, eva}@ac.upc.edu

Abstract— There is a vast amount of data being generated every day in the world, coming from a variety of sources, with different formats, quality levels, etc. This new data, together with the archived historical data, constitute the seed for future knowledge discovery and value generation in several fields of eScience. Discovering value from data is a complex computing process where data is the key resource, not only during its processing, but also during its entire life cycle. However, there is still a huge concern about how to organize and manage this data in all fields, and at all scales, for efficient usage and exploitation during all data life cycles. Although several specific Data LifeCycle (DLC) models have been recently defined for particular scenarios, we argue that there is no global and comprehensive DLC framework to be widely used in different fields. For this reason, in this paper we present and describe a comprehensive scenario agnostic Data LifeCycle (COSA-DLC) model successfully addressing all challenges included in the 6Vs, namely Value, Volume, Variety, Velocity, Variability and Veracity, not tailored to any specific environment, but easy to be adapted to fit the requirements of any particular field. We conclude that a comprehensive scenario agnostic DLC model provides several advantages, such as facilitating global data organization and integration, easing the adaptation to any kind of scenario, guaranteeing good quality data levels, and helping save design time and efforts for the research and industrial communities.

Keywords—Data LifeCycle, Data Management, Data Complexity, Vs Challenges

I. INTRODUCTION AND MOTIVATION

Hundreds of terabytes are being generated every second all over the world. Data is created from multiple, different and distributed sources and devices (Smart Cities, IoT, scientific modeling and simulations [1], users' social, professional and everyday activities, etc.), with distinct data types and formats. Such huge flow of heterogeneous and often noisy data is accumulated over the historical data sets, constituting the complex universe of digital data. This data can then be used in different forms (reading, writing, transforming or removing), therefore drawing the life cycle of data. Managing data during its life cycle, i.e., from data creation and collection, to data storing and processing, becomes a complex and challenging issue [2]. The ultimate goal of managing data is to obtain some

kind of value, by generating knowledge or discovering new insights. To that end several complex computing processes are required, where data is the key resource during all its entire life. For this reason, efficient data management and organization is a key issue for an effective extraction of value from data.

Data LifeCycle (DLC) models provide an effective solution for data management. They provide a high level framework to plan, organize and manage all aspects of data during their life stages, from data generation to knowledge extraction [2-5]. DLC models clarify all phases of data life, from production to consumption, define the policies for each phase and the relationship between phases [3]. Normally, a DLC model is usually targeted to a particular scenario, addressing its specific requirements and challenges. Consequently one effective DLC model can be defined for any science, scenario or use case [2, 5]. The benefits of designing a DLC model are i) easing for planning and handling complexity of data management in all data life stages [3, 4]; ii) preparing data products ready for end-users, matching the expected constraints and efficiency [2-4]; iii) showing elucidate the quality level of data, removing any kind of waste and noise [3]; iv) illustrating a sequence of any essential activities related to data life [3]; and v) helping designers to create sustainable software [6, 7].

In a previous work [8], we surveyed most related DLC models and evaluated qualitatively each of them in terms of a set of 6Vs challenges, namely Value, Volume, Variety, Velocity, Variability and Veracity. Although each DLC model successfully addresses its specific set of requirements and challenges, and each DLC model is targeted to its particular field of application, we concluded that: i) there is no DLC model completely addressing all 6Vs challenges, and ii) there is no DLC model to be considered comprehensive, in the sense that could be useful to manage data in any scenario or scientific field. For this reason, in this paper we present the design of a comprehensive DLC model that successfully addresses the 6Vs challenges, and that can be easily adapted to manage any scenario and science. The proposed comprehensive DLC model can be understood as an abstract model, i.e. it has been designed agnostic to any particular context and scenario. But because of its completeness, it can easily address the requirements and challenges of any specific scenario.

This rest of this paper is organized as follows. Section 2 reviews the main existing DLC models and highlights their limitations. Section 3 describes our proposal for managing data complexity, i.e., the comprehensive, scenario agnostic, DLC (COSA-DLC) model. In Section 4, the comprehensive DLC model is evaluated with respect to the 6Vs challenges as benchmark test. And finally, Section 5 highlights the contributions of this model and concludes the paper.

II. RELATED WORK

Research on data management and analysis has traditionally been oriented to the context of Relational Databases Management Systems (RDBMS) and the recent Extract-Transform-Load (ETL) process for modeling the typical data life cycles in data warehousing [9-11]. However, the advent of Big Data imposes additional difficulties to the traditional data management and analysis systems [11, 12]. DLC models are conceived to manage data beyond these systems, and consider diverse, heterogeneous, and large volumes of data, applicable to any scenario or science, throughout all their data life stages.

Several DLC models have been designed targeted to particular scenarios and addressing specific requirements and challenges. One of the first authors to define DLC were Levitin and Redman in [3]. Later, authors in [13] introduced the Data LifeCycle Management (DLM) as a policy based approach to clarify and organize the flow of data throughout all phases of the data life cycle. New generation intensive data sciences, such as eScience, are interested to manage large and heterogeneous amounts of data through Scientific Data LifeCycle Management (SDLM) [14]. And more recent, some research in the field of Big Data recognize the importance of DLC models in data organization [11, 12].

In a recent survey [8], we evaluated a number of public DLC models. We observed that some models concentrated only on a limited number of data stages, such as data discovery [2], data sharing [15], data security [16], data quality [3, 6]. We also realized that each DLC model was specifically designed to address a specific science, such as ecology in [4, 5], library and information science in [17], or geomorphology in [18], or to address special scenarios, such as Smart Cities in [19, 20], support of sciences in [7, 14], etc. Finally, we evaluated the completeness of each DLC model by analyzing to what extent the model was addressing the 6Vs challenges that we found convenient for DLC models. Although all DLC models were able to address most challenges, we found that none of them was addressing all 6Vs challenges.

As a summary, we can observe that all existing DLC models have the following limitations:

- As they have been designed for a particular scenario, they cannot be easily adaptable to any new scenario;
- As they have been designed according to specific requirements, most of them do not always include all stages of the data life cycle; and finally,
- There is not any DLC model ready to address completely all challenges included in the 6Vs.

For this reason, in this paper we present a comprehensive, scenario agnostic, DLC model that overcomes all these limitations and, therefore, can be easily adaptable to any scenario and science and, in addition, is able to address successfully the set of 6Vs challenges.

III. PROPOSING A COMPREHENSIVE SCENARIO AGNOSTIC DATA LIFE CYCLE MODEL

A comprehensive scenario agnostic DLC (COSA-DLC) model provides all aspects of data management and organization, including data collection, data process, data storage, and other issues regarding data quality, data veracity, and data security, among others. In addition, a COSA-DLC model can be customized and fitted to any science and scenario easily and quickly to achieve specific requirements and guaranteeing high level of data quality.

Some potential advantages of a COSA-DLC model are: i) managing and organizing global datasets for any future data discovery, integration, and processing; ii) providing easy customization and adoption to any science or scenario; iii) improving data quality levels in any specific context, and; iv) eliminating any additional waste and effort for designers, including data, software and system designers, to design their appropriate and efficient architecture.

Our proposed COSA-DLC model is organized in three main blocks, each one further developed into a set of more detailed phases, covering main actions related to the data management and organization for the whole data life cycles. In addition, each phase is specified in terms of the Data Lifecycle Management (DLM), which defines the phase's policies and actions, and the interrelation among phases.

A. Main blocks in the COSA-DLC model

We propose a modular design for the COSA-DLC model structured into three blocks (see Fig. 1), Data Acquisition, Data Processing and Data Preservation. These blocks are responsible for collecting and organizing high level data quality for further processing and storing purposes. This modularity eases the process of adaption to specific scenarios by tailoring the architecture to the specific scenario demands.

Data is gathered into the system through the Data Acquisition block, which collects data from different sources, assesses quality, and tags it with any additional description required in the model. Data can then be processed or stored. The Data Processing block is responsible for performing any data to information/knowledge/value transformation, through complex analysis and/or analytical techniques. Processed data can be used by end users or stored for future reuse. Finally, the Data Preservation block is responsible for data archiving, storing high quality data (curated in the acquisition and/or processing blocks). This data can then be prepared for publication or dissemination, and used by end users or in further processing steps.

The data flow is as follows. Data is created and collected through the Data Acquisition block. If data is immediately processed, this is assumed real-time data; otherwise, if stored, it is considered archivable data. Note that all or part of the

processed data can also be preserved, and vice versa, i.e. these two data sets are not exclusive. When archived data in the Data Preservation block is used for processing, this is considered historical data. So the Data Processing block is able to use both real-time and historical data for processing. Finally, the results of data processing can be stored back through the Data Preservation block: this data is considered higher value data.

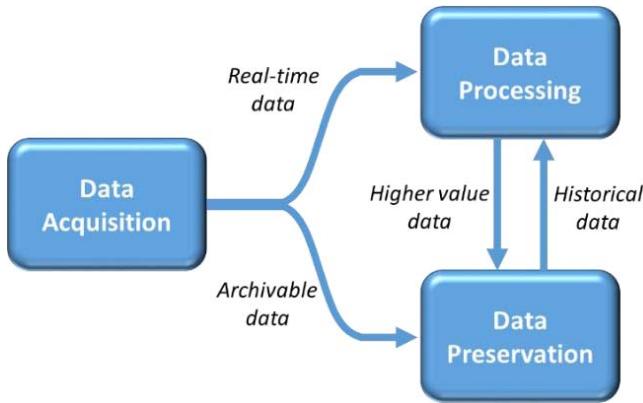


Fig. 1. Main structure of the comprehensive DLC model.

B. The COSA-DLC model

The proposed COSA-DLC model is organized in three main blocks, as described in the previous section. This set of blocks includes a sophisticated set of phases implementing all required tasks to make comprehension, agnosticism and adaption true. Thus, as shown in Fig. 2, the Data Acquisition block is developed in four phases, the Data Processing block is developed in three phases, and the Data Preservation block is developed in four phases. The description of all functionalities and activities of each phase, together with the relationship among phases, is called the Data LifeCycle Management (DLM), and is presented next.

The Data Acquisition block is made by the following four phases, namely Data Collection, Data Filtering, Data Quality and Data Description:

- The Data Collection phase aims to collect data from all sources and devices, according to the business requirements and scientific demands. Specifically, it is responsible of:
 - Collecting data, directly and indirectly, from any valid source, such as basic or complex devices (sensors, smart devices), databases, web-generated data, third party applications, etc.
 - Managing the ranges of valid and trusted sources for data collection.
 - Exploring and discovering new sources for data collection.
- The Data Filtering phase is responsible for performing some basic data transformations in order to optimize the

volume of data flowing from the collection to the quality phases. Particular data transformations are specific of the context and business requirements. However, filtering, aggregation, curation, sorting, classification, or compression, are some data transformations that could be considered as well.

- The Data Quality phase aims to appraise the quality level of collected data. It is responsible for guaranteeing both, Quality Control (QC) and Quality Assurance (QA), in particular:
 - Checking the quality level of data and discarding or repairing low quality data, according to the provided policies (QC).
 - Monitoring the quality of data flows and, in case of continuous failures, proceeding according to the provided policies (QA).
- The Data Description phase aims to tag data with some additional information for an optimal future usage. Any available metadata considered in the business model can be used, such as timing (creation, collection, modification, etc.), location or origin (city, country, coordinates), authoring, and so on.

Once the data has been described appropriately, it can be used for either processing on real-time, or for archiving for future queries over historical data.

The Data Processing block consists of the Data Process, Data Quality and Data Analysis phases:

- The Data Process phase provides a set of processes to transform (raw) data into more sophisticated data/information. These processes could include one or several internal steps, such as pre-processing or post-processing, depending on the particular business requirements.

Data considered for processing can be either real-time, just generated, data (from the Data Acquisition block), or historical archived data (from the Data Preservation block). The output of this phase is considered higher value data, meaning that this data is more mature than the original (raw) input data.

- The Data Quality phase aims to appraise the quality level of processed data. It can check both QC to the output of the processing and QA to the processing procedure.

This phase could seem redundant or repetitive with respect to the Data Quality phase in the Data Collection block; however, they perform specific checking targeted to the specific life cycles. In addition, in order to provide completeness and guarantee a maximum level of quality, any additional quality appraising is always useful.

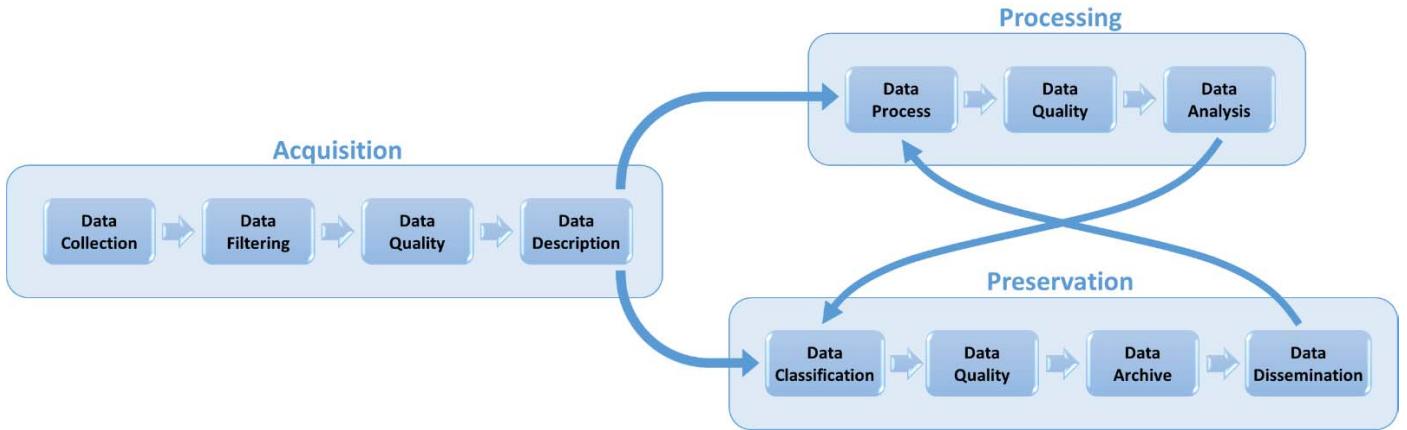


Fig. 2. The Proposed DLC model.

- The Data Analysis phase is responsible for developing all data analysis and data analytics for extracting knowledge and discovering new insights.

This phase is the last step in the procedure of value generation, and it is usually the natural interface with the end-user. Alternatively, this data can also be considered for storing, as part of the Data Preservation block, thus allowing future data re-processing.

The Data Preservation block consists of the following four phases, namely Data Classification, Data Quality, Data Archive and Data Dissemination:

- The Data Classification phase aims to organize and prepare data for efficient storage, by applying some optimization, such as classification, arrangement, compression, etc.

Furthermore, some additional descriptive information could be also attached to this data related to the archiving policies, such as access permissions, privacy, expiry time, or sharing, use and reuse capabilities. In this phase, data provenance or data versioning could be considered.

- The Data Quality phase aims to appraise the quality level of classified data, before storing. It can check both QC to the output of the classification and QA to the preservation procedure.

Again, note that this phase is specific for the data preservation block, and it is aimed at guaranteeing a maximum level of quality.

- The Data Archive phase aims to store a large set of high quality data in the available permanent or temporary storage resources. This phase must be able to perform long-term preservation over large amounts of data. It is also responsible of some additional tasks, such as data cleaning according to the corresponding expiry time or other business policies.

- The Data Dissemination phase aims to prepare archived data for private or public end-users' access. Any sharing procedures could be managed in this phase to guarantee access permissions, privacy, expiry time, or any other sharing capabilities.

This phase is the natural interface with the end-user for stored data. Additionally, this data can also be considered for processing, as part of the Data Processing block.

IV. COMPREHENSIVE DLC MODEL EVALUATION

Several authors [21-23], propose a list of problems and challenges that should be considered in large and complex data management, often related to Big Data. These contributions have already identified several challenges, such as Value, Volume, Variety, Velocity, Variability, Veracity, and some others (note they all start with V). This set of challenges is known as the Vs challenges. Thus, the Vs challenges depict some strong barriers, difficulties and complexities for data management in different scenarios. However, existing contributions propose to work with different number of challenges, including 3Vs, 4Vs, 5Vs, 6Vs, or 7Vs challenges – perhaps more in the future-. In a previous work [8], we analyzed the appropriate challenges to be addressed in DLC models and considered the aforementioned 6Vs challenges. We also revisited most DLC models and evaluated them with respect to the 6Vs challenges as benchmark test. We concluded that although each model is adequate for its particular purpose, there is no any comprehensive model that addresses the 6Vs challenges completely. In this section we evaluate the proposed COSA-DLC model in order to demonstrate it is certainly comprehensive according to the 6Vs challenges.

1. **Value:** The value challenge refers to the valuable information that can be extracted from (a huge volume of) data, after some processing and/or analysis steps.

The proposed COSA-DLC model has several merits to address the value challenge. Firstly, the Data Process and Data Analysis phases are precisely included for extracting

value from data. Secondly, all Data Quality phases included in the model guarantee a high level of data quality and, therefore, data is more valuable. In fact, any DLC model, just because of its nature, induces to be designed for obtaining any kind of goal, or benefit. So this challenge can be assumed for any DLC model.

2. **Volume:** The volume challenge refers to the huge volumes of data, in any format, that must be considered for management.

The proposed COSA-DLC model addresses this challenge in the Data Collection phase, as it is prepared to collect data from multiple sources, and in the Data Archive phase, as it should be designed to store large amounts of data. Again, any DLC model is able to address this challenge by definition.

3. **Variety:** The variety challenge refers to the diverse types and formats of the data to be considered, mainly because they provide from different sources with different types.

The proposed COSA-DLC model addresses this challenge in the Data Collection phase by collecting data, directly and indirectly, from any source, such as basic or complex devices (sensors, smart devices), databases, web-generated data, third party applications, etc. In this phase new sources for data collection are also explored, therefore expanding the candidate sources for data collection. In addition, other phases, such as Data Filtering, Data Description or Data Classification have been included precisely for supporting data organization and classification with high variety of formats.

4. **Velocity:** The velocity challenge refers to the speed rate of data stream generation and the subsequent capability to process it efficiently. This challenge is closely related to performance.

The proposed COSA-DLC model has been designed for achieving high performance, both during data stream collection and during data processing. For this reason, a specific phase is proposed to manage each of these tasks, namely Data Collection and Data Process. Of course, the final performance will depend on the particular resources deployment, but by considering specific phases, the design helps allocating specific resources in these steps.

5. **Variability:** The variability challenge refers to the possibility that historical data varies its semantic meaning over time.

The proposed COSA-DLC mode includes a Data Description phase to tag data for future usage and a Data Classification phase where additional tagging could be done, including expiring date. The Data Archive phase also offers the option to implement some data cleaning policies. Finally, in the Data Analysis phase, some data analytics processes could be implemented to analyze and predict eventual context changes.

6. **Veracity:** The veracity challenge can also be understood from two perspectives, according to different authors' interpretations: data quality or data security. Data quality concepts include quality of control (QC) and quality of assurance (QA). And data security prevents datasets from any kind of modification from unsecured and unauthorized sources and devices.

The proposed COSA-DLC model includes a Data Quality phase in all blocks (Data Acquisition, Data Processing and Data Preservation), guaranteeing both QC and QA. First, all data is checked and if quality is too low (according to the business model), this can be discarded (QC). If a low quality level is reported continuously, the whole process can then be checked, in order to improve procedures for better quality and performance (QA).

Furthermore, the COSA-DLC model is able to address the data security challenge in different phases. Initially, by guaranteeing sources to be secure and trusted during data collection. Some additional metadata can also be included during the Data Description phase to implement some eventual encryption mechanisms. In addition, during Data Dissemination, different access policies can be defined and implemented to manage accesses, permissions, etc. And finally, a deep security analysis can be performed on data during the data quality phase.

V. CONCLUSIONS

In this paper we have presented a comprehensive scenario agnostic DLC (COSA-DLC) model, and demonstrated its completeness with respect to the 6Vs challenges. This model is abstract, as it has not been designed for any specific scenario; however, it can be easily adapted to any particular scenario or eScience, where data complexity has to be addressed. In fact, adapting the COSA-DLC model just requires selecting those phases that are relevant according to the specific scenario requirements

The advantages of the COSA-DLC model are numerous. It is an interesting reference for data engineers that must design a new DLC model for their particular environment. Instead of designing from scratch, they can use this model and easily adapt it to fit their requirements and business model, thus saving design time. In addition, eventual modifications or extensions can also be made, just keeping in mind the original COSA-DLC model. Furthermore, note that during the adaption process some additional facilities are provided, such as the facility to analyze and detect an eventual lack of data quality checking. On the other side, the COSA-DLC model has been proved to address all 6Vs challenges considered in this work. This means that any adaption of this model will also be ready to address these challenges, as long as the particular business model requires such feature.

As part of our future work, we are adapting the COSA-DLC model to a real Smart City scenario with complex and flexible data management requirements. We will implement

the different phases of this model and test the performance of the new model.

ACKNOWLEDGMENT

This work was supported by the Spanish Ministry of Economy and Competitiveness and by the European Regional Development Fund, under contract TEC2015-66220-R (MINECO/FEDER) and by the Catalan Government under contract 2014SGR371 and FI-DGR scholarship 2015FI_B100186.

REFERENCES

- [1] R. Darby, S. Lambert, B. Matthews, et al., "Enabling scientific data sharing and re-use," in IEEE 8th International Conference on E-Science (e-Science), 2012, pp. 1-8.
- [2] R. Grunzke, A. Aguilera, W. E. Nagel, et al., "Managing complexity in distributed Data Life Cycles enhancing scientific discovery," in IEEE 11th International Conference on E-Science (e-Science), 2015, pp. 371-380.
- [3] A. Levitin and T. Redman, "A model of the data (life) cycles with application to quality," *Journal of Information and Software Technology* on Elsevier, vol. 35, pp. 217-223, 1993.
- [4] W. K. Michener and M. B. Jones, "Ecoinformatics: supporting ecology as a data-intensive science," *Journal of Trends in ecology & evolution*, vol. 27, pp. 85-93, 2012.
- [5] J. Rüegg, C. Gries, B. Bond-Lamberty, et al., "Completing the Data Life Cycle: using information management in macrosystems ecology research," *Journal of Frontiers in Ecology and the Environment*, vol. 12, pp. 24-30, 2014.
- [6] J. M. Schopf, "Treating data like software: a case for production quality data," in Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries, 2012, pp. 153-156.
- [7] W. Lenhardt, S. Ahalt, B. Blanton, L. Christopherson, and R. Idaszak, "Data management Lifecycle and Software Lifecycle management in the context of conducting science," *Journal of Open Research Software*, vol. 2, 2014.
- [8] A. Sinaeepourfard, X. Masip-Bruin, J. Garcia, and E. Marín-Tordera, "A Survey on Data Lifecycle Models: Discussions toward the 6Vs Challenges," Technical Report (UPC-DAC-RR-2015-18), 2015.
- [9] S. Henry, S. Hoon, M. Hwang, D. Lee, and M. D. DeVore, "Engineering trade study: extract, transform, load tools for data migration," in IEEE Conference on Design Symposium, Systems and Information Engineering, 2005, pp. 1-8.
- [10] S. Kurunji, T. Ge, B. Liu, and C. X. Chen, "Communication cost optimization for cloud Data Warehouse queries," in IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom), 2012, pp. 512-519.
- [11] H. Hu, Y. Wen, T.-S. Chua, and X. Li, "Toward scalable systems for big data analytics: A technology tutorial," *Journals & Magazines on IEEE Access*, vol. 2, pp. 652-687, 2014.
- [12] F. L. F. Almeida and C. Calistrut, "The main challenges and issues of big data management," *International Journal of Research Studies in Computing*, vol. 2, 2012.
- [13] M. Rouse. (2010). Data Life Cycle management (DLM) definition. Available on: <http://searchstorage.techtarget.com/definition/data-life-cycle-management>
- [14] Y. Demchenko, Z. Zhao, P. Grossi, A. Wibisono, and C. De Laat, "Addressing big data challenges for scientific data infrastructure," in IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom), 2012, pp. 614-617.
- [15] A. Burton and A. Treloar, "Publish my data: a composition of services from ANDS and ARCS," in IEEE 5th International Conference on E-Science (e-Science), 2009, pp. 164-170.
- [16] X. Yu and Q. Wen, "A view about cloud data security from data life cycle," in International Conference on Computational Intelligence and Software Engineering (CiSE), 2010, pp. 1-4.
- [17] J. Starr, P. Willett, L. Federer, C. Horning, and M. L. Bergstrom, "A collaborative framework for data management services: the experience of the University of California," *Journal of eScience Librarianship*, vol. 1, p. 7, 2012.
- [18] L. Hsu, R. L. Martin, B. McElroy, K. Litwin-Miller, and W. Kim, "Data management, sharing, and reuse in experimental geomorphology: Challenges, strategies, and scientific opportunities," *Journal of Geomorphology*, vol. 244, pp. 180-189, 2015.
- [19] M. Emaldi, O. Peña, J. Lázaro, and D. López-de-Ipiña, "Linked Open Data as the fuel for Smarter Cities," in *Modeling and Processing for Next-Generation Big-Data Technologies*, ed: Springer, 2015, pp. 443-472.
- [20] J. Jin, J. Gubbi, S. Marusic, and M. Palaniswami, "An information framework for creating a Smart City through Internet of Things," *Journal of Internet of Things Journal on IEEE*, vol. 1, pp. 112-121, 2014.
- [21] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, et al., "The rise of "big data" on cloud computing: Review and open research issues," *Journal of Information Systems on Elsevier*, vol. 47, pp. 98-115, 2015.
- [22] R. Rossi and K. Hirama, "Characterizing Big Data Management," *Journal on Issues in Informing Science and Information Technology (IISIT)*, vol. 12, 2015.
- [23] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Journal on Mobile Networks and Applications*, vol. 19, pp. 171-209, 2014.

Semantic Accountable Matchmaking for E-Science Resource Sharing

Zeqian Meng

School of Computer Science

The University of Manchester

Email: zeqian.meng@manchester.ac.uk

John Brooke

JMB Associates Ltd.

Manchester

Email: jmartinbrooke@gmail.com

Rizos Sakellariou

School of Computer Science

The University of Manchester

Email: rizos@manchester.ac.uk

Abstract—E-Science is inherently a collaborative activity, which enables users from different institutions to share computational resources for analysing data. These resources are provided by different distributed systems, e.g. Clouds, Grids, local Clusters. These can have different access and accounting mechanisms and the providers may have no direct connection to the institutions involved in the collaboration. We address the problem of how e-Science collaborations can manage access to such resources within the collaboration in a fair and accountable way. This requires accountable matchmaking on a fine grained level (e.g. per user request). This is enabled by a standard-based ontology in our work that extends models widely used for resource matching but provides extra functionality in the accounting domain.

I. INTRODUCTION

E-Science is increasingly a collaborative activity, where researchers from different organisations pursue common research goals, to share their knowledge, data and expertise. To do this they must utilise computing resources from e-Infrastructures [1]. An e-Infrastructure can be established using various models, including Grids, Clouds (private or hybrid) and individually administered parallel Clusters [2]. We do not wish our e-Scientists to be tied to any single one of these models but to use resources administered in all these different ways. Attempts to develop such interoperability have given rise to the development of common standards for resource access [3]. The existence of such standards makes it possible to supply e-Scientists with a large pool of resources available from multiple infrastructures to meet their specific demands [4]. More choice of resources can contribute to improved throughput and increased success rate of jobs [5], which can shorten research lifecycle.

However this interoperability comes at a price. It is no longer realistic to expect both the resources of the collaboration and the control of resources shared within the collaboration to be applied by the providers because the relationship of user-provider is too dynamic [3]. Other methods of management of both e-Scientists and computing infrastructures, e.g. Virtual Organization Membership Service (VOMS) [6], require a more stable relationship of users and providers, often involving long term contracts. We therefore distinguish the management of the collaboration of e-Scientists from the management of the infrastructures that offer computing resources [7], as shown in Fig. 1. Here the e-Scientists in the

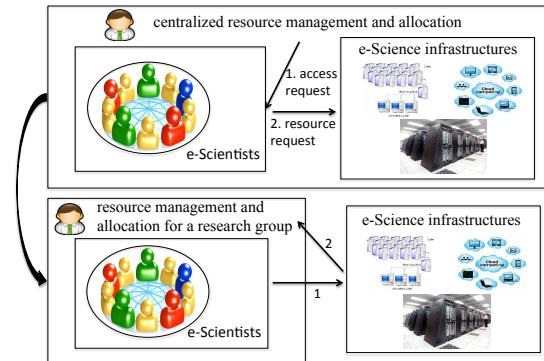


Fig. 1: E-Science resource management

collaboration require user access points (e.g. software running on the individual machines of the e-Scientists) served by a software as collaboration manager [8], that can run on a server accessible to the whole collaboration. The collaboration manager reaches resource provision agreements with the resource provider infrastructures and provides fine-grained accountable resource matchmaking, meaning that resource allocation and consumption can be measured per job. In this paper, this is achieved by developing:

- 1) An ontology model extended from a widely applied information model, which facilitates fine-grained accountable resource sharing for e-Science enabling models, including Grid, Cloud and Cluster.
- 2) An implementation of the proposed ontology model with fine-grained accounting properties in authorisation policies, which are managed by a research group and conduct accounting per job request for application execution.

We evaluate the proposed semantic model and developed programs using Amazon Web Services (AWS).

II. METHODOLOGY

A. Semantic Accountable Resource Matchmaking

Defined as a “formal, explicit specification of a shared conceptualisation” [9], an ontology describes terms, relations, and constraints of domain knowledge. The semantic features of an ontology include transitivity, reflexivity, existence of inverse,

and complement of properties. These are not supported by non-semantic methods [10], but are important to enable cross-domain knowledge sharing. Ontologies also permit a reasoning capability that enables automatic and dynamic information inheritance, inference, collection, processing, and generation [11].

So far, much work has been proposed for semantic matchmaking to facilitate e-Science resource sharing in resource level, namely a user has to supply all required resource details, including CPUs, memory size, network, etc. [4][5]. In addition to resource-level matchmaking, work in [12] considers enabling resource providers to register and manage their resources for trading. Some of these papers present the terms used to model resource information, which are project-specific and do not consider accountable properties. By contrast our work bases its ontology upon a widely applied and comprehensive information model, and extends it for fine-grained accountable resource-oriented and application-oriented matchmaking for resource sharing in research group level.

B. Information Model Based on GLUE 2.0

Grid Laboratory Uniform Environment 2.0 (GLUE 2.0) [13] is a widely applied conceptual information model for Grid entities [3]. It includes also concepts of Cloud to support growing virtualisation-based e-Infrastructures. A comprehensive and widely used model can ease extension, collection, and processing of information from a specific infrastructure.

In view of e-Science resource sharing, GLUE 2.0 includes concepts for resource, service, application, execution environment, as well as participant organisations, projects, and persons. This makes GLUE 2.0 suitable to manage a complete resource sharing lifecycle in e-Science collaborations, including matchmaking, authorisation, and accounting. Based on these reasons, we conclude that GLUE 2.0 contains knowledge that can be applied for accountable resource sharing in e-Science, and apply it as base model for our ontologies.

As an information management model, GLUE 2.0 considers coarse-grained constraints for group members to consume resources. However, such constraints cannot control with fine granularity how many resources each member per job or the whole group can consume. It neither considers constraints for commercial Cloud usage. To enable this fine-grained accounting capability for Cloud-based e-Science, we have extended GLUE 2.0, as presented in next section.

C. Overall Ontology Deployment Architecture

To integrate resource description with description of the roles of participants in an e-Science collaboration, four ontology files have been designed and implemented according to concepts defined by GLUE 2.0, which are named Base, Share, MappingPolicy and Service¹. The architecture of the developed ontologies is shown in Fig. 2.

The Share and Service ontologies model service supply contract terms and service registry terms respectively, while

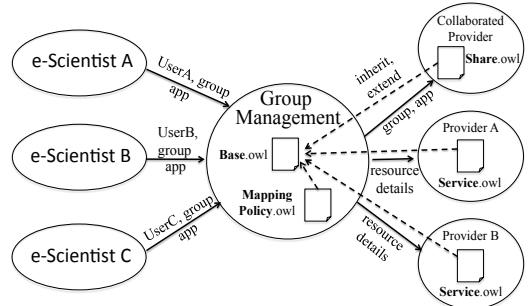


Fig. 2: Deployed ontologies architecture

the Group Manager with MappingPolicy model a management layer for access control and accounting. The entire information processing procedure imitates resource matchmaking in existing e-Infrastructures, namely information advertisement, collection, and processing [14].

Different from dynamic instances of Share, Service, and MappingPolicy, the Base ontology is more static, modelling the entities and their relations in e-Science collaboration and resource sharing. However, extensions can be achieved in Base ontology for interoperation with specific infrastructures, by defining relations with concepts specified by GLUE 2.0. The separation of ontology files according to different functional purposes permits autonomy, which facilitates entities with different roles to conduct their work in an independent manner.

D. Extended Model in Layered Ontologies

1) Accountable Properties Extension in Base Ontology: We construct a class framework for entities and properties to represent conceptual models and entities in GLUE 2.0 [15]. To facilitate fine-grained accountable resource sharing, classes and properties are extended by: considering properties applied in Clouds and Cloud-based Grids [16]; analysing the relations between job types and corresponding accountable attributes; and re-using accounting attributes specified in Usage Record [17], which is a widely applied Open Grid Forum recommendation to track resource usage. The extended properties for accounting purpose include:

- *cpuTime*: remaining CPU time for a research group in an execution environment in Share or for a single user in a group in MappingPolicy.
- *balance*: remaining currency for a research group in an execution environment in Share or a single user in a group in MappingPolicy.
- *charge*: the cost of a service for per CPU hour in a Share instance or Service instance.
- *maxCost*: the cost limit set for a member, an execution environment, or an application execution per request.
- *maxTotalCost*: the cost limit for resources can be allocated to an application that includes more than one sub-application per request.

¹Applied ontologies: <https://figshare.com/s/71fede3576bd35ec3978>

- *paymentMethod*: the way to calculate consumption of resources, including *fixed* and *dynamic* in a Share instance or Service instance.
- *measurement*: the type of measurement used for quantifying the associated resource consumption, including *hour* and *second* in a Share instance or Service instance.
- *hasJobType*: the type of job, including *SingleJob*, *WorkflowJob*, *InteractiveJob*.

More properties can be added for usage of application, network service, storage service, etc.

2) *Class Extensions and Ontology Instances*: In our implementation, a Share instance represents an agreement between an e-Scientist or a research group and a provider. It describes agreed resource provisioning details, while the resources may be allocated according to access control policies contained in a corresponding MappingPolicy instance.

To illustrate the relations maintained in a collaboration, we extend Share with connection with *AdminDomain*, *UserDomain*, *ProviderDomain*, and *ApplicationEnvironment*. These relations describe the following case: e-Infrastructure A provisions computing resources for user B from group C to run application D, which requires E amount of F resources, where A, B, C, D, E, F represent information that can be reasoned over using ontologies. Accountable attributes' values in Share and MappingPolicy instances can be updated dynamically, by applying the ontology reasoning library's APIs. The combination of a Share and corresponding MappingPolicy instances can realise fine-grained accounting for a pool of shared resources. Share is for coarse-grained resource management for a group from provider's view, while MappingPolicy is for fine-grained resource management from a group manager's view. A Service instance includes available service details, which can be advertised by providers. Basically, accountable resource matchmaking and management are achieved by merging, updating, and reasoning distributed ontology files.

E. Resource Discovery via Ontology Reasoning

Two scenarios for matchmaking are categorised in our work: application-oriented matchmaking and resource-oriented matchmaking, as shown in Algorithm 1 and Algorithm 2 respectively. Application-oriented matchmaking assumes a customised environment for application execution has already been established [5][18]. In this case, only performance-related features are investigated, like execution finish time and cost of service. Resource-oriented matchmaking will search for satisfying resources based on a full package of requirements of an application, which may include memory size, CPU model, CPU speed, operation system, etc.

The designed matchmaking grants higher priority to providers who have previously collaborated than to new providers, assuming it may ensure good performance, especially when customised job execution settings are required. Accordingly, resource searching conducts application-oriented matchmaking previous to resource-oriented matchmaking.

1) *Ontology Resource Discovery for A Job*: As elements in a resource supply agreement are inherently interdependent, and

should be measurable and quantifiable for resource provisioning, we designed algorithms to evaluate and verify these elements. According to the scenarios we set up, different reasoning algorithms for application-oriented and resource-oriented matchmaking are applied, where resource-oriented matchmaking is activated after unsuccessful application-oriented matchmaking.

Algorithm 1 Application-Oriented Matchmaking

```

Input: user name, group name, app name
if requester is a member of the group then
    fetch requester's balance, default CPU number required by
    application, maxCost/maxTotalCpuTime set by manager
    if requester has enough balance, execution environments with
    at least equal CPU number as required, 1 hour's cost at most equal
    with maxCost then
        return execution environments with resource details
        end if
    end if

```

Algorithm 2 Resource-Oriented Matchmaking

```

if application-oriented matchmaking failed then
    fetch requester's balance, default CPU number required by
    application, maxCost/maxTotalCpuTime set by manager
    if requester has enough balance, execution environments with at
    least equal CPU number as required, 1 hour's cost at most equal
    with maxCost, required OS and CPU model with at least equal
    clock speed then
        return execution environments with resource details
        end if
    end if

```

2) *Resource Sharing Access Control*: Apart from resource reasoning, users' access privileges are considered by including an integer number representing different privilege levels. A larger number indicates higher privilege. A requester's privilege will only be applied when more than one user in the same group competes for the same resource during application-oriented matchmaking. If it fails, new infrastructures upon Service ontologies will be investigated.

These reasoning functions enable searching for resources applied to both single jobs and workflow jobs, where workflow jobs can be co-allocated in either a single domain or multiple distributed domains that satisfy sub-jobs' requirements.

III. EVALUATION

As GLUE 2.0 was not designed with commercial Clouds in mind, we demonstrate the generality of our ontology based on GLUE 2.0 by applying it to AWS as a resource provider. Since the values of properties contained in ontologies can be edited for specific instances, all the values applied are based on scenarios to model small scale research collaborations. Our definition of a small-scaled research group consists of around 5 to 30 members, so in our evaluation we assume that a group of 15 e-Scientists collaborate in a project, who use AWS for their experiments. To avoid a member in the group consuming an unreasonable amount of resources, we allow a group manager to set the maximum amount of resources per job by each

member with properties *maxCost* in a *MappingPolicy* instance. Being consistent with the payment method of AWS, services are measured in unit of hours specified by *measurement*. Both *balance* values for group in each instance defined in *Share* and for member defined in *MappingPolicy* will be checked if they are enough for at least 1 hour job execution during matchmaking.

As the test applies a simple application, which is not computationally intensive, the instance type used is t2.micro. Accordingly, service details are constructed for t2.micro as execution environment in a *Share* ontology². For demonstration purposes, *maxCost* set for group members for t2.micro is \$0.013, and instance type t2.small is also included in *Share* ontology with group *balance* as 0.

To evaluate resource-oriented matchmaking, we assume that users require more CPUs than the internal instances of the collaboration can supply. The instance types applied for this case are t2.medium, t2.large, m4.xlarge, m4.2xlarge in a Service ontology. The program searches for services in details: operation system (Linux), CPU model (Intel Xeon), CPU speed at least 3.3GHz, and at least 2 CPUs.

Overall, the scenarios established in AWS for ontology-based resource matchmaking include:

- Scenario 1: Application-oriented matchmaking with *physicalCpus*. It is activated with application name submitted by requesters. The number of CPU required is at least 1, and job execution is managed by *maxCost*.
- Scenario 2: Access control by *balance* of AWS instance. The balance for the group in the satisfying instance is not sufficient to execute the job, while other conditions are met.
- Scenario 3: Access control by *balance* of group member. When submitting a job, requester does not have enough balance to run the job, while other conditions are met.
- Scenario 4: User access control by privileges identified with *level*. When group members with different privileges try to access the same instance, the request from member with the highest privilege will be accepted, while the ones with lower privileges will be declined.
- Scenario 5: Resource-oriented matchmaking with *physicalCpus* specified by requester. During matchmaking, requester's balance will be checked for job execution for at least 1 hour.

The reasoning capability for each property is tested by controlling other properties' values for corresponding scenarios as described. The results, as shown in Table I for each scenario (denoted by S), are delivered by ontology reasoning realised by Java programs. They illustrate that the ontologies and reasoning functions facilitate application-oriented and resource-oriented matchmaking based on requester's demands, research group's expense control and requester's privileges.

The pure runtime reasoning performance (i.e. excluding network delays) of the deployed ontologies is evaluated for scenarios 1, 2, 3, and 5. The performance of scenario 4

²AWS instances' details: <https://aws.amazon.com/ec2/pricing/>

TABLE I: AWS JOB ONTOLOGY MATCHMAKING EVALUATION

S	Result
1	Instance t2.micro is returned with service details. Requester's balances in <i>MappingPolicy</i> ontology for requester and group's balance in <i>Share</i> ontology for instance t2.micro are updated when requester stops job execution or job execution approaches 1 hour. In the latter case, stop command is sent to running instance.
2	No satisfying instance is returned.
3	No satisfying instance is returned.
4	Assuming UserA and UserB send requests at the same time and only 1 instance can meet demands, while other conditions are met. Supposing UserA has higher privilege UserB, UserA is returned with service details of instance t2.micro, and UserB is informed no satisfying service is found.
5	Assuming requester demands 2 CPUs, and has balance as \$0.110. Instance's details of t2.medium and t2.large are returned.

TABLE II: AWS JOB MATCHMAKING PERFORMANCE

Scenario	1	2	3	5
Mean (ms)	267.10	279.33	272.93	280.82
Deviation (ms)	74.45	74.37	72.54	59.11

is not evaluated as manual procedures required introduce uncontrollable factors and makes it difficult to evaluate pure reasoning performance.

To measure only the speed of reasoning, ignoring network delays, all ontology files are deployed locally. The evaluation is conducted on Mac OS X with 2.8GHz Intel Core i7 and 4GB memory. We calculate mean values for duration of reasoning and the standard deviation of the data collected, shown in Table II as Mean and Deviation respectively.

Since the processing time of the matchmaking is only a fraction of a second, our methods can be applied where the job duration is anything over a second; many Grid or Cloud jobs take orders of magnitude more. It also indicates that extension for more accountable properties would not decrease performance, as increasing reasoned elements do not increase reasoning time significantly, as can be seen by comparing performance of Scenario 5 and other scenarios. In a real distributed system network delays would need to be considered but these would affect also the job return times, so the reasoner would still have minimal impact.

IV. CONCLUSION

We propose a conceptual semantic model based on widely applied information model GLUE 2.0 extended with terms describing accountable characteristics for resource sharing across different models (e.g Grids of Clouds). Exploiting the inheritable and reasoning capabilities of ontologies, we demonstrate that it meets scenarios involving dynamic, automatic, and distributed accountable resource matchmaking and sharing for e-Science collaborations.

We have applied the ontology model presented to increasingly flexible resource supply via the negotiation protocol proposed in [8]. This can enable accountable resource sharing facilitated by ontology reasoning, while job submission and

management can be conducted by existing application management tools for example, Application Hosting Environment AHE3 [18].

REFERENCES

- [1] *E-Infrastructure - Research Councils UK.* Web. Retrieved from: <http://www.rcuk.ac.uk/research/xrcprogrammes/otherprogs/einfrastructure/>. 16 Mar. 2016
- [2] N. Sadashiv. and S.D. Kumar, *Cluster, grid and cloud computing: A detailed comparison.* Computer Science & Education (ICCSE), 2011 6th International Conference on e-Science, IEEE, 2011.
- [3] M. Riedel, L. Erwin, T. Sodemann, L. Field, J.P. Navarro, J. Casey, M. Litmaath et al. *Interoperation of world-wide production e-Science infrastructures.* Concurrency and Computation: Practice and Experience 21, no. 8, pp. 961-990, 2009.
- [4] H. Yoo, C. Hur, S. Kim, and Y. Kim, *Y. An ontology-based resource selection service on science cloud.* In Grid and Distributed Computing, pp. 221-228. Springer Berlin Heidelberg, 2009.
- [5] Somasundaram, S. Thamarai, K. Govindarajan, U. Kiruthika, and R. Buyya. *Semantic-enabled CARE Resource Broker (SeCRB) for managing grid and cloud environment.* The Journal of Supercomputing 68, no. 2, pp. 509-556, 2014.
- [6] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, A. Frohner, A. Gianoli, K. Lorentey, F. Spataro. *VOMS, an authorization system for virtual organizations.* In Grid computing, pp. 33-40. Springer Berlin Heidelberg, 2004.
- [7] J.M. Brooke, and M.S. Parkin, *Enabling scientific collaboration on the Grid.* Future Generation Computer Systems. 26(3), pp. 521-30, 2010.
- [8] Z. Meng, and J. Brooke. *Negotiation Protocol for Agile and Reliable E-Science Collaboration.* In e-Science (e-Science), 2015 IEEE 11th International Conference on, pp. 292-295. IEEE, 2015.
- [9] R. Studer, V.R. Benjamins, D. Fensel. *Knowledge engineering: principles and methods.* Data & knowledge engineering. 25(1), pp. 161-97, 1998
- [10] J.M. Pérez, J.B. Bernabè, J.M. Calero, F.J. Clemente, G.M. Pérez, A.F. Skarmeta. *Semantic-based authorization architecture for grid.* Future Generation Computer Systems, 27(1), pp. 40-55, 2011.
- [11] M. Hartung, F. Loebe, H. Herre, E. Rahm. *Management of evolving semantic grid metadata within a collaborative platform.* Information Sciences, 180(10), pp. 1837-49, 2010.
- [12] G.A. Vouros, A. Papasalouros, K. Tzonas, A. Valarakos, K. Kotis, J.A. Quiané-Ruiz, P. Lamarre, P. Valduriez. *A semantic information system for services and traded resources in Grid e-markets.* Future Generation Computer Systems, 26(7), pp. 916-33, 2010.
- [13] S. Andreozzi, S. Burke, F. Ehm, L. Field, G. Galang, B. Konya, M. Litmaath, P. Millar, J.P. Navarro. *GLUE Specification v. 2.0.* In Open Grid Forum Recommendation Documents, Open Grid Forum, 2009.
- [14] R. Raman, M. Livny, M. Solomon. *Matchmaking: Distributed resource management for high throughput computing.* In HPDC, IEEE, 1998.
- [15] W. Xing, M.D. Dikaiakos, R. Sakellariou. *A core grid ontology for the semantic grid.* In Cluster Computing and the Grid, 2006, CCGRID 06, Sixth IEEE International Symposium, Vol. 1, pp. 178-184, IEEE, 2006.
- [16] S. Pullinger, J. Gordon, D. Lezzi, B. Parak. *Federated Cloud Accounting.* Retrieved from: https://wiki.egi.eu/wiki/Federated_Cloud_Accounting, EGI. Web, 17 July 2016.
- [17] R. Mach, S. Jackson, L. McGinnis, and Pittsburgh Supercomputing Center. *Usage Record-Format Recommendation.* Open Grid Forum, 2006.
- [18] S.J. Zasada, D.C. Chang, A.N. Haidar, P.V. Coveney. *Flexible composition and execution of large scale applications on distributed e-infrastructures.* Journal of Computational Science. 5(1), pp. 51-62, 2014.

Apache Airavata Security Manager: Authentication and Authorization Implementations for a Multi-Tenant eScience Framework

Supun Nakandala
 Pervasive Technology Institute
 Indiana University, USA
 snakanda@iu.edu

Hasini Gunasinghe
 Computer Science
 Purdue University, USA
 huralai@purdue.edu

Suresh Marru
 Pervasive Technology Institute
 Indiana University, USA
 smarru@iu.edu

Marlon Pierce
 Pervasive Technology Institute
 Indiana University, USA
 marpierc@iu.edu

Abstract—eScience middleware frameworks integrating multiple virtual organizations must incorporate comprehensive user identity and access management solutions. In this paper we examine usage patterns for these systems and map the patterns to widely used security standards and approaches. We focus on science gateways, a class of distributed system cyberinfrastructure. Science gateways are end user environments that provide access to a wide range of academic and commercial computing and storage resources for virtual organizations. Successful gateways focus on specific scientific communities and domains, but they build on many reusable features that can be provided by general purpose hosted platform services that can support multiple tenants. Providing a security framework for identity and access management for such hosted service removes the burden for each gateway to handle its user identity management and control access to its critical resources. From the resource provider's point of view, it provides a basis for more uniform accounting and auditing. Challenges arise from the range of gateways (both legacy and newly created), the range of technologies used to build them, and the range of end user environments (Web, mobile, desktop, and programmatic API clients) that gateways provide. Using Apache Airavata as an implementation, we examine three common gateway types based on where the user identity information is held and how these can be treated in a unified manner using OAuth2 and OpenID-Connect. Our solutions for identity and access management are not specific to Apache Airavata but can be generally applied to any e-Science platform.

Index Terms—science gateways, identity management, distributed systems security

I. INTRODUCTION

Science Gateways [1] [2] [3] [4] are user environments and supporting services that help researchers make effective and enhanced use of a diverse set of computing, storage, and related resources. Gateways serve as Virtual Organizations, brokering access for their users to a broad collection of resources from different, often unaffiliated resource providers including campus grids, national scale cyberinfrastructure, and commercial cloud vendors. Thus, identity management, authentication, and authorization are critical capabilities that a gateway must provide.

Security management is one example of the general-purpose features that underlie many domain specific gateways. An important architectural trend is for gateways serving specific

Virtual Organizations to use hosted, general purpose platform services. The goal of the Apache Airavata project [5] [6] is to implement and integrate many of these general purpose services into a common framework that can be run as a multi-tenanted platform service that can support multiple gateways simultaneously. In Apache Airavata, these services are implemented by multiple components (see [5]) but are exposed through a common API and a single, logical connection point, the API Server. Here, we describe the design and implementation needed to secure the interactions between a gateway tenant and the Apache Airavata API Server.

Science gateway tenants to Apache Airavata services run remotely and are typically under the control of the gateway provider. Gateway clients interact with Apache Airavata through an Apache Thrift-based API. For an overview of the API, see [6]. The Thrift-defined API and data models allow Airavata to provide client SDKs in many programming languages, including Java, PHP, Python and C++. These SDKs manage over-the-wire communications. Our technical challenge is to implement security features over the top of these communications. We note an additional challenge: gateways are not only Web-based but may also be desktop or native mobile applications. They may also be embeddable clients intended to be called directly from end-user scripts and notebooks. This introduces a challenge since Airavata SDKs are distributed directly to the end user's device. We cannot assume access is restricted behind a Web server.

In this paper, we discuss the design and implementation of the solution for securing the Airavata API. This includes authenticating and authorizing end users into the Airavata API, based on different use cases that depend on the different types of user identity management scenarios in gateways. The primary contributions of this paper are identifying general patterns for gateway identity management, mapping these patterns to OAuth2 scenarios, and implementing these solutions. We examine the implementation using three Apache Airavata gateway clients.

We first identify three scenarios in which a gateway tenant interacting with Airavata middleware may manage its users.

- Scenario 1: The gateway client does not have a user

store and will use Airavata infrastructure to provide user management.

- Scenario 2: The gateway has a user store and in-house identity management mechanisms. In this scenario, different gateways have different preferences on the level at which they share user identity information. This is typical of mature gateways.
- Scenario 3: The gateway authenticates users into the gateway using a federated identity provider such as InCommon using mechanisms such as SAML SSO, OpenID, and OAuth.

We must be able to provide a unified identity management solution that can meet the requirements of the above scenarios and provide proper Airavata API security that can be seamlessly adopted by all types of gateways including web based and native (desktop and mobile) clients.

II. SOLUTION OVERVIEW

When devising a solution, one option that we considered was to provide user authentication at the gateway layer using the authentication protocol of the gateway's choice, and use the system-to-system authentication mechanism between the gateway and the Airavata server. Examples of two of the system-to-system authentication mechanisms that can be used are mutual authentication using SSL certificates and basic authentication (i.e: Gateway credentials over TLS). The main drawback of the certificate-based mutual authentication approach is that it is not scalable from the management point of view. Each gateway can have different types of applications (web, native), and ideally each of these applications should have different credentials. When the number of gateway clients increases, management of gateway credentials and PKI infrastructure is not scalable [7]. Also, both of these approaches can be used only for web based gateways, where credentials or certificates can be securely maintained in a web server. They are not secure enough to be used in native clients as a malicious user can reverse engineer the client and gain access to the credentials/keys. One way to facilitate native clients using these approaches is to route the requests from native clients via a proxy server and establish the mutual trust between the proxy server and Airavata using gateway credentials or certificates. This requires extra effort from gateway developers and requires end user information to be passed to the Airavata API Server by the clients with every API request.

The solution we adopted is to use OAuth 2.0 based authorization delegation [8] to the gateway by the user authenticated at the gateway. The main advantage of this approach is scalability. This approach does not require any management of gateway credentials or PKI infrastructure. The OAuth access tokens can be generated by a separate dedicated authorization server and not by the API Server. This approach also benefits from wider adoption in the general "Software as a Service" and "Platform as a Service" communities to which science gateways and gateway frameworks belong. OAuth 2.0 is the de-facto standard for access delegation. It can be used in conjunction with existing authentication protocols such as

SAML 2.0 based single sign on via the OAuth 2.0 extension profiles. Even though OAuth is an authorization delegation protocol, authentication of users can be done using OpenID-Connect [9]. Hence OpenID-Connect + OAuth 2.0 can be used as a comprehensive authentication and authorization protocol.

Figure 1 illustrates the high level architecture of the solution with mappings to the standard OAuth 2.0 based authorization-delegation solution architecture. Because OAuth 2.0 and related technologies are widely adopted, we chose to integrate a third party implementation rather than develop it ourselves. Our solution makes use of the identity management features offered by WSO2 Identity Server (IS) [10]. WSO2 IS is a widely used open source (Apache V2 License) identity management system that provides out-of-the-box support for many identity management standards such as OAuth/OpenID, SAML SSO, XACML, and features such as multi-factor authentication, password policies, etc. It supports multi-tenancy, per-tenant user store configuration, and custom pluggable user store manager extensions, all of which are relevant to the problems discussed in this paper.

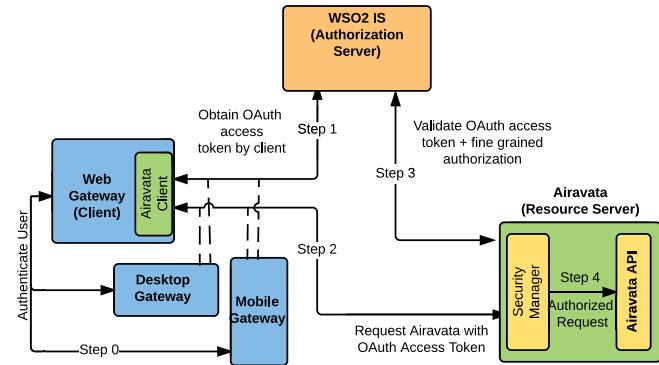


Fig. 1. High level overview of the solution

Details of the interactions illustrated in Figure 1 are as follows.

- 0) User is authenticated to Airavata. This paper examines three specific scenarios.
- 1) OAuth token is obtained from the WSO2 IS to access the Airavata API on behalf of the authenticated user.
- 2) Requests to Airavata, depending on the actions the user wants to perform, are sent along with the obtained OAuth token.
- 3) Each request sent to the Airavata API Server is authorized by the Security Manager. The Security Manager acts as a client within Airavata to various WSO2 IS and other potentially services. First the attached OAuth token is validated and user's profile information is retrieved. Then the request is authorized; authorization approaches include role-based or group based decisions.
- 4) Only the authorized requests are allowed to reach the Airavata API.

Authorization decisions for subsequent requests from the same user are handled using caching to avoid contacting WSO2

IS for every call. This step can use different authorization mechanisms, such as role determination using XACML capabilities of IS or group membership via Grouper. The above solution supports multi-tenancy; when the user identity is exchanged, it includes the tenant domain that the user belongs to so that the authentication and authorization are performed with respect to that tenant domain in WSO2 IS. The same client side logic to authenticate a user and obtain an OAuth token can be implemented in web, desktop and mobile clients using the recommended grant types available in the OAuth 2.0 specification.

III. SOLUTION IN DETAIL

A. Authenticating & Obtaining an OAuth 2.0 Access Token

User authentication and obtaining an access token by the client application (Step 0 and Step 1 in Figure 1) are the only steps in the high level solution that differ among the three scenarios that we identified. OAuth 2.0 specifies five grant types to obtain an access token by the client app. Each of these grant types serves a different purpose and is used in a different way. We evaluate each and discuss their applicability to our gateway client scenarios.

The Authorization Code grant type is the recommended grant type to be used if the client application is capable of spawning a web browser to redirect the user to the authorization server's authentication page. The user signs in to the authorization server and authorizes the gateway application to obtain an access token for the user. This authorization is a one-time-only application authorization. If the authorization server authorizes the request, the user will be redirected back to the client with a token (called the authorization code) in the query string, which the client application will capture and exchange for an access token in the background. In order to use this grant type, the application itself should already be trusted by the authorization server through some application registration process. This is applicable for standard Web-based gateway tenants.

The Implicit grant type is very similar to the Authentication Code grant type. The difference is that the access token is directly returned to the client application after the user signs in for Implicit grants. Implicit grants are for clients that are not capable of keeping the client application's own credentials secret. An example scenario is a thick browser client that uses JavaScript to directly access the Apache Airavata API methods rather than going through an intermediate Web server.

If the user (resource owner) already trusts the client application to provide his/her own credentials, the Authorization Code grant type is not needed. Resource Owner Password grant type is an alternative that can be used to simplify the flow in this case. This grant type is useful in scenarios where the client application cannot spawn a web browser; examples include desktop and mobile client applications. In this grant type instead of redirecting the user to the authorization server's authentication page, the client application itself obtains the user credentials and then sends these to the authorization server

along with the client's own credentials. If the authentication is successful then the client will be issued an access token.

The Client Credential grant type is similar to the Resource Owner Password grant except the client application's credentials are used to authenticate a request for an access token. This grant type should only be used by trusted clients. This grant is suitable for machine-to-machine authentication that involves no user interaction or user authorization. In this grant type we do not need to use OpenID-Connect to authenticate users.

Authorization servers that support Refresh Code grant types are able to issue a refresh token when they return an access token to a client application. When the access token expires, the client can use the refresh token to retrieve a new access token with the same permissions. The client has to maintain the state of each token. This can be done by either periodically using the active refresh token or by using the refresh token to acquire a new token after an access failure.

B. Science Gateway Client Scenarios

We now map the general OAuth2 solutions to three different gateway identity management scenarios. All solutions use the Security Manager component within Apache Airavata (see Figure 1) that acts as a client to various WSO2 IS services.

1) The gateway doesn't have a user store and would like to depend on Airavata to provide user management features: The gateway application makes use of the Airavata Client SDK to create users, which in turn invokes the User Admin API of the WSO2 IS; see Figure 2. When users are authenticated to the gateway, their credentials are validated against those stored in the WSO2 IS user store. The gateway uses Airavata's client SDK to authenticate users and obtain OAuth access tokens using OpenID-Connect. All the client applications are provided by the gateway itself and no user generated or third party applications need to connect to Airavata. Thus all gateway client applications can be considered as trusted clients, and therefore instead of Authorization Code grant type we use the Resource Owner Password grant type to obtain access tokens for both web and native clients. If the web application is a browser application that directly interacts with the Airavata API through a JavaScript SDK, the Implicit grant type should be used instead and privileges of access token obtained by this grant type should be highly restricted.

2) The gateway has its own user-store and identity management mechanisms: Here we consider three solution categories to address this scenario based on the differences in preference of the gateway to share user identity information with Airavata. In the first category, the gateway will not share any information about the end user's identity and will use Airavata only as a job execution engine. Therefore the gateway client will obtain an OAuth access token by authenticating to the WSO2 IS. The Client Credential grant type is the appropriate type for this case. After retrieving an access token, end user requests to Airavata attach this token (see Figure 3). In this case the types of gateway applications that can be supported are limited to web based gateways where the client credentials can be securely maintained in the web server. For native (JavaScript

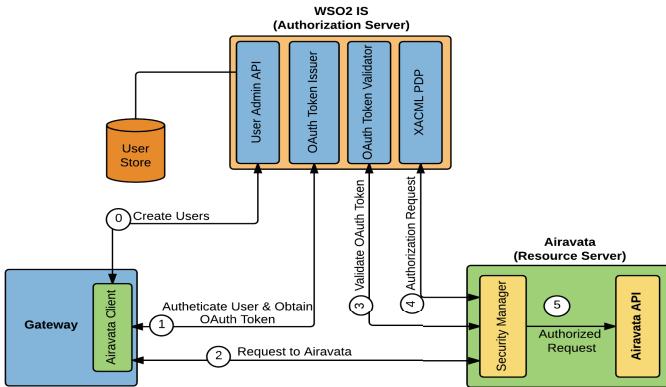


Fig. 2. Gateway uses Airavata-provided user store

and desktop) client gateways that fall in this category, the requests should be sent through an intermediary proxy server that can securely maintain the client credentials.

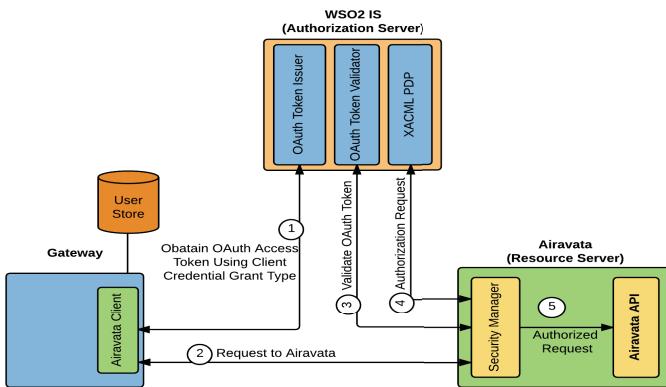


Fig. 3. Gateway does not share any user identity information with Airavata

In the second category, the gateway shares user identity information but does not allow Airavata to connect to the gateway's user store. User accounts need to be provisioned with the identity information required by Airavata. In this case the gateway uses Airavata's identity provisioning SDK client to provision user accounts to Airavata's identity manager (i.e WSO2 IS) at the time of user creation in the gateway. Already created user accounts in the gateway user store can be provisioned through a bootstrapping phase. Invoking the Airavata SDK for user provisioning will in turn invoke the System for Cross-domain Identity Management (SCIM) [11] endpoint in WSO2 IS for identity provisioning. This enables the execution flow of accessing the Airavata API to be the same as in Scenario 1; see Figure 4. In this category user information will be duplicated in both the gateway user store and the WSO2 IS user store. It is the responsibility of the gateway to maintain them in coherence.

In the third category, the gateway allows Airavata to connect to its organizational user store in read-only mode. In this case, the identity manager of Airavata (i.e WSO2 IS), is connected to the gateway's organizational user store through the user store manager extension provided by the WSO2 IS. This

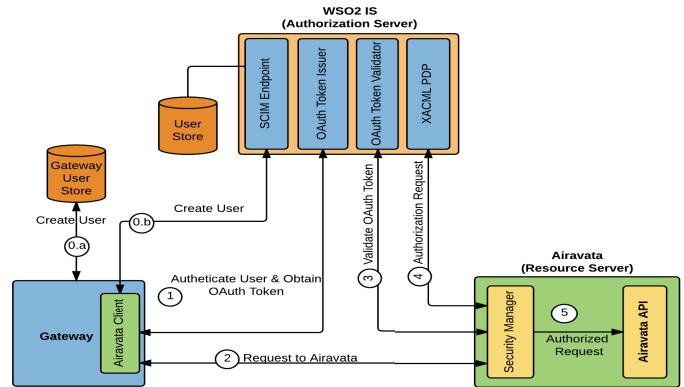


Fig. 4. Gateway provisions user accounts to the Airavata provided user store

enables the user authentication, access token retrieval, and the rest of the execution flow for accessing the Airavata API to be the same as in scenario 1; see Figure 5. The WSO2 IS distribution by default provides user store manager extensions that can be used to integrate Active Directory and LDAP based user stores. Custom user store manager extensions can be written to integrate any custom user store, such as a database.

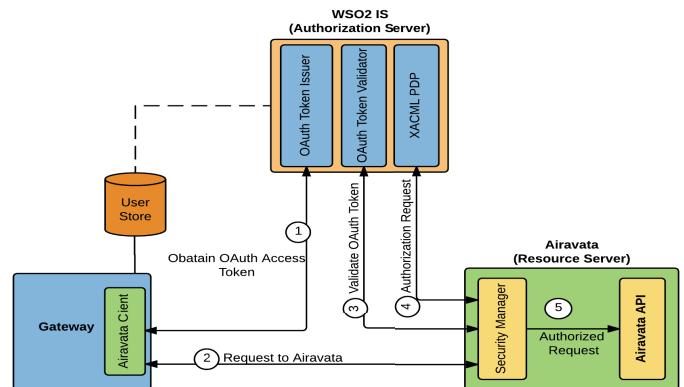


Fig. 5. Gateway allows Airavata to have read only access to the organizational user store

3) The science gateway does not have a user store but instead authenticates users into the gateway using a third party federated identity provider: In this scenario the gateway becomes a relying party, and it needs to authenticate users to Airavata through the federated authentication mechanism that is being used in the gateway community. To solve this issue, we used the inbound authentication configuration feature in WSO2 IS. Using this feature, it is possible to plug external federated authentication providers as inbound authentication providers to a user store. Out of the box, WSO2 IS supports OpenID, SAML, OAuth2/OpenID-Connect, WS Federation, Google, Yahoo, Microsoft and Facebook federated authentication providers. For a federated authentication protocol not in the previous list, it is possible to plug in a custom inbound authentication provider.

If the federated authentication protocol supports retrieval of the user's identity attributes from the identity provider (such as

SAML, OpenID), a user account is created in WSO2 IS with this identity information using just-in-time provisioning. When a user tries to authenticate to WSO2 IS, the user can select the configured identity provider and login to that provider. Once the user is authenticated via the federated identity management protocol, an OAuth access token is generated in WSO2 IS that has access to the Airavata API, and the rest of the flow of execution continues in the same way as in other cases; see Figure 6. Since federated authenticators typically support only web-based access, a web based OAuth 2.0 grant type such as Authorization Code grant is appropriate. In the case of thick web clients, Implicit grant type should be used. Native clients are now required to spawn an embedded browser for user authentication and extract the access token.

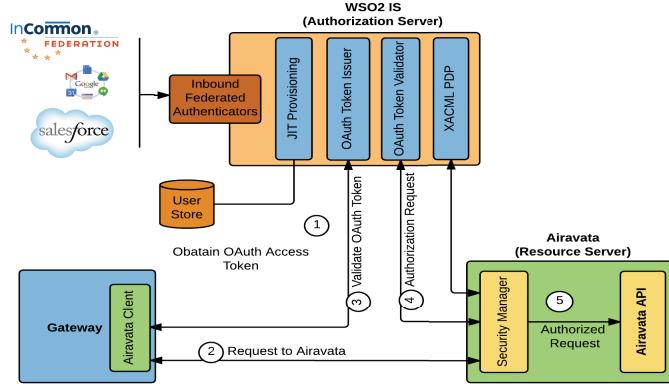


Fig. 6. Gateway uses federated identity provider.

IV. IMPLEMENTATION AND EVALUATION

Every Apache Airavata API method definition was changed to incorporate a mandatory field named `AuthzToken`, which contains the OAuth access token and the Airavata tenant ID information that is set at the gateway client layer. To process this token, we added the new Security Manager component to the Airavata API Server. This new component is designed to be pluggable so that we can change it if required without affecting the rest of Airavata. The Security Manager intercepts all API requests and validates them from the separately hosted WSO2 Identity Server, which acts as the Authorization Server for Airavata. For intercepting the API requests the Security Manager uses annotations supported by the Google Guice library.

Authorization validation at the Security Manager is a two step process. First the OAuth access token is validated by calling the OAuth token validator endpoint in WSO2 IS, and then any authorization policy is enforced. When invoking these endpoints, the requests need to be authenticated using HTTP basic authentication by providing the respective tenant's admin credentials in WSO2 IS. Therefore, Airavata needs to know and store the respective credentials for IS tenants. For this we used Airavata's credential store server [12]. Gateway administrators can store their IS tenant credentials by updating the `GatewayProfile` object in Airavata.

Ideally, every Airavata API request should be validated against the WSO2 IS as shown in Step 3 in Figure 1. However, we found that this imposes a significant performance overhead (about 350 ms) on the Airavata API method latency. To overcome this issue we added an authorization cache module into the Security Manager that caches the authorization decisions for each access token-tenant-API method combination after successful validation from WSO2 IS. Subsequent API requests to the API Server from the same user are checked in the authorization cache instead of directly invoking the IS validation endpoints. The caching duration of each authorization decision is set to the remaining valid duration of the corresponding OAuth access token, which can be obtained from the OAuth token validation response. This OAuth token validation response contains various user attributes such as username and email that are also cached.

We have implemented the designs described here to support several science gateway tenants to Apache Airavata. The SEAGrid science gateway [13] is an example of a Scenario 1 tenant. SEAGrid uses a MySQL database as the user store in WSO2 IS and provides a web based gateway and a desktop client. Both of these clients use the Resource Owner Password grant type in OAuth 2.0 specification to authenticate users and retrieve access tokens as both are trusted clients. Another client is a USDA-funded science gateway, currently under development by a collaborating team, for accessing bioinformatics applications. This client needs to integrate the user management with the existing user store. Thus this falls to the Category 3 of Scenario 2 in our taxonomy. The USDA user store is an LDAP server, and hence it was straightforward to integrate with IS. This gateway is web-based, so a Resource Owner Password grant type is used for authentication and access token retrieval. The UltraScan science gateway [14] provides its own user store and does not share any user information with Airavata. Thus, it is a Category 1, Scenario 2 gateway in our taxonomy. As described in the solution section, the Client Credential grant type is used to retrieve an access token that is used to submit API requests to Airavata.

V. RELATED WORK

Science gateway security, especially the aspect of user credential management for individual gateways, is a well studied area. One approach is to support end-to-end single sign on for users. Whenever a gateway user submits a job to a remote compute resource, the gateway middleware uses that particular user's credentials on the remote host for authentication. Per user grid certificates, MyProxy and various MyProxy services such as OAuth for MyProxy [15], CILogon for MyProxy, and the MyProxy Gateway are implementations of this approach [16]. Security Assertion Markup Language based Single-Sign-On (SAML SSO) is used by the MoSGrid science [17].

The per-user credential mechanism can be difficult to operate in practice. The Authentication, Authorization, Auditing and Accounting (AAAA) model [18] is an alternative; XSEDE uses a simplified version of this model in production. The main

idea of the AAAA model is to adopt community user accounts and avoid using per user credentials to authenticate to the remote resources. This significantly reduces the effort required for user management tasks in compute resources as now there are a limited number of science gateway community accounts. This approach outsources gateway user management tasks to the gateway layer itself. In this model, multiple gateway users share the same community account to submit jobs to compute resources. Within Airavata, community credentials are managed by the credential store component [12].

Globus Nexus [19] is very similar to WSO2 IS. It is a “platform as a service” application that provides user identity management, profile management, and group management features. Its identity management capabilities allow users to create a unique Globus identity that can be associated with federated external identities from campus identity providers (e.g CILogon), computing resource identity providers (XSEDE accounts using MyProxy OAuth), and commercial identity providers such as Google. This Globus Identity can be consumed by subscribing applications using an OAuth-based workflow to create a Single-Sign-On environment. Globus Nexus is a hosted service, whereas WSO2 IS is downloadable, open source software that is also available as a for-fee service. This difference introduces tradeoffs that gateways and gateway platform service providers should consider.

VI. CONCLUSIONS AND FUTURE WORK

This paper examines the over-the-wire access patterns that exist between a wide range of gateway clients and multi-tenanted platform services like Apache Airavata and how they can be mapped to OAuth2 protocol and OpenID-Connect. This paper does not cover fine-grained authorization decisions. These can be implemented either as roles or as groups; our current implementation uses XACML role expressions within WSO2 IS to encode and enforce roles associated with different API calls. We are currently comparing this approach with group-based approaches that can be implemented using open source Grouper software.

ACKNOWLEDGMENT

This work was supported by NSF award #1339774 “Collaborative Research: SI2-SSI: Open Gateway Computing Environments Science Gateways Platform as a Service (OGCE SciGaP)”. S. N. and H. G. were supported by Google Summer of Code. The Center for Trustworthy Scientific Cyberinfrastructure (NSF awards #1234408 and #1547272) consultations are summarized at <http://trustedci.org/scigap/>.

REFERENCES

- [1] K. A. Lawrence, M. Zentner, N. Wilkins-Diehr, J. A. Wernert, M. Pierce, S. Marru, and S. Michael, “Science gateways today and tomorrow: positive perspectives of nearly 5000 members of the research community,” *Concurrency and Computation: Practice and Experience*, vol. 27, no. 16, pp. 4252–4268, 2015.
- [2] S. Gesing and N. Wilkins-Diehr, “Science gateway workshops 2014 special issue conference publications,” *Concurrency and Computation: Practice and Experience*, vol. 27, no. 16, pp. 4247–4251, 2015.
- [3] N. Wilkins-Diehr, S. Gesing, and T. Kiss, “Science gateway workshops 2013 special issue conference publications,” *Concurrency and Computation: Practice and Experience*, vol. 27, no. 2, pp. 253–257, 2015.
- [4] P. Kacsuk, *Science Gateways for Distributed Computing Infrastructures*. Springer, 2014.
- [5] S. Marru, M. Pierce, S. Pamidighantam, and C. Wimalasena, “Apache airavata as a laboratory: architecture and case study for component-based gateway middleware,” in *Proceedings of the 1st Workshop on The Science of Cyberinfrastructure: Research, Experience, Applications and Models*. ACM, 2015, pp. 19–26.
- [6] M. Pierce, S. Marru, B. Demeler, R. Singh, and G. Gorbet, “The apache airavata application programming interface: overview and evaluation with the ultrascan science gateway,” in *Proceedings of the 9th Gateway Computing Environments Workshop*. IEEE Press, 2014, pp. 25–29.
- [7] P. Gutmann, “Pki: it’s not dead, just resting,” *Computer*, vol. 35, no. 8, pp. 41–49, 2002.
- [8] D. Hardt, “The oauth 2.0 authorization framework,” 2012.
- [9] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore, “Openid connect core 1.0,” *The OpenID Foundation*, p. S3, 2014.
- [10] W. Inc, “Wso2 identity server: The first enterprise identity bus,” <http://wso2.com/products/identity-server/>.
- [11] K. Grizzle, E. Wahlstrom, C. Mortimore, and P. Hunt, “System for cross-domain identity management: Core schema,” *System*, 2015.
- [12] T. A. Kanewala, S. Marru, J. Basney, and M. Pierce, “A credential store for multi-tenant science gateways,” in *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*. IEEE, 2014, pp. 445–454.
- [13] S. Pamidighantam, S. Nakandala, E. Abeysinghe, C. Wimalasena, S. Rathnayaka, S. Marru, and M. Pierce, “Community science exemplars in seagrid science gateway: Apache airavata based implementation of advanced infrastructure,” *Procedia Computer Science*, vol. 80, pp. 1927–1939, 2016.
- [14] B. Demeler and G. E. Gorbet, “Analytical ultracentrifugation data analysis with ultrascan-iii,” in *Analytical Ultracentrifugation*. Springer, 2016, pp. 119–143.
- [15] J. Basney, R. Dooley, J. Gaynor, S. Marru, and M. Pierce, “Distributed web security for science gateways,” in *Proceedings of the 2011 ACM workshop on Gateway computing environments*. ACM, 2011, pp. 13–20.
- [16] J. Basney, J. Gaynor, S. Marru, M. Pierce, T. A. Kanewala, R. Dooley, and J. Stubbs, “Integrating science gateways with xsede security: A survey of credential management approaches,” in *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*. ACM, 2014, p. 58.
- [17] S. Gesing, R. Grunzke, J. Krüger, G. Birkenheuer, M. Wewior, P. Schäfer, B. Schuller, J. Schuster, S. Herres-Pawlis, S. Breuers *et al.*, “A single sign-on infrastructure for science gateways on a use case for structural bioinformatics,” *Journal of Grid Computing*, vol. 10, no. 4, pp. 769–790, 2012.
- [18] J. Basney, V. Welch, and N. Wilkins-Diehr, “Teragrid science gateway aaaa model: implementation and lessons learned,” in *Proceedings of the 2010 TeraGrid Conference*. ACM, 2010, p. 2.
- [19] K. Chard, M. Lidman, B. McCollam, J. Bryan, R. Ananthakrishnan, S. Tuecke, and I. Foster, “Globus nexus: A platform-as-a-service provider of research identity, profile, and group management,” *Future Generation Computer Systems*, vol. 56, pp. 571–583, 2016.

Privacy-protected Social Media User Trajectories Calibration

Shuo Wang
 Computing and Information Systems
 The University of Melbourne
 Melbourne, Australia
 Email: shuow4@student.unimelb.edu.au

Richard Sinnott
 Computing and Information Systems
 The University of Melbourne
 Melbourne, Australia
 Email: rsinnott@unimelb.edu.au

Surya Nepal
 Data61
 CSIRO
 Sydney, Australia
 Email: Surya.Nepal@csiro.au

Abstract—Advanced data analytics have become an integral part of a number of eScience initiatives including the many challenges facing the urban sciences. Understanding the movement of people and their spatial trajectories would greatly aid the development of policies for sustainable urban living including urban traffic analysis and smart city management. Due to the widespread popularity of mobile devices with location-aware capabilities and the extensive prevalence of location-based social networks, citizens' movements and their trajectories are being produced and gathered at an unprecedented rate. In this context, there are two fundamental issues that need to be addressed: trajectory data hold private information of citizens that require privacy preserving solutions for data release and analysis, and the heterogeneous nature of the trajectory data makes it hard to effectively measure their similarity, which is fundamental to trajectory analysis. In this paper, we address these challenges by proposing an innovative private trajectories calibration model that not only guarantees the privacy of citizens, but also increases the utility. We have conducted comprehensive experiments using real-life user trajectories extracted from Twitter data. The results reveal the effectiveness and efficiency of the proposed approach, which is also reported in this paper.

I. INTRODUCTION

eScience typically includes the support for development of innovative infrastructures that assists researchers from multi-disciplinary fields to cooperate with each other by providing improved research capabilities. This can include the development of novel computational platforms offering data capture and advanced data analysis[1]. eScience aims to shorten the gap between the requirements for data-intensive scientific research through up-to-date data analytics solutions. Data-driven science now makes possible the concept of smart "everything", e.g. smart farms in agriculture, smart health in health science and smart cities for urban planning. Smart city applications utilize data to make intelligent decisions on urbanization, sustainability, and mobility.

Traffic management is a key issue in urban planning. Transport-related issues are continually being confronted by most metropolitan areas with the ever-increasing levels of urbanisation. Typically this is manifest through traffic jams, increased traffic accidents and unsustainable expenditure of energy. Urban data analytics should improve the operation and organization stratagems for better city-scale transportation system management. Due to the development of state-of-the-

art sensing technologies and advancements in data analytics, it is now feasible to improve the scientific knowledge of data-driven urban traffic systems and their planning and management.

With the widespread popularity of mobile devices with location-aware capabilities and the extensive prevalence of location-based social networks like Twitter, users locations and their movements (trajectories) are being produced and gathered at an unprecedented rate. In this context it is vital to consider trajectory data from various aspects. For example, it is now directly feasible to use trajectories produced by the spatial-temporal data from organisations such as Twitter to estimate the historic and current road and traffic conditions and possible traffic jams and accidents that are occurring[2], [3]. Understanding the movement of users helps urban planners to efficiently use existing transport networks as well as inform policies on where to build new roads and rail networks. However, there are great many challenges in using trajectory data due to its inherent heterogeneity and the privacy sensitivities of user citizen trajectory data (whether it is captured through social media or other sensing devices).

Trajectories are typically captured and stored in discrete time-ordered sequences including user locations. Various sampling methods (e.g. distance-based or time-based) and different sampling strategies (e.g. low-frequency strategies like tweets or high-frequency strategies like GPS-Sensors) can be used for establishing location-aware trajectories. The choices taken can result in large variations in the trajectory sampling and their representation. Due to the inherent heterogeneity of trajectories, it can be difficult to effectively measure their similarity. This is essential for traffic flows, movements of crowds or other urban challenges. A key challenge is therefore to allow meaningful similarity-based trajectory processing. [4] have shown that over 90% of trajectories can be identified using no more than four given locations. Applying coarse and finer-grained sampling approaches leads to significantly different results. As calibration can convert heterogeneous trajectories data into that with fine sampling results and better representation of the raw data, it is essential to perform a calibration as pre-process of the trajectory data analytics. To achieve this, a trajectory-independent reference system is necessary to be built and used to reconstruct raw trajectories.

Essential properties of this reference system depend upon a stable reference set, data sources that are independent, and robust approach for calculating the trajectories [5].

Inherent to this work is user privacy. Figure 1 illustrates how Twitter data can provide more information on individuals than they would ever have thought possible, colour codes are individual Tweeters and vertical lines represent increasing times of day: tracking them throughout the day to potentially discover many aspects of their lives; where they live; where they travel; what they are doing; what time of day they are actually doing it etc. Furthermore once an individual has tweeted they can in principle be tracked directly and potentially forever using the Twitter Search API that is itself openly (programmatically) accessible.

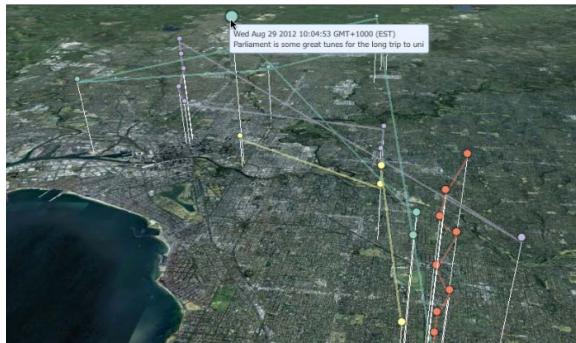


Fig. 1: Geospatially tracking Tweeters around Melbourne.

Approaches that can restrict location privacy leakage risk could inspire more users to share their geo-tagged data. As a robust privacy preserving solution, differential privacy [6] is broadly adopted to protect the privacy of sensitive data including location and trajectory data. However, the usability of differentially private solutions is always limited [7]. Differential privacy provides robust privacy guarantees at the expense of significantly restraining the utility of the data. Conversely, there is no such assumption on the use of data in k-anonymity solutions, which ensures reasonable levels of utility for the released data at the expense of some defects in privacy guarantees [8]. It is thus natural to adopt k-anonymity for differentially private location and trajectories analysis to improve the utility without lessening the privacy guarantees. Similarly, it is feasible to propose privacy preserving trajectory calibration solutions for similarity analysis of trajectories under differential privacy mechanisms, whilst enhancing the utility generally via k-anonymity. In this paper, we propose a private trajectory calibration system (PTCS) for heterogeneous social media user trajectories.

We make the following contributions. We conduct a k-anonymity approach to lessen the sensitivity of raw location data which can decrease the quantity of noise required to achieve ϵ -differential privacy and increase the trajectory utility. Specifically, the Corner Reference Points Micro-aggregation is proposed to reach k-anonymity. Based on anchor points discovered from the k-anonymized location data, we propose a novel differentially private clustering method (CC-DBSCAN) for mining the feature-based anchor points

considering both density and features. A private reference system for calibrating trajectories is proposed using the feature anchor points. Two trajectory similarity measures are proposed based on the geographic and semantic similarities. Experiments over real social media user trajectories are conducted, comprising spatial-tagged Twitter data harvested on major national Cloud facilities in Australia. The results empirically demonstrate that the private trajectory calibration system (PTCS) can improve the utility for both geometric and semantic similarity measures over heterogeneous trajectories.

The rest of this paper is structured as follows. Section II describes the related work, focusing especially on trajectories calibration and the adoption of differential privacy compared to other approaches used for trajectories release and mining. Section III introduces the preliminary concepts used in the work as well as providing an overview of the solution. Section IV presents the data and methods adopted in the private trajectories calibration. Section V presents the evaluation metrics that have been used and the experimental results of the privacy-preserved trajectories using real-life data focused especially on trajectory similarity analysis. Finally, Section VI draws conclusions on the work as a whole and outlines areas of future work.

II. RELATED WORK

Numerous efforts have been used to monitor and estimate public transport issues in urban areas based on real trajectory data sets [9]. However, directly using original individual trajectories raises many privacy concerns as discussed in [10]. Many works try to preserve individual trajectory sensitive data whilst still ensuring their utility. K-anonymity [8], [11] and (k, δ) -anonymity [12] are common approaches. [11] proposed a novel approach that cloaks trajectories using space translation. This approach can be effective only when there are at least k trajectories able to be accessed for every event. In addition, due to the high heterogeneity of trajectories, it is hard to use k-anonymity solutions that need at least k individuals with the same trajectory to protect privacy in individual trajectories. Furthermore, the available privacy-preserving trajectories analysis methods based on k-anonymity cannot preserve the privacy in location and trajectory datasets sufficiently. This barrier can be solved by differential privacy. Differential privacy [6] has been broadly adopted to protect sensitive data including data with geospatial location [13], and for both sequential data [14] and trajectory-based data [15]. [13] demonstrated that sensitive data in locations could be protected by using reasonable amounts of noise, while maintaining a degree of utility for location-based services. [16] introduced a classical approach to implement a differentially private solution for location data mining, aimed at providing a private location pattern mining method to release the results of the clustering algorithm for the public with sensitive data protected. However, this approach could not scale to city-wide data. The algorithms in [16] were established to handle these types of drawbacks and provide a basis for extensibility of the privacy-demanding solutions. Based on the results of these

explorations, the benefit of differentially private approaches for location and trajectory privacy is that they provide a rigorous privacy model to protect sensitive spatial information while still ensuring considerable utility of the synthesized data for analysis and mining. Several other differential privacy approaches have also been developed to release and mine spatial-temporal data. However, these approaches reduce the utility of the data when implementing differential privacy in practice, due to the degrees of noise that are introduced. Too much noise reduces the utility of spatial data for analysis and mining, especially with aggregation of large scale geospatial information; too little noise increases the risk of privacy leakage. [17] put forward a wavelet-conversion solution to decrease the magnitude of the noise. However, these improvements are mainly for point level spatial information rather than sequences of point data, i.e. trajectories. Other researchers have implemented differential privacy to preserve sensitive data in trajectories for safe mining and analysis. For example, [18], [14], [16], [19] have explored differentially private trajectories (or sequential) data synthesis solutions protecting sensitive data from potential attacks such as re-identification attacks. [14], [19] represent trajectories based on the prefix tree structure and released synthesized trajectories from a differentially private noisy prefix tree. However, due to the heterogeneous nature of raw trajectories, more noise will be required due to the high global sensitivity according to the definition of differential privacy. Appropriate discretization of the trajectory domains can reduce the magnitude of the trajectories datasets to help tackle the heterogeneous nature of trajectories, but this increases the likelihood of privacy issues arising. [18] adopted a reference system with homogeneous references to decrease the scale of possible transitions. However, this reference system is coarse grained and without privacy guarantee mechanisms. [15] use a hierarchical reference system to capture serial time intervals of raw individual trajectories which can reduce location loss. In addition, they proposed a privacy preserving synthesis mechanism under ϵ -differential privacy that is used to support a hierarchical reference system to discretize an individuals' trajectory according to differing velocities. However, the reference system is based on space-based anchors that divide the whole domain into equal grid cells whose centroids are used as the anchor points. This solution is easy to implement but cannot capture the distribution of the trajectories. Thus it may be too fine-grained for sparse trajectory distributions but too coarse for dense ones [5]. To the best of our knowledge, there is no existing research about private trajectory calibration. This paper is the first exploration to support a privacy supporting trajectory calibration system for trajectory data similarity analysis by combining k-anonymity with Differentially-private network trace analysis to improve the data utility.

III. PRELIMINARIES

In this section, related preliminary concepts are given, followed by a solution overview.

A. Reference System

1) *Anchor Points:* Anchor points are relatively stable locations in the spatial domain that do not change much, compared to trajectory data. Anchor points are used to construct reference systems. There are several different kinds of anchor points that are used in our work: cluster-based anchors aggregate clusters from sufficient archived trajectories data and use centroids of clusters as anchors. Specifically, we cluster historical trajectory datasets through density-based clustering approaches such as DBSCAN [20], which can then merge densely populated clusters to specific anchor points, e.g. known buildings or parks. However, there are some drawbacks when performing cluster approaches directly. For example, anchors should be relatively stable and have sufficient amounts of archived trajectories. Furthermore, in some cases, significant locations may be filtered out as noise. To overcome these drawbacks, anchors can be harvested by solutions that improve the clustering approach to discover clusters that not only meet the density condition, but also the particular feature situations (e.g. direction change in the trajectory), which can reduce the scale of clusters and ensure that they contain the desired features. Such features, like turning points and congestion points, can characterize the shape of trajectories comprised of fewer location points. Figure 2 illustrates cluster-based anchor discovery, and the anchor point can be extracted from each cluster centroid. Here a1 is the Melbourne Central station, a2, a3 are State Library of Victoria, and a4-a6 are clusters on roads.

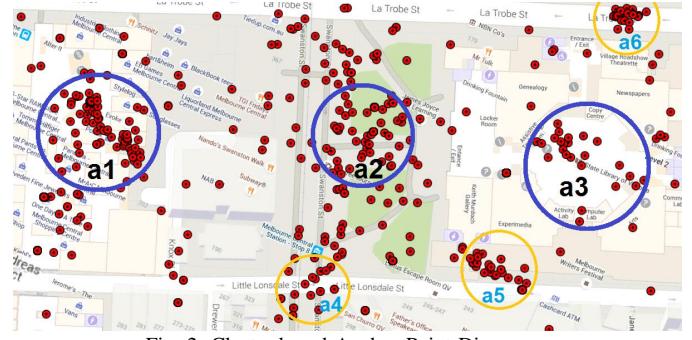


Fig. 2: Cluster-based Anchor Point Discovery.

2) *Reference System:* A reference system RS is an anchor point dataset, namely $RS = [a_1, a_2, \dots, a_m]$, where A is a set of anchor points with geometry based on their relation to fixed and globally known landmarks. This feature can guarantee the consistency of the trajectory when processed by the reference system.

3) *Calibration Process:* After constructing a reference system, trajectories can be calibrated based on the set of homogeneous anchor points. This process occurs as follows: with a reference system RS consisting of a set of anchor points A, the calibration process for one trajectory $t_j = [p_1, p_2, \dots, p_n]$ based on RS involves transforming t_j into $t'_j = [a_1, a_2, \dots, a_m]$ ($a_i \in A$ where $1 \leq i \leq m$, and m may not be equal to n). Hence t'_j is the calibrated trajectory of t_j . The main purpose of calibrations is to construct a new trajectory t'_j that can respect

the original trajectory as much as possible, and reduce the amount of points identified in the original route [5]. Figure 3 illustrates the anchor points and calibration process in the trajectory reference system.

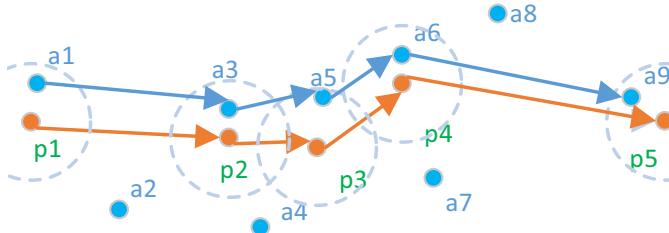


Fig. 3: Trajectory Reference System Calibration Process.

B. Differential Privacy

Differential privacy was proposed by Dwork in [6]. It is based on the idea that valuable knowledge can be gained from datasets without disclosing sensitive information. It offers rigorous privacy assurances that one individual cannot be recognized whenever this individual is in or is deleted from the dataset, i.e. the results will not change sufficiently to identify the difference.

The formalized definition of differential privacy is that if an individual is deleted from a database, there is no output that becomes obviously changed. Specifically, a private function F with ϵ -differential privacy for databases D_1 and D_2 , differing at most one element from each other, satisfies differential privacy if for all outcomes of the database S ($S \subset \text{Range}(F)$) there is:

$$\Pr[F(D_1) \in S] = e^\epsilon \Pr[F(D_2) \in S], \quad (1)$$

The maximum query output variation when removing an element of the database is represented by the global sensitivity of a given query [6].

$$\Delta f = \max_{D_1, D_2} \|f(D_1) - f(D_2)\|, \quad (2)$$

Differential privacy has two important properties:

- Sequential Composition: differential privacy provided by a set of mechanisms M_i on an input set D is $\sum_i \epsilon_i$.
- Parallel Composition: if every mechanism M_i acts on a disjoint subset $D_i \in D$, the privacy provided will be $(\max(\epsilon_i))$ -Differential Privacy for all M_i .

C. Differential privacy utility enhanced with In-sensitive Micro-aggregation

1) *Micro-aggregation*: A k-anonymity data set meets the condition that for every arrangement of quasi-identifier attributes values, there is an integer $k > 1$ for at least k records in the data set sharing that combination [7]. Micro-aggregation is a general approach to achieve k-anonymity. The anonymization can be achieved by building homogeneous groups of k or more records within the data set so that we can later change their values with aggregated values, normally the cluster centroid [21].

Let $S = \{x_1, x_2, \dots, x_n\}$, $x_i = (x_{i1}, x_{i2}, \dots, x_{im})$ be the n records and m attributes data set, micro-aggregation function $F(S) = \bar{S}$, \bar{S} is the k -anonymity version data set. Let $M(S)$ be a ϵ -differentially private mechanism, for an arbitrary query function q the response of $M(S) = q(S) + \text{noise}$. The scale of the random noise is adjusted to the sensitivity of the q . Hence, the utility of an ϵ -differentially private mechanism for q can be improved by reducing the misrepresentation introduced by the random noise. The query function q can be changed to a less sensitive one, that is $q \circ F$, namely run q on the k -anonymity version data set \bar{S} , that is $M(X) = (q \circ F)(S) + \text{noise}$, to reduce sensitivity of the results [7]. The information loss of $M(S)$ induced by adding noise will be reduced if the ϵ -differentially private mechanism supports lower sensitivity query functions, namely $q \circ F$. \bar{S} consists of the centroids of the clusters, hence centroids should be steady against modifications of one record in S .

2) *Differential Privacy and In-sensitive Micro-aggregation*: Let $S = \{x_1, x_2, \dots, x_n\}$ and $S' = \{x'_1, x'_2, \dots, x'_n\}$ be two data sets with only one different record, and $F(S) = \{c_1, c_2, \dots, c_r\}$ and $F(S') = \{c'_1, c'_2, \dots, c'_r\}$. If every corresponding clusters pair in $F(S)$ and $F(S')$ differs at most one record, F is an insensitive micro-aggregation algorithm. The insensitive micro-aggregation can be achieved by using an order relation consistent distance metric. A distance function $d : X \times X \rightarrow \mathbb{R}$ is consistent with an order relation \leq_x if $d(x, y) \leq d(x, z)$ whenever $x \leq_x y \leq_x z$ [7]. The effective approach to achieve the order relation consistent distance is to define a total order relation among the records in S . Let $R, R \in X$, be a reference point, if for a pair of records x and $y \in S$ and $dis(R, x) \leq dis(R, y)$, we can say that $x \leq y$. Here, the dis can be the Euclidean distance. In addition, the reference point R should be selected from the boundaries to improve the within-cluster homogeneity.

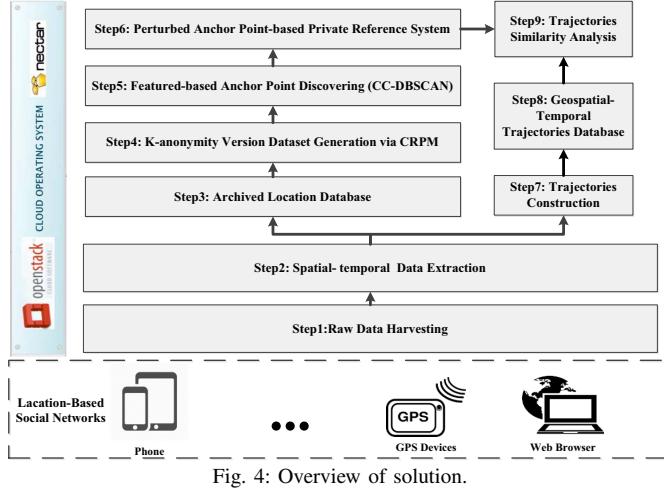
For the \bar{S} generated through insensitive micro-aggregation, i.e. the k -anonymized version of S , the research objects are changed from individuals to groups of k individuals. In the location and trajectory scenario, the data set has only numerical attributes. Hence, the insensitive micro-aggregation function $F(S) = \bar{S}$ generates \bar{S} with minimum cluster size k . In the generated data set, all records are replaced by the centroid (calculated by taking the mean of all records in each cluster), and all records within each cluster are identical. Let the qr be the function for the values of r^{th} attribute of the data set, $qr \circ F$ obtains the centroid of $F(r^{th}$ record in S), and the maximum change in any centroid is at most $\Delta(qr)/k$. Consequently, the sensitivity of each individual query $qr(\bar{S})$ is upper bounded by $\Delta qr(S)/k$, namely the global sensitivity of differentially private mechanism on qr is $\Delta(qr \circ F) \leq \Delta(qr)/k$. Since there are n/k different queries in S , the sensitivity of $qr(\bar{S})$ is upper bounded by $n/k \times \Delta qr(S)/k$. If we let the $n/k \times \Delta qr(S)/k \leq \Delta qr(S)/k$, then $k \geq \sqrt{n}$.

D. System Overview

The purpose of the work is to construct a privacy-preserving trajectory calibration model based on a private reference system with enhanced utility. This uses a Corner Reference Points

Micro-aggregation (CRPM) to establish a k-anonymity version of the dataset that lessens the sensitivity of raw locations and thereby reduces both the execution time and information loss. A novel differentially private clustering method (CC-DBSCAN) is used to discover feature-based anchor points considering both point density and features. A private reference system is also used to calibrate trajectories based on perturbed feature-based anchor points.

Figure 4 provides an overview the utility-enhanced private trajectory calibration system under differential privacy along with its core components.



IV. PRIVATE TRAJECTORIES CALIBRATION

In this section, the method used to establish the private trajectory calibration system (PTCS) is described. Specifically, we focus on Tweet harvesting and location extraction followed by feature-based anchor aggregation and perturbed the centroid of each cluster as perturbed anchor points and use them to be the spatial index as a reference system. Individual trajectories are aligned to this private reference system according to various geometric strategies, namely the trajectory calibration. In the calibration part we process and publish the anchor point perturbed by different privacy, so it will not breach the differential privacy definition.

A. K-anonymity Version Dataset Generation via CRPM

This is the step 4 in the Figure 4. To support private trajectory calibration we must define a total order for the records in $\text{Dom}(X)$ by choosing reference points from the boundary (corner points in this algorithm). Provided that the sequence of order relations is constant, different total order relations can be adopted in every micro-aggregation stage. The CRPM uses reference points set given as four fixed reference points $R1, R2, R3$ and $R4$, to achieve multiple total order relations. This approach is beneficial to promote the within-cluster homogeneity. The CRPM with four reference points can maintain the total order and reduce the execution time. It also decreases the amount of groups produced by the first stage to increase the data quality [22]. For two-dimensional space, the reference points are obtained from the four extreme points on the

boundary as follows: $B_{Min} = \min\{x_j(i) | 1 \leq j \leq n, 1 \leq i \leq 2\}$, $B_{Max} = \max\{x_j(i), 1 \leq j \leq n, 1 \leq i \leq 2\}$. Two-dimensional coordinates of the four corners are $R1 = \{B_{Min}, B_{Min}\}$, $R2 = \{B_{Min}, B_{Max}\}$, $R3 = \{B_{Max}, B_{Min}\}$ and $R4 = \{B_{Max}, B_{Max}\}$.

Algorithm 1: CRPM Algorithm

```

Input: locations data set LS and k.
Output: KS: n/k k-version datasets.
Compute the four Corner references points R1 to R4;
for i:1 to 4 do
     $\forall p \in LS$ , compute  $d(p, R_i)$  and sort them in
    decreasing order on array Dis [i-1][];
    KS =  $\emptyset$ ;
    for i:1 to n/k do
        m = i%4;
         $rp = \text{MaxDis}_{p \in LS}$ ; // Select the furthest point to
        Ri in Dis[m][ ]
        CLi = rp;
        LS = LS - rp;
        for each p in LS do
            d ← dis(p, rp);
             $p' = \text{kthelement}(LS)$ ; //use median-of-medians to
            choose the  $k^{th}$  nearest distance to rp
        for each p in LS do
            if  $dis(p, rp) \leq dis(p', rp)$  then
                CLi = CLi + p;
                LS = LS - p;
        KS = KS + CLi;
    for each p in LS do
         $d_{min} = \infty$ ;
    for each CLi in KS do
        cp ← calculate the centroid of CLi;
        if  $dis(p, cp) \leq d_{min}$  then
             $d_{min} = dis(p, CLi)$ ;
            CLi = CLi + p;
            Update(KS);
            LS = LS - p;
    return KS;

```

Algorithm 1 depicts the CRPM algorithm. Given 11 locations in the datasets, namely $n=11$, and k is chosen as 3, it contains $n/k=3$ iterations to initially produce $n/k=3$ groups. In every loop, the farthest location rp from the reference point is chosen as the initial point, and then the distance of every other vector to rp is calculated; this is followed by selecting the $k-1=2$ closest vectors to rp in order to aggregate the cluster CLi . After $n/k=3$ loops, there are $n \% k=2$ vectors left. Furthermore, these remaining vectors are merged into their nearest group.

B. Featured-based Anchor Point Discovery

This is the step 5 in the Figure 4. Anchor point extraction is the first step to construct the reference system. In order to improve the utility of anchor points, some important features of trajectories must be taken into account, e.g. congestion

points, intersections and turning points. We propose a novel clustering method for mining the feature points in an archived trajectories database considering both point density and features. Generally, the regions with denser locations are more reasonable to be used to re-describe the raw trajectories if they are used as anchor points. In our approach, we aggregate dense areas into clusters and adopt the centroids of each cluster as an anchor point. Given a trajectory $tj = [p_1, p_2, \dots, p_n]$ and $tj' = [p'_1, p'_2, \dots, p'_n]$, their moving direction at point p_i are given as $\overrightarrow{p_ip_{i+1}}$ and $\overrightarrow{p'_ip'_{i+1}}$ respectively. There are likely to be many trajectories that change direction at intersections. Therefore, if the angle θ between these two vectors is in $(0, \pi)$, there is a potential intersection (it is a T-junction or crossroads). If θ tends to $\pi/2$, there is a higher probability of finding an intersection.

Algorithm 2: CC-DBSCAN

```

Input: the universal set of trajectories  $p$ ,  $\mu$ ,  $\varphi$  and  $r$ .
Output: Features Clusters.
Construct R-Tree index over P;
for each point  $p$  in  $p$  do
    if  $p.clustered$  is false then
        Queue SD= new Queue(); // new seeds queue
        Points RL= new Points (); // new points set
        SD.add(p); RL.add(p);
        while SD is not null do
            point  $p'=\text{SD.pop}();$ 
            Points  $P'=\text{RQ}(p',r)$ ; //Collect all other points
            in around  $p'$  within  $r$ ;
            for  $i = 0$  to  $|P'|$  do
                if  $P'[i]$  is not clustered and  $P'[i]\ominus p'$  then
                    RL.add( $P'[i]$ ) //add  $P'[i]$  to result set;
                    SD.add( $P'[i]$ ) //use  $P'[i]$  as new seed;

            if  $|RL| \geq \varphi$  then
                //result set is bigger than the threshold;
                C.add(RL); //add result set into cluster set;
                for each point  $p''$  in RL do
                     $p''.\text{clustered}=\text{true};$ 
                RL.clear(); //clear RL;

for each cluster  $C$  in clusters do
    if there is surge point in  $C$  then
        Split C into clusters SC[]; for  $SC \in SC[]$  do
            if not satisfies the minimum number
            requirement then
                 $\forall$  points  $\in SC$  marked as noise;

```

In order to address this situation, the general DBSCAN algorithm is improved to aggregate locations in trajectories while considering the direction changes through the CC-DBSCAN which is used for discovering anchor points. This novel clustering algorithm is realized in two steps: (1) using DBSCAN with smaller Eps and bigger MinPts[20] to cluster the archived location database, and (b) CC-DBSCAN is applied

to distinguish points that satisfy direction changes, and then perform density-based clustering on previous point sets with noisy point filtering. In CC-DBSCAN, the density-connected relation is modified to coherence-connected relation, where coherence of point p_1 and p_2 is given as: Numbered Equation

$$\text{coherence}(p_1, p_2) = \exp\left(-\left(\frac{\text{dis}(p_1, p_2)}{\delta}\right)^\alpha\right) \cdot |\sin\theta|^\beta \quad (3)$$

Here, α and β are tuning parameters, $\text{dis}()$ is Euclidean Distance, and δ is a scaling factor [23]. In addition, if the $\text{coherence}(p_1 \text{ and } p_2)$ is bigger than the threshold μ , we can say they are directly coherence-connected, denoted by $p_1 \ominus p_2$. In addition, a temporal sequence should be considered in the trajectories database. Therefore, we provide CC-DBSCAN with the temporal constraint, i.e. all locations in a group should be temporally sequential. Temporal constraint reveals that the sequential order in the cluster should rise gradually without sudden surge (a tweet every hour vs every second impacts on the trajectory). If the temporal constraint is broken, the cluster should be distributed into 2 candidate groups at the surge location and then each cluster will have to be evaluated whether it fulfills the minimum number required or marked as noise [24].

C. Perturbed Anchor Point-based Private Reference System

This is the step 6 in the Figure 4. In Algorithm 3, $\text{Lap}(\sigma_{ct})$ is used for perturbing the counts of each cluster C_j extracted by CC-DBSCAN, and $\text{Lap}(\sigma_{cc})$ is used to perturb the centroid of each cluster C_j extracted by CC-DBSCAN. If the perturbed count CT' is greater than the threshold ic , then the region C_j is marked as a perturbed anchor point.

Algorithm 3: Perturb the Anchor Point via DP

```

Input:  $\mu, \varphi, p$  and Threshold  $\gamma$ .
Output: Perturbed Anchors Set.
M=CC-DBSCAN ( $\mu, \varphi, P$ );
for  $j = 1$  to  $|M|$  do
     $CT' = |Mc_j| + \text{Lap}(\sigma_{ct}^j);$ 
    if  $CT' > \gamma$  then
         $CentroidCC_i = \frac{\sum_{k=1}^{|Mc_j|} (x_k, y_k)}{|Mc_j|};$ 
         $CC' = \text{NoisyLap}(\sigma_{cc}^j)(CC_j);$ 
         $G = G \cup \{CC'\};$ 
     $CC' = (0, 0); CT' = 0;$ 

```

The count sensitivity and the centroid sensitivity for the cluster C_j are given as follows. The count sensitivity Δf_{ct}^j is defined as $\text{MAX}(\text{NUMindividual(points)})$, $\forall \text{individual} \in C_j$. So $\sigma_{ct}^j = \frac{\Delta f_{ct}^j}{\epsilon_{ct}}$, where ϵ_{ct} is the privacy distribution in the counting points step. The centroid sensitivity Δf_{cc}^j is defined as $\text{MAX}(\text{distance}(p_i, p_j))/2$ $\forall p_i, p_j \in C_j$. So $\sigma_{cc}^j = \frac{\Delta f_{cc}^j}{\epsilon_{cc}}$, where ϵ_{cc} is the privacy level of the counting centroid step.

Note that the method $\text{NoisyLap}(\sigma)$ perturbs a real location coordinate $lr(xr, yr)$ to a perturbed location coordinate $lp(xp, yp)$

was introduced by [25]. After the anchor points are perturbed, the private reference system can be generated based on them.

D. Trajectory Calibration and Complement

Based on the private reference system, each new trajectory can be calibrated by performing an appropriate alignment. Specifically, the locations in a trajectory are aligned to specific anchor points according to a geometric strategy. This geometry-based calibration method maps the locations in the trajectory to the anchor points according to the spatial relationships between them. The strategy used in geometry-based calibration method is to discover the nearest anchor point a^* for every location in the trajectory that will be presented by the a^* anchor point. The a^* uses the maximum distance threshold η to restrict the maximum distance used to map locations in trajectories to anchors points. η is empirically set as 50m. If we use a constrained nearest neighbour searching solution for geometry-based calibration methods on raw data, the logarithmic-scale complexity is huge. Therefore, an R-Tree index [5] is constructed on the raw trajectories database before calibration, which can reduce the logarithmic-scale complexity to $(O(\log|A|))$ for each location in the trajectory, and $(O(|T|\log|A|))$ for all locations in T .

V. EXPERIMENTAL EVALUATION

In this section, our private trajectories calibration is empirically evaluated using a location-based social network dataset comprising geo-tagged tweets.

A. Database

The geo-tagged tweets collecting and pre-processing is executed on the Australian National eResearch Collaboration Tools and Resources (NeCTAR) Research Cloud (www.nectar.org.au). The harvester adopts RESTful client using Twitter Search API to harvest tweets. The next step is to process and incorporate harvest tweets into the NoSQL database (CouchDB). CouchDB was selected in part as it natively supports MapReduce.

The system supports elastic scaling and more harvesters can be deployed across Cloud resources. Eight medium-sized virtual machines with 8GB memory and eight virtual CPUs with 250GB volume storage and 100GB object storage were used as the basis for the work. Multiple harvesters enhance the reliability of the tweet harvesting procedures. The trajectories database contains two parts: tweets generated on roads or within ξ_1 distance of a road, and traffic congestion related context made around roads within distances (ξ_1, ξ_2) ; here the $\xi_1 = 5m, \xi_2 = 50m$.

After harvesting the Twitter data, it is necessary to process the tweets to generate useful results. For this purpose, we have used the MapReduce functions of CouchDB. The total number of tweets combining the data from the Search and Stream APIs after removing duplicates was 1,248,569. After pre-processing, the final data set was composed of 132,407 locations. These data were saved with the following structure: *UserId|PointID|Longitude|Latitude|Date*.

Finally, the raw individual trajectories database has constructed using the individual locations of geospatial tagged tweets. Specifically, if the distance between two or more consecutive locations in the same user's geotagged tweets within $t = 120$ min is more than $d = 100m$, the trajectory can be aggregated using these location points [26]. Here, t is the temporal threshold to distinguish two consecutive locations of the same user's tweets, which is the minimum spatial interval. Trajectories are produced through merging two or more geotagged tweets made from the same user with locations $\geq 100m$ away between each other, and the time interval ≤ 75 min. Based on this method, 7953 trajectories were recognized in total.

B. Utility Evaluation Metrics

To measure the degree of utility of the private trajectories calibration system, three effective trajectories similarity evaluation methods are adapted to measure the distortion between the raw trajectories and calibrated ones. The anchor point utility is also measured.

1) *Anchor Point Utility Measures*: To evaluate the utility improvement using our insensitive microaggregation method, we adopt the Sum of Distance Drift (SDD) and Recall to measure the information loss when performing the CC-DBSCAN on different k-anonymity versions and the original data sets. The first step is to find the corresponding real anchor points for each perturbed anchor points discovered by the CC-DBSCAN algorithm. And then, we find the nearest real anchor points to the corresponding perturbed anchor points, where these anchor points have been reduced to their centroids. The first metric is recall, namely, the percentage of real anchor points inferred by the perturbed data users. The recall can be defined as follows:

$$\text{Recall} = \frac{\text{Count of IST}}{\text{Count of anchor points in the CT}}, \quad (4)$$

Here, IS is the set of anchor points discovered through performing CC-DBSCAN on the raw location database, and CT is the set of anchor points discovered through performing CC-DBSCAN with differential privacy guarantees on the raw location database and k-anonymity version data sets. IST is the set of anchor points in CT that have more than one inferred anchor points from IS. For example, a_1, a_2, \dots, a_5 , namely CT, are the anchor points discovered on original location dataset, followed by performing CC-DBSCAN with differential privacy guarantees on it in which the perturbed anchor point a_1^*, a_2^* , namely IS, can be found and the distance between a_1^*, a_1 and a_2^*, a_2 are both less than the threshold, namely they one inferred anchor points from IS, then the Recall is 20%. As seen from the definition of recall, this can be used to assess the percentage of anchor points that have been discovered from the set of real anchor points.

2) *Trajectories Similarity Measures*: In this paper, a sequence of spatial temporal points can be used to represent the trajectory t_j of a social media user, $t_i = \{(x_1^i, y_1^i, t_1^i), (x_2^i, y_2^i, t_2^i), \dots, (x_N^i, y_N^i, t_N^i)\}$, where N is the length of the trajectory. Two distinctive similarity measures

are adopted in this part: geographic similarity and semantic similarity. Euclidean Distance is used as the geographic similarity that reveals the spatial relation among the trajectories and semantic similarity demonstrates the shape variance of the trajectories [27].

Euclidean Distance. Euclidean distance $\text{dis}(t_1, t_2)$, namely L2-norm, is used as the value of similarity of trajectories with the same length, which can be denoted as follows:

$$\text{dis}(t_1, t_2) = \frac{\sum_{i=1}^n \text{dis}(p_{1i}, p_{2i})}{n}, \quad (5)$$

Semantic Similarity. Semantic Similarity s_{sem} calculate is defined to describe the shape Similarity of trajectories. Here, the Hausdorff distance $d_H(t_i, t_j)$ is adopted to describe the Semantic Similarity.

$$s_{sem}(t_i, t_j) = d_H(t_i, t_j) = \max(h(t_i, t_j), h(t_j, t_i))$$

$$h(t_i, t_j) = \max \min_{a \in t_i, b \in t_j} \|a - b\|,$$

where a and b are 2D points belong to t_i, t_j , respectively.

Mixed Similarity. Let cr_i and cr_j be geometric centroids of t_i and t_j , \vec{se}_i and \vec{se}_j be vectors from the start locations to the end locations, respectively, and the angle between them is $\theta < \vec{se}_i, \vec{se}_j >$. The Mixed Similarity is defined as:

$$s_{geo}(t_i, t_j) = d(cr_i, cr_j) + \frac{(\|\vec{se}_i\| + \|\vec{se}_j\|)}{2} e^{(\lambda|\theta|/\pi)}, \quad (6)$$

where $ctr(t_i) = (\frac{\sum_{i=1}^{n-1} (x_{i+1}^2 - x_i^2)}{2 \times \sum_{i=1}^{n-1} (x_{i+1} - x_i)}, \frac{\sum_{i=1}^{n-1} (y_{i+1}^2 - y_i^2)}{2 \times \sum_{i=1}^{n-1} (y_{i+1} - y_i)})$, and λ is an argument related to the magnitude of the decreasing rate of the exponential function.

C. Evaluation Analysis

In this section, we evaluate the utility of the private trajectories calibration approach over tweet trajectory datasets in terms of the metrics above. As noted our implementation was performed on virtual machines offered through the NeCTAR Research Cloud. The implementation itself was done in Java.

1) General Trajectories Feature Mining Results: Using the improved density-based clustering algorithms, namely CC-DBSCAN, we cluster the archived trajectories location database and different k-anonymity version data sets. Based on the clustered results, the anchor point sets are generated by extracting the centroid of each cluster.

2) Perturbed Anchor Extraction Results: To address the privacy concerns, these raw anchor points are perturbed through differential privacy mechanisms. Here, the threshold value parameters were set as $\text{MinPts} = 50$ and $\text{Eps} = 0.01$ Km. The differential privacy guarantees can be implemented by performing a sequence of differential privacy mechanism steps. The visualization of perturbed anchor points is shown in Figure 5. The privacy leak level ϵ can be composed of $\epsilon = \epsilon_{ct} + \epsilon_{cc}$ where ϵ_{ct} is the privacy leakage level used for perturbing the count of the numbers of point in each cluster and ϵ_{cc} is the privacy leakage level used for perturbing the count of centroids comprising each cluster. Note that the experiments were performed on three classical differential

privacy leakage levels obtained by previous experiments [28], that is $\epsilon = \{\text{Strong} : (0.01, 0.01), (0.05, 0.05); \text{Normal} : (0.1, 0.1), (0.5, 0.5); \text{Weak} : (1, 1), (10, 10)\}$. The privacy level can be considered as strong when epsilon set at 0.01 and 0.05, and as normal at 0.1 and 0.5, and as weak at 1, 10. Here, the first ϵ_1 is the privacy budget used for private reference system, and ϵ_2 is the privacy budget used for private trajectories release.



Fig. 5: Visualization of Perturbed Anchor Points.

3) Visualization of Calibration and Perturbation Effect: We visualize the results to give an instinctive explanation of the calibration effect in Figure 6, prior to carrying out the evaluation of the quantitative performance. The green curve illustrates the raw trajectory sequences recognized from individual routes, and the darker curve demonstrates the calibrated trajectories using our private reference system based on noisy anchor points. Intuitively, a better reference system can provide supplementary information for those anchor points "around" the trajectory segments in space, e.g., Point 0 to R1, Point 7 to R13. The completeness approach can include more anchor points to obtain better calibrated results, e.g., R2, R4, R6. As shown in this figure, the trajectory calibration system can improve the utility of raw trajectory for similarity analysis.

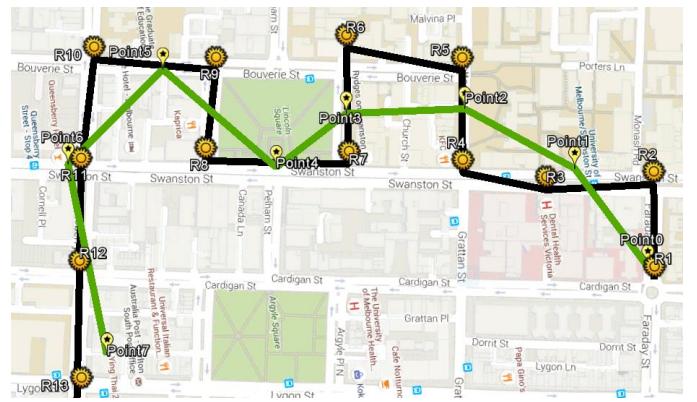


Fig. 6: Visualization of Calibration Effect.

4) Anchor Point Utility Measures: In this part, the most important task was to find the threshold that can be used to declare whether the real anchor point was discovered or not. An optimal threshold can be used to ensure a high recall and associated low distance between real anchor point and released anchor point. The way we address this is to set

the minimum Euclidian Distance between the real trajectory and the obfuscated one at which the recall is higher than 70% of the threshold with the appropriate threshold. We have assessed the recall of the Perturbed Anchor Point Mechanism on different k-anonymity version data sets ($k=1$ is the original data set), as demonstrated in Figure 7. It is clear that the recall of the whole rate increases as ϵ increases. Namely, privacy and utility are trade-offs, i.e., the higher degree of privacy preservation allows fewer anchor points to be recognized. Using the method described above, the thresholds are also determined. The results show that the larger the value of $\epsilon_{cc} + \epsilon_{ct}$, the more great the recall will be. As above, if the degree of privacy protection is higher, fewer anchor points will be found. At a given privacy budget, with the increase of k , the recall shows growing trends, the recall is lowest when $k=1$ because the global sensitivity is the most that will be reduced when performing K-anonymity preprocessing. At the same time, the threshold distance decreases, which proves that the accuracy and utility are improved by the insensitive micro-aggregation approach.

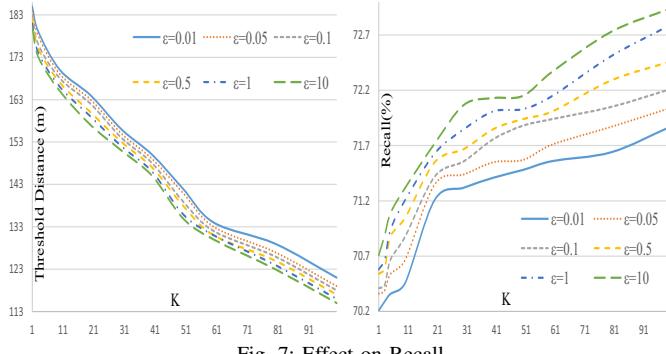


Fig. 7: Effect on Recall.

5) *Trajectories Similarity Measures:* In this experiment, we determine the Geographic Similarity, Semantic Similarity and Mixed Similarity between all private calibrated trajectories based on different k-anonymity version data sets and their associated raw trajectories. Based on these features, we can identify whether sufficient utility of private calibrated trajectories is guaranteed by our approach or not. The experiment is performed based on the calibration mechanism and private calibration mechanism on the various k-anonymity version datasets respectively. The experimental analysis is illustrated in Figure 8-11. As we can see from these figures, smaller distances denote the released trajectories are closer to the original trajectories. At a given privacy budget, with the increase of k , the ED (Geographic Similarity) shows decreasing trends between private calibrated and original trajectories. The Hausdorff Distance (Semantic Similarity) and Mixed Similarity show fluctuations with the increase of k , but Semantic Similarity and Mixed Similarity perform better on various k-anonymity version datasets than without k-anonymity mechanism (i.e. $k=1$), which will reach on optimized value around $k=50$ as the results shown. All evaluated distances grow with a descent in the privacy budget; hence a stronger privacy preserving level through increased Laplace noise should be added.

This proves the accuracy and utility of private calibrated trajectories can be improved through insensitive aggregation.

Based on these experimental results at different privacy preserving levels, we can conclude that stronger privacy preservation will damage the utility, but our private trajectories calibration solution can maintain the overall utility.

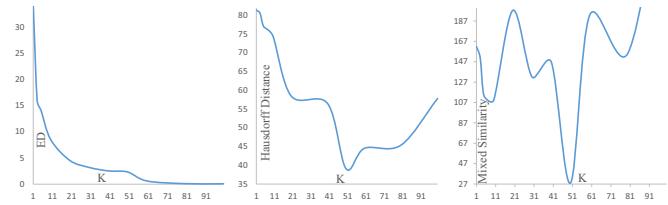


Fig. 8: Similarity Evaluation of Calibrated Mechanism.

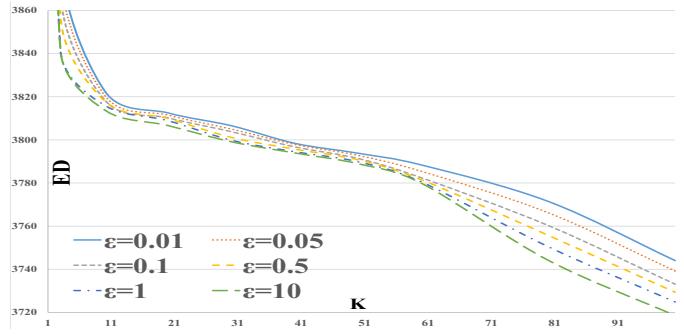


Fig. 9: Geographic Similarity on PTCS.

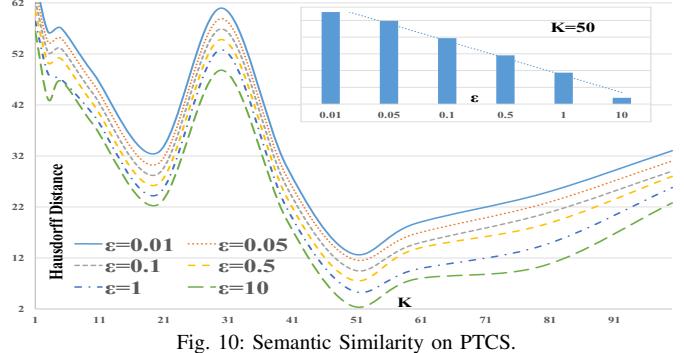


Fig. 10: Semantic Similarity on PTCS.

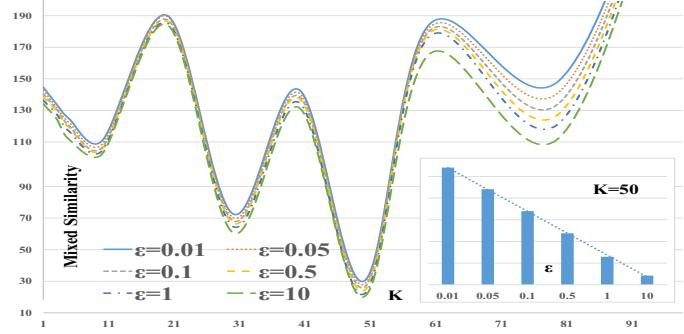


Fig. 11: Mixed Similarity on PTCS.

VI. CONCLUSIONS

In this paper, we proposed a Private Trajectories Calibration System (PTCS) under Differential Privacy, to release real trajectories in social media with focus here on Twitter. Our approach uses both differential privacy and k-anonymity

mechanisms to enhance the utility of the data without increasing the disclosure risks. Our PTCS mechanism proposes a differentially private reference system that enables sampling of discrete raw trajectories according to obfuscated feature-based anchor points, which can scale to large geo-spatial areas. Moreover, the PTCS mechanism extends the utility of differentially private approaches via Corner Reference Points Micro-aggregation. The CRPM reduces the sensitivity of raw locations data which can reduce the magnitude of noise required to achieve differential privacy. Comprehensive experimental results over real geo-tagged tweets demonstrated that the PTCS mechanism is efficient for trajectory similarity analysis. Implementing urban social media data analytics with privacy guarantees, our approach can be beneficial to similarity-based trajectory processing, and can help to improve the operation and organization stratagems for better city-scale transportation system management.

This paper reveals a new orientation for social media privacy with focus on geospatial privacy. For future work, the utility and precision of the private trajectories calibration system (PTCS) could be further enhanced. Specifically, the upper bound of PTCS can be established by effectively estimating and shrinking the sensitivity when performing differential privacy mechanisms in a hierarchical manner. This will allow for example to divide the large space of a city into suburbs or street levels, whilst ensuring that the accuracy and effectiveness of the calibrated trajectory complements can be improved by adding more possible locations that might be predicted using hidden Markov models or K-nearest Neighbours models into the calibrated trajectories.

REFERENCES

- [1] S. Xiao, X. C. Liu, and Y. Wang, "Data-driven geospatial-enabled transportation platform for freeway performance analysis," *Intelligent Transportation Systems Magazine, IEEE*, vol. 7, no. 2, pp. 10–21, 2015.
- [2] V. Agarwal, N. V. Murali, and C. Chandramouli, "A cost-effective ultrasonic sensor-based driver-assistance system for congested traffic conditions," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 10, no. 3, pp. 486–498, 2009.
- [3] R. O. Sinnott and S. Yin, "Accident black spot identification and verification through social media," in *2015 IEEE International Conference on Data Science and Data Intensive Systems*. IEEE, 2015, pp. 17–24.
- [4] Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel, "Unique in the crowd: The privacy bounds of human mobility," *Scientific reports*, vol. 3, 2013.
- [5] H. Su, K. Zheng, H. Wang, J. Huang, and X. Zhou, "Calibrating trajectory data for similarity-based analysis," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. ACM, 2013, pp. 833–844.
- [6] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," *Theory of cryptography*, pp. 265–284, 2006.
- [7] J. Soria-Comas, J. Domingo-Ferrer, D. Sánchez, and S. Martínez, "Enhancing data utility in differential privacy via microaggregation-based k-anonymity," *The VLDB Journal*, vol. 23, no. 5, pp. 771–794, 2014.
- [8] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, 2002.
- [9] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: concepts, methodologies, and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 3, p. 38, 2014.
- [10] M. Xue, P. Kalnis, and H. K. Pung, "Location diversity: Enhanced privacy protection in location based services," *Location and Context Awareness*, pp. 70–87, 2009.
- [11] H. Hu, J. Xu, S. T. On, J. Du, and J. K.-Y. Ng, "Privacy-aware location data publishing," *ACM Transactions on Database Systems (TODS)*, vol. 35, no. 3, p. 18, 2010.
- [12] O. Abul, F. Bonchi, and M. Nanni, "Never walk alone: Uncertainty for anonymity in moving objects databases," in *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*. Ieee, 2008, pp. 376–385.
- [13] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: Differential privacy for location-based systems," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 901–914.
- [14] R. Chen, B. Fung, B. C. Desai, and N. M. Sossou, "Differentially private transit data publication: a case study on the montreal transportation system," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 213–221.
- [15] X. He, G. Cormode, A. Machanavajjhala, C. M. Procopiuc, and D. Srivastava, "Dpt: differentially private trajectory synthesis using hierarchical reference systems," *Proceedings of the VLDB Endowment*, vol. 8, no. 11, pp. 1154–1165, 2015.
- [16] S.-S. Ho and S. Ruan, "Differential privacy for location pattern mining," in *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*. ACM, 2011, pp. 17–24.
- [17] X. Xiao, G. Wang, and J. Gehrke, "Differential privacy via wavelet transforms," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 23, no. 8, pp. 1200–1214, 2011.
- [18] F. Pratesi, A. Monreale, W. H. Wang, S. Rinzivillo, D. Pedreschi, G. Andrienko, and N. V. Andrienko, "Privacy-aware distributed mobility data analytics," in *SEBD*, Roccella Jonica, 2013.
- [19] R. Chen, B. C. Fung, N. Mohammed, B. C. Desai, and K. Wang, "Privacy-preserving trajectory data publishing by local suppression," *Information Sciences*, vol. 231, pp. 83–97, 2013.
- [20] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.
- [21] J. Domingo-Ferrer and J. M. Mateo-Sanz, "Practical data-oriented microaggregation for statistical disclosure control," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 14, no. 1, pp. 189–201, 2002.
- [22] C.-C. Chang, Y.-C. Li, and W.-H. Huang, "Tfrp: An efficient microaggregation algorithm for statistical disclosure control," *Journal of Systems and Software*, vol. 80, no. 11, pp. 1866–1878, 2007.
- [23] Z. Chen, H. T. Shen, and X. Zhou, "Discovering popular routes from trajectories," in *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*. IEEE, 2011, pp. 900–911.
- [24] L. Gong, H. Sato, T. Yamamoto, T. Miwa, and T. Morikawa, "Identification of activity stop locations in gps trajectories by density-based clustering method combined with support vector machines," *Journal of Modern Transportation*, vol. 23, no. 3, pp. 202–213, 2015.
- [25] R. Dewri, "Local differential perturbations: Location privacy under approximate knowledge attackers," *Mobile Computing, IEEE Transactions on*, vol. 12, no. 12, pp. 2360–2372, 2013.
- [26] L. Gabrielli, S. Rinzivillo, F. Ronzano, and D. Villatoro, "From tweets to semantic trajectories: mining anomalous urban mobility patterns," in *Citizen in Sensor Networks*. Springer, 2014, pp. 26–35.
- [27] M. Ra, C. Lim, Y. H. Song, J. Jung, and W.-Y. Kim, "Effective trajectory similarity measure for moving objects in real-world scene," in *Information Science and Applications*. Springer, 2015, pp. 641–648.
- [28] J. Hsu, M. Gaboardi, A. Haeberlen, S. Khanna, A. Narayan, B. C. Pierce, and A. Roth, "Differential privacy: An economic method for choosing epsilon," in *2014 IEEE 27th Computer Security Foundations Symposium*. IEEE, 2014, pp. 398–410.

A Machine Learning Analysis of Twitter Sentiment to the Sandy Hook Shootings

Nan Wang, Blesson Varghese

School of EEECS

Queen's University Belfast, UK

Email: {nwang03, varghese}@qub.ac.uk

Peter D. Donnelly

Dalla Lana School of Public Health

University of Toronto, Canada

Email: peter.donnelly@oahpp.ca

Abstract—Gun related violence is a complex issue and accounts for a large proportion of violent incidents. In the research reported in this paper, we set out to investigate the pro-gun and anti-gun sentiments expressed on a social media platform, namely Twitter, in response to the 2012 Sandy Hook Elementary School shooting in Connecticut, USA. Machine learning techniques are applied to classify a data corpus of over 700,000 tweets. The sentiments are captured using a public sentiment score that considers the volume of tweets as well as population. A web-based interactive tool is developed to visualise the sentiments and is available at <http://www.gunsontwitter.com>. The key findings from this research are: (i) There are elevated rates of both pro-gun and anti-gun sentiments on the day of the shooting. Surprisingly, the pro-gun sentiment remains high for a number of days following the event but the anti-gun sentiment quickly falls to pre-event levels. (ii) There is a different public response from each state, with the highest pro-gun sentiment not coming from those with highest gun ownership levels but rather from California, Texas and New York.

I. INTRODUCTION

On the morning of 14th December 2012 a heavily armed twenty-year-old man with “*significant mental health issues*” [1] and access to his mother’s legally owned firearms shot his way into the locked building of the Sandy Hook Elementary School (SHES) in Connecticut, USA and in less than eleven minutes murdered 20 children and 6 adults. Before driving to the school he had shot and killed his mother in her bed and at the conclusion of the shootings he took his own life.

The world media response to this tragedy was unprecedented in its scale and much has subsequently been written about the events themselves and how future similar events could be avoided [2]. Predictably much attention focussed on issues of gun ownership and control with those advocating tighter gun control clashing with those intent on exercising and protecting (as they saw it) the US constitutional second amendment right to bear arms. That debate continues and, in its totality, is beyond the scope of this paper in which we largely restrict ourselves to seeking to understand and interpret pro-gun and anti-gun sentiment as expressed on a social media platform, namely Twitter in a forty day period starting seven days before the shootings.

In order to achieve this we have used the application of machine learning to a sample of over 700,000 tweets made by individuals in the United States that contain one or more predetermined relevant key words. What we sought

to capture was pro-gun and anti-gun sentiment and how it changed over time. For this we developed a framework to collect, pre-process and classify tweets and further visualise the sentiments. A hand curated gold standard dataset is employed in classifying the entire sample. A number of machine learning approaches are evaluated and the approaches that provide maximum accuracy over a small sample are used for classifying the entire tweet sample. We visualise this sentiment by state taking into account state population size and gun ownership levels through an application we have hosted at <http://www.gunsontwitter.com>.

The key observations are that there is a peak of both anti-gun and pro-gun sentiment on the day of the shooting. Anti-gun sentiment quickly falls to pre-event levels whereas pro-gun sentiment remains high for a number of days. It is interesting to note that all states do not respond in the same way; the highest gun ownership states are not the most pro-gun, but California, Texas and Florida have highest pro-gun sentiment even when their larger population size is taken into account.

The remainder of this paper is organised as follows. Section II presents the methodology adopted in this paper to collect, pre-process, classify, summarise and visualise the sentiment of the tweets. Section III considers the machine learning approaches employed for classifying the tweets. Section IV takes the Sandy Hook school shooting case study into account by applying the methodology on a volume of tweets and visualising pro-gun and anti-gun sentiments. Section V presents the public health implications of the research reported in this paper. Section VII concludes this paper.

II. METHODOLOGY

The methodology for analysing public sentiment incorporates machine learning and (1) collects, (2) pre-processes, (3) classifies, (4) summarises and (5) visualises data. Figure 1 illustrates the sequence of steps in the methodology.

Raw data from Twitter is collected and pre-processed into a trimmed data set in an appropriate format, which is further split into training and testing data sets. In the machine learning stage different algorithms, such as Support Vector Machines (SVM), Maximum Entropy (ME), Tree, Bagged Tree, Boosted Tree, Random Forest (RF), Neural Network (NN) and Naïve Bayes (NB), are explored to build classifiers. A subset of more accurate classifiers are used for classifying the tweets.

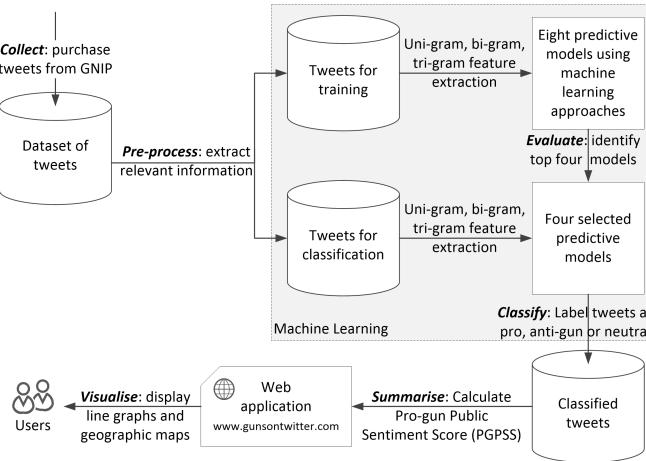


Fig. 1. Methodology for analysing and visualising public sentiment

A number of functions that can be used to derive meaningful information are applied on the classified tweets. The results are summarised using multiple visualisation techniques.

Step 1 - Collect: The raw data of tweets in JSON format is purchased from GNIP¹, a third-party authorised to resell historical tweets determined by predefined rules that takes into account filters based on the time-period, key words and geographic location. Compared to alternatives such as custom-built web crawlers or programming the Twitter Streaming API², this approach saved time and provided data in a consistent format, although the tweets had to be purchased.

Step 2 - Pre-process: A file containing a list of JSON files and their remote locations was obtained from the reseller. Each JSON file is organised for a ten minute period and contains the tweets and a large amount of information regarding the tweet³. The JSON files were parsed to extract the most relevant data for the research and were converted into the CSV format. All further tasks were performed using the R programming language in RStudio⁴. This was preferred to exploit the readily available packages for data mining and natural language processing.

Step 3 - Classify: In the machine learning stage, the sentiments of the tweets are extracted. The training data set contains 5000 tweets selected from the trimmed data while the remaining tweets are left for classification. The sentiment of each tweet is manually labelled into three classes, namely; pro-gun, anti-gun and neutral, to obtain a gold standard dataset. It is assumed that the distribution of tweets is not equal among these classes on grounds of the characteristic of Twitter - a convenient platform for users to express subjective ideas (which convey a pro-gun or anti-gun sentiment). Therefore, the numbers of tweets for three classes are adjusted as 2000 respectively for the pro-gun and anti-gun classes and 1000 for the neutral class. Intentionally giving extra weight to subjective

classes is known to produce classifiers that are more sensitive to these classes [3].

Using the extracted features classifiers are built using supervised training approaches which are presented in the next section. Prior to the deployment of more accurate classifiers, the effectiveness of all models are evaluated by the output of sample accuracy in a 10-fold cross validation [4]. Only top classifiers are selected and further developed based on the features extracted for the classification of the remaining tweets.

Step 4 - Summarise: In addition to the time stamps and coordinates that the data originally contains, a series of methods for calculating a Pro-Gun Public Sentiment Scores (PGPSS) are utilised for comparison. Consider a geographic region defined by g and a time frame t . The baseline PGPSS is denoted as

$$PGPSS_1 = \frac{count_{(g,t)}(\text{positive tweets})}{count_{(g,t)}(\text{negative tweets})}$$

Baseline PGPSS measures the number of pro-gun tweets over anti-gun tweets from a set of tweets originating from a state in a given time frame. It gives the degree of positivity in the selected tweets as an index which measures the positive tweets over negative tweets. However, $PGPSS_1$ is not the best indicator of positivity since it does not take the volume of tweets that originates from g . This poses a problem which is illustrated in the following example.

Consider the country g has population of 11 million, which has two states g_1 and g_2 . Suppose the population of g_1 is 10 million and g_2 is 1 million, and the total number of available tweets from the states be 1 million (200,000 positive and 800,000 negative) and 10,000 (8,000 positive and 2,000 negative) respectively. Now $PGPSS_1$ for g_1 is 0.25 and for g_2 is 4 which indicates a higher positive sentiment for g_2 than g_1 . The normalised $PGPSS_1$ for g_1 is 0.0625 and for g_2 is 1. However, this is not entirely accurate in that the volume of tweets that originated from g_2 is relatively a lot smaller than the tweets from g_1 . Hence, a score that takes the volume of tweets that originates from g_1 and g_2 into account is useful.

Hence, we define a PGPSS score, which is corrected for volume of tweets defined as

$$PGPSS_2 = \frac{count_{(g,t)}(\text{positive tweets})}{count_{(g,t)}(\text{negative tweets})} * \frac{count_{(g,t)}(\text{tweets})}{count_{(t)}(\text{tweets})}$$

The volume correction ratio for g_1 is $1,000,000/1,010,000 = 0.990099$ and for g_2 is 0.009901 . Now $PGPSS_2$ for g_1 is 0.247525 and for g_2 is 0.039604, and the normalised $PGPSS_2$ for g_1 is 1 and for g_2 is 0.16.

An additional correction for population is considered by defining a different PGPSS score which is denoted as

$$PGPSS_3 = \frac{count_{(g,t)}(\text{positive tweets})}{count_{(g,t)}(\text{negative tweets})} * \frac{count_{(g,t)}(\text{tweets})}{count_{(t)}(\text{tweets})} * \frac{population_g}{population}$$

¹<http://gnip.com/>

²<https://dev.twitter.com/docs/streaming-apis>

³<http://gnip.com/sources/twitter/historical/>

⁴<http://www.rstudio.com>

The population correction ratio for g_1 is $1\text{million}/11\text{million} = 0.090909$ and for g_2 is $1000/11\text{million} = 0.000909$. The PGPSS_3 score for g_1 is 0.225023 and for g_2 is 0.000036, and the normalised PGPSS_3 for g_1 is 1 and for g_2 is 0.000016.

All PGPSS scores are normalised between 0 and 1 for the purpose of visualisation. To interpret the PGPSS scores in the case of public sentiment towards gun use, g_2 is more supportive of guns than g_1 according to PGPSS_1 , but after correcting the scores, as should be g_1 is presented as a state more supportive of guns than g_2 . The corrections applied to the public sentiment scores provide a better overview in the context of the volume of tweets and population.

Step 5 - Visualise: A web application is developed under the Shiny framework⁵ for visualising data. Motion charts, line graphs and geographic maps are generated using Google Charts API⁶ through the googleVis package [5]. These techniques present results at the country and state levels as well as hourly and daily analyses.

Motion charts display a continuous bubble view of data for a combination of six variables simultaneously; for example, the x and y axes, a bubble, its size and colour and a slider all display relevant information. This technique provides an animated overview of the changes through an interface to observe variations over time. Additionally, a bar plot and a line graph are embedded in the motion chart.

The line graph presents the volume of classified tweets over the dimension of time. A control gadget is available for users to monitor the data on daily or hourly basis. Options on zooming and changing time period are embedded in the graph.

The geographic map visualises a series of PGPSS results in the form of a state-level map. A particular type of Public Sentiment Score and time frame are selected by users each time to be projected into a gradient colour on the map. This technique is useful to interpret the statistics from a geographic perspective.

III. MACHINE LEARNING APPROACHES

Feature extraction and modelling are crucial steps in any machine learning approach. Various techniques on feature selection have been examined in previous research, from the baseline approach of N-gram features [6] to the advanced linguistic analysis on POS tags [7]. Meanwhile, SVM and NB are the top two frequently used algorithms for sentiment analysis and favoured by many researchers due to their superior performance [8]. The evaluation of model performances compares their accuracy from the aspect of training data sizes, N-gram features and algorithms.

A. Feature Extraction

In the feature extraction stage, uni-gram, bi-gram and tri-gram are considered in the domain of N-gram features [6]. As the most frequently used method in text classification, feature extraction using uni-grams process each word within

the sentence individually. Phrases of two and three words are extracted as Bi-gram and tri-gram features for contrasts.

Hashtags and reply/mention tags within each tweet are used as additional features in that sometimes they contain strong subjective opinions. For example in the case of sentiment towards gun use, #2ndamendment may indicate pro-gun views on the use of guns whereas #guncontrol may suggest more neutral or anti-gun views. Similarly, @NRA may be a strong indicator of emotional text, albeit vague in terms of polarity.

B. Modelling

Eight predictive models are developed using the training data set obtained from machine learning approaches, namely Support Vector Machine (SVM), Naïve Bayes (NB), Maximum Entropy (ME), Tree, Bagging, Boosting, Random Forest (RF) and Neural Network (NN). With the exception of Nave Bayes (NB), which is developed with e1071 package [9], all the other classifiers are built with RTextTools [10], a machine learning library for text classification.

SVM uses kernels to find a hyperplane that divides data into different categories with the maximum margin [11, chapter 9]. In RStudio this model is built with the *train_model* function in RTextTools by calling the SVM algorithm from e1071.

NB classifier is a simple probabilistic classifier applied with Bayes Theorem with the assumption that all features are independent to each other [12]. Although this assumption may not be realistic in real-world situations, studies show that it outperforms other models for certain problems [13]. This model is built with the NB function in e1071.

ME is a classification method that generalises logistic regression to a multi-class problem and predicts the probabilities of different possible results. This algorithm is used as a contrast to NB, because of their opposite assumptions about the relationships between features. When the conditional independence assumption of NB fails in a certain case, ME might potentially yield a better result. ME algorithm is called from maxent [14] via the train model function in RTextTools.

A classification tree is a predictive model using decision tree learning through a process of binary recursive partitioning. This is a popular approach for classification problems in data mining [11, chapter 8]. This is the foundation of other tree-based models that will be discussed later, including bagging, random forest and boosting. Tree algorithm is called from tree [15] via the train model function in RTextTools.

Bagging, also called bootstrap aggregating is an ensemble algorithm used in machine learning to improve the accuracy and stability of an existing model. In this case a bagged classification tree is trained with ipred [16] and RTextTools.

Similar to bagged tree, RF is another ensemble learning algorithm. Instead of fully growing each binary tree, RF controls the number of features to search over in its pursuit of the best split for each tree. Using 200 trees, RF algorithm is called from randomForest [17] via the *train_model* function in RTextTools.

To reduce the bias of a predictive model against incorrectly classified data, boosting algorithms aim to build a strong

⁵<http://shiny.rstudio.com>

⁶<https://developers.google.com/chart/>

TABLE I
OUTPUT OF SAMPLE ACCURACY GENERATED THROUGH 10-FOLD CROSS VALIDATION USING UNI-GRAM FEATURE.

Training Data Size	SVM	ME	Tree	Bagged Tree	Boosted Tree	RF	NN	NB
1000	60%	9.3%	56.8%	56.9%	84.4%	60.4%	50.3%	29.1%
2000	68.6%	7.2%	62.4%	65.3%	82.5%	68.6%	59.8%	40.3%
3000	73.5%	6.2%	65.6%	70.2%	84%	73%	66.4%	44.4%
4000	83.2%	5.2%	78.1%	80.6%	87.9%	83%	72.9%	43.7%
5000	84.5%	5.3%	78.3%	88.9%	88.5%	91.8%	78.8%	41.6%

TABLE II
OUTPUT OF SAMPLE ACCURACY GENERATED THROUGH 10-FOLD CROSS VALIDATION USING A TRAINING DATASET OF 5000 TWEETS.

N-grams	SVM	ME	Tree	Bagged Tree	Boosted Tree	RF	NN	NB
Uni-gram	84.5%	5.3%	78.3%	88.9%	88.5%	91.8%	78.8%	41.6%
Bi-gram	53.7%	2.3%	50.9%	53.7%	96.5%	52.9%	53.5%	41.3%
Tri-gram	44.2%	37.4%	43.9%	44.4%	99.6%	44.3%	44.2%	N/A

classifier on a weak one by iteratively adding weights to misclassified data [11, chapter 8]. Logit boosting algorithm from caTools [18] is employed here to build a boosted tree.

NN performs non-linear statistical modelling and is robust against noise and generalisation [19]. In this research we apply NN for text categorisation, which is seldom done. NN algorithm is called from nnet [20] via the *train_model* function in RTextTools.

C. Evaluation

Table I compares model performance under 40 conditions generated by different training data sizes of 1000, 2000, ..., 5000 tweets and algorithms using the uni-gram feature. Model performances enhance along with larger training data set, with the only exception of ME. The top four performing models from this evaluation are highlighted and chosen for classifying the data set - the best performing model is RF with nearly 92% accuracy, followed by Bagged Tree and Boosted Tree which have almost similar performances and the SVM which has almost 85% accuracy.

As shown in Table II, model performances are not improved by increasing N-gram features, with the exception of Boosted Tree reaching over 99% accuracy with tri-grams. The accuracy of ME using tri-gram feature, which is the other exception, and is nearly 20 times higher than using bi-gram. There is an ongoing debate on whether N-gram feature can effectively enhance model performance or not [12]. On one hand, uni-gram features cover more data where as bi-gram and tri-gram features better capture patterns of expressions. In this research, the uni-gram features are clearly the most effective and it is believed that the choice of N-gram features should depend on the specific corpus. When selecting well-performing models, the two exceptions above are regarded as special cases that cannot represent the general trend. Therefore, the four highlighted best classifiers in Table I, namely RF, Bagged Tree, Boosted Tree and SVM, are accepted for classifying the test data, the results of which will be compared visually in the web application.

Although NB is strongly recommended in [21] as a better choice for snippet sentiment tasks than SVM, it fails in this

case. Two factors might account for this result: first, the data sets used in other studies are usually a mixture of topics, such as movie reviews; second, the snippet sentiment analysis explores the short version of documents, which contain much less informal languages than tweets. Given the fact that this research focuses on the public sentiment towards one specific topic extracted from tweets, it is possible that so-called strong classifier such as NB could actually fail. NB using tri-gram feature is not available due to the lack of decisive phrases.

IV. TWITTER ANALYSIS OF SANDY HOOK ELEMENTARY SCHOOL SHOOTING

In this paper, we analyse the tweet corpus purchased from GNIP surrounding the Sandy Hook elementary school shootings of Friday, 14 December, 2012 at 0940 local (14:40 GMT) in Newtown, Connecticut, USA and visualise the results as a web application which is available at <http://www.gunsontwitter.com>.

A. Analysing the Tweets

The acquired Twitter corpus contains 700,437 tweets for a 40 day period from Friday, December 7 2012, 00:00:01 GMT until Tuesday, January 15 2012, 23:59:59 GMT. The relevant tweets were obtained from GNIP by applying a number of filter rules. The rules can incorporate a number of keywords (for example, Sandy Hook, Newtown and guns), phrases (for example, gun violence and firearm magazine), hashtags (for example, #SandyHook and #CTshooting) and reply/mention tags (for example, @sandyhook and @SHWeChooseLove) along with a variety of filters such as the country of origin of the tweet (country_code:us), language (lang:en) used for tweeting and a means to specify that the tweet is or is not a repost (-is:retweet) as no retweets were selected).

During pre-processing relevant information such as the tweets, date, time and location were extracted. A golden training data set of 5000 tweets were manually selected and classified on the basis of the following three sentiments towards the event

- 1) Pro-gun: favouring the freedom of gun ownership and against gun control. For example, “The only thing that stops

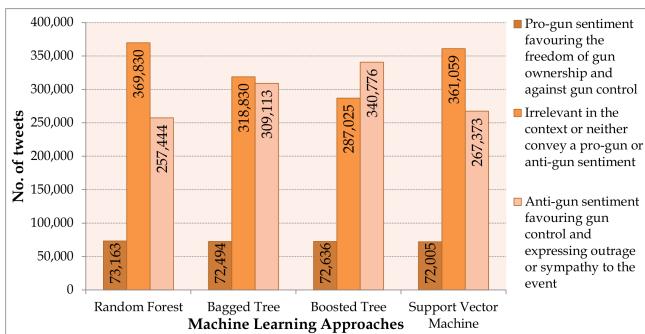


Fig. 2. Number of classified tweets from December 7 2012, 00:00:01 GMT to January 15 2012, 23:59:59 GMT

a bad guy with a gun, is a good guy with a gun" or "Gun control isn't gonna help. God bless the 2nd amendment! #sorrynotsorry".

- 2) Anti-gun: favouring gun control and express outrage and sympathy to the event. For example, "I cannot fathom how one could be so vile that they open fire on children or anyone. We need STRICT gun controls. #Newtown" or "BAN GUNS!!! Let our children be safe".
- 3) Neutral: irrelevant in the context or neither convey a pro-gun or anti-gun sentiment. For example, "Obama is possibly visiting Newtown" or "Not sure if gunshot or firework".

The four machine learning approaches that demonstrated best performance during evaluation in Section III, namely RF, Boosted Tree, Bagged Tree and SVM, were used to train models with the training data and then applied to classify the remaining tweets.

Figure 2 highlights the numbers of classified tweets using the four selected machine learning approaches. Pro-gun tweets account for about 10% of the data set regardless of approach chosen. The major difference between the approaches is in the number of neutral and anti-gun tweets. In the evaluation stage, a difference in accuracy by 10% was observed between the RF and SVM approaches. However, both these approaches assign over half of the tweets into the neutral class. The bagged tree model classifies nearly the same volume of tweets as neutral and anti-gun.

Figure 3 summarises the most frequently used hashtags and reply/mention tags in the data set. Although some of these tags do not convey obvious sentiments they act as indicators of emotions to some extent. For example #guncontrol and @BarackObama usually appear in anti-gun tweets while #NRA is more favoured in tweets that are more pro-gun and advocate gun ownership and gun rights. It would seem that there is more neutral sentiment conveying reply/mention tags (for example @justinbieber or @machinegunkelly).

B. Interactive Visualisation

An interactive web application (app), which is available on <http://www.gunsontwitter.com>, was developed to display multiple dimensions of the data. Data on population⁷ and

percentage of gun ownership (as of 2007)⁸ in each state were incorporated to add value to the visualisation. The following sections demonstrate the three visualisation techniques.

1) *Motion Chart*: Figure 4 is an example of the different visualisations possible using motion chart. For these figures the RF approach is selected. Figure 4a is a bubble view with six variables visualised simultaneously: X-axis for numbers of neutral tweets, Y-axis for public sentiment scores balanced by data volume and population ($PGPSS_3$), bubbles for the states, sizes of bubble for the populations of the states, colours of bubbles for the gun ownership as percentage of population and the slider at the bottom for date. In this chart for December 7 2012 (one week before the shooting event), larger the population of the state more pro-gun is the sentiment for gun use. In contrast, states that have a large volume of registered gun owners show little pro-gun sentiment.

A trail feature can be set by the user for viewing the variation of any variable over a time period. For example, refer Figure 4b. Texas is selected, and an overview of its variation during the 40-day period is seen on the chart. The highest $PGPSS_3$ appears when the number of neutral tweets increased to its top level.

Figure 4c shows the embedded bar chart view provided by the motion chart. Five groups of statistics are presented: numbers of pro-gun tweets on X-axis, volumes of tweet on Y-axis, states as bars, populations in the colours of bars and dates on the slider. In this example, the results of tweets on December 16 2012 are presented. The majority of tweets come from states with large population except Connecticut standing out in the middle of X-axis. This is probably because the shooting happened there and people were more concerned.

Figure 4d is the line graph view of the motion chart. Four variables are available on this view: time series on X-axis, numbers of anti-gun tweets on Y-axis, states shown in lines and public sentiment scores balanced by volumes of data ($PGPSS_2$) represented as the colour of the lines. This graph compares the statistics at the state-level and cannot provide a holistic view at the country-level. Therefore, a different line graph is generated as the second visualisation technique to analyse the results of U.S. at the country-level.

2) *Line Graph*: Figure 5 presents the visualisation using line graph. These results are presented for the Bagged Tree model. The numbers of tweets in the three sentiment classes can be explored either on a daily (Figure 5a) or an hourly (Figure 5b) basis. Zooming options are realised through a series of buttons at the top left corner, from one hour to the maximum time period. The time frame at the bottom is designed to customise the view a certain period. By default the line graph displays the results for the 40-day period. These graphs highlight that during the one-week period past the shooting, the number of tweets increase considerably regardless of the sentiment (pro-gun or anti-gun) with peaks observed on December 14 2012. An interesting fact is that although

⁷<https://www.census.gov/popest/data/state/totals/2012/>

⁸<http://usliberals.about.com/od/Election2012Factors/a/Gun-Owners-As-Percentage-Of-Each-States-Population.htm>

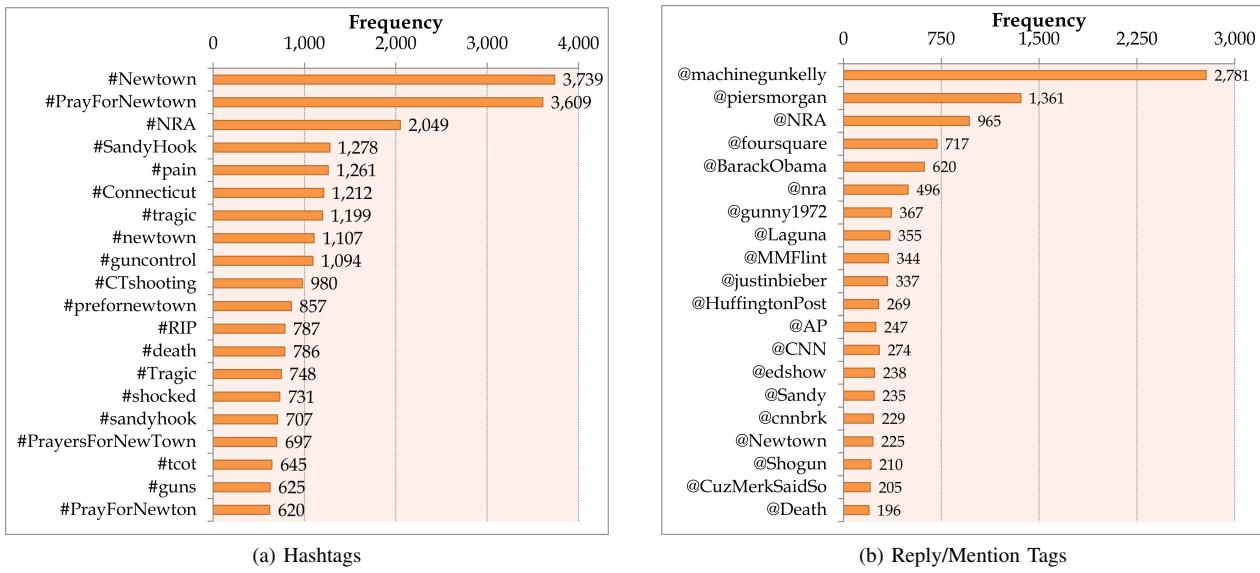


Fig. 3. Top 20 tags frequently used in tweets from December 7 2012, 00:00:01 GMT to January 15 2013, 23:59:59 GMT

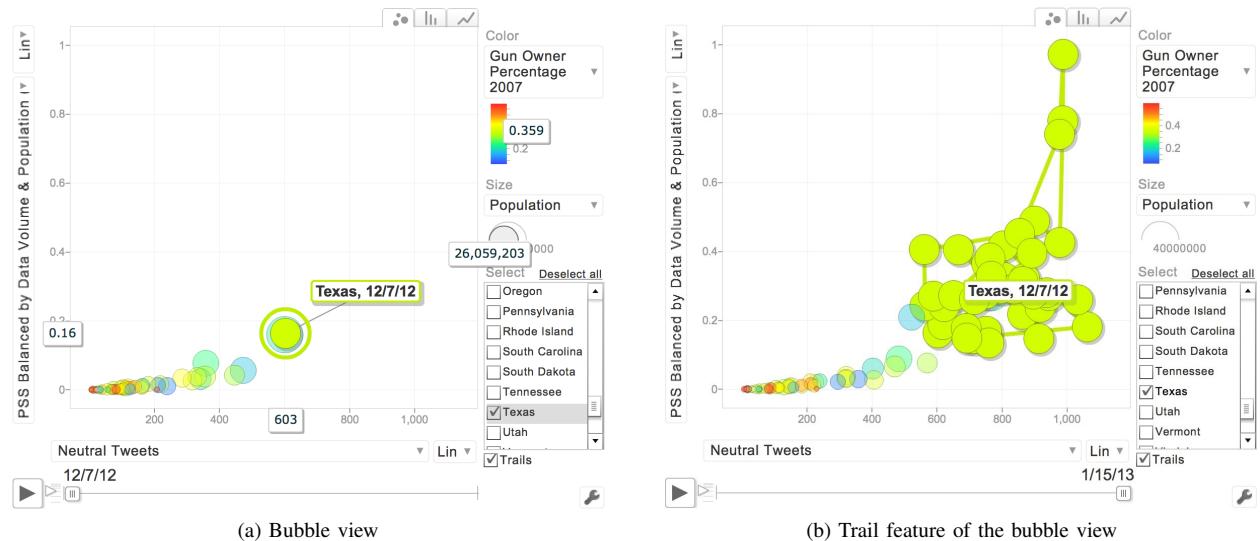


Fig. 4. Motion chart of the interactive web application

anti-gun sentiment was normally stronger than the pro-gun sentiment, on December 19 2012 the anti-gun sentiment is as low as the pro-gun sentiment. This five-day period (from December 14 to 19) could be important for using public opinion as an instrument for facilitating policy making.

3) *Geographic Map:* The coordinates of each tweet (latitude and longitude) were used to categorise the tweets from each state. This was facilitated using four R packages, namely regos [22], sp [23], maps [24] and maptools [25]. The coordinate information was used to generate a geographic visualisation as shown in Figure 6 and Figure 7.

Figure 6 presents three maps based on $PGPSS_1$, $PGPSS_2$ and $PGPSS_3$ for the 40-day period. $PGPSS_3$ is the best estimation since it is corrected for both volume of data and population. $PGPSS_1$ may not be a good indicator since it only takes the ratio of pro-gun and anti-gun tweets into account. In addition, the percentage of gun ownership appears in a floating box when the cursor hovers over a state. By default the geographic map displays $PGPSS_3$ over the whole 40-day period until the user chooses dates of interests through the date picker. For example, refer Figure 7. By comparing Figure 6c and Figure 7c, the three states that have strong pro-gun sentiment are California, Texas and New York.

V. DISCUSSION

The Sandy Hook school massacre was a deeply distressing and tragic event. It raised issues related to gun ownership and mental health. The mental health issue is not addressed by our data but we note that others who have carefully considered the issue [26] conclude that “notions of mental illness that emerge in relation to mass shootings frequently reflect larger cultural stereotypes and anxieties...”. The focus thus logically returns to gun ownership and access. High profile events typically evoke a huge and often opinionated social media response [27], [28], [29]. In instigating the analysis our prior assumption was that there would be a peak of anti-gun sentiment expressed in the hours immediately following the shooting. This did indeed turn out to be the case. We also assumed that after a period of time there may be a pro-gun response; that this response may come in reply to calls for greater gun control and would be seen by those involved as a reassertion of their second amendment rights. Our assumption was that this response may have occurred after about 7 to 10 days and if so we were interested in delineating the period between the anti-gun and pro-gun spikes as this perhaps defined a period during which preventative political action could be initiated.

Surprisingly, at least to us, the pro-gun spike in sentiment was not delayed but actually occurred on the day of the killings. We do not assume that this was a heartless response to a truly shocking and distressing event. It is quite possible that some of those expressing pro-gun sentiments were unaware that there had been fatalities at the time they tweeted. For good operational and practical reasons the fact that there had been so many deaths did not start to appear in the media until quite some hours after the event.

It is also interesting to note that whilst anti-gun sentiment peaked and then quickly fell to pre-shooting levels, pro-gun sentiment ran at an elevated level for some time. Again we think this probably reflects a genuine and deeply ingrained difference in opinion between individuals about the role of guns in causing or preventing such mass shooting events.

Indeed we think it is likely that pro-gun tweeters on the day of the killings did so because they genuinely believe that the event could have been avoided had members of staff been armed. A full discussion of the evidential basis of any such presumed belief or indeed of its more general moral, social and educational consequences is beyond the scope of this paper. State level analysis showed considerable variation in response. The highest gun owning states were not the most pro-gun in this situation. This may be because reasons for gun ownership vary considerably. Some traditional hunting states with high level of gun ownership demonstrated relatively low levels of pro-gun sentiment. California, Texas and New York demonstrated high levels of pro-gun sentiment. A full exploration of this interesting observation is beyond the scope of the data at the heart of this paper.

VI. RELATED WORK

Sentiment analysis of social media is an important avenue of research in the area of natural language processing. Previous research includes document-level analysis of product reviews [30] to term-level inspection on the polarity of words [31] [32]. Since tweets are character limited sentence-level classifications are frequently adopted to extract public sentiment. Classifying tweets is challenging due to the unique nature of micro-blogging with its frequent use of informal and colloquial language including slang and emoticons. Although there are different approaches for classifying tweets, there is no consensus on the best solution. In natural language processing of tweets, engineering linguistic features and automating text categorisation are two important tasks.

A. Linguistic Features

Twitter sentiment has been explored from the traditional linguistic perspectives and extracting N-gram features is a baseline method among researchers [6]. Uni-gram is the most common feature used and is reported to result in good predictive models with accuracy around 70%-80% [33] [34]. Bi-gram and tri-gram features are claimed to be effective in improving model performances. However, research suggests that uni-grams are better when building classifiers [12].

Other features such as Part-Of-Speech (POS) tags [7] and the position of words in text [12] are strong indicators of emotional text [13]. However, experiments show that POS tags may not be useful for sentiment analysis of micro-blogs [6]. Hash-tags and emoticons are typical contents in Twitter that indicate latent sentiment which are made use of in the process of automatically generating training data [36] [37].

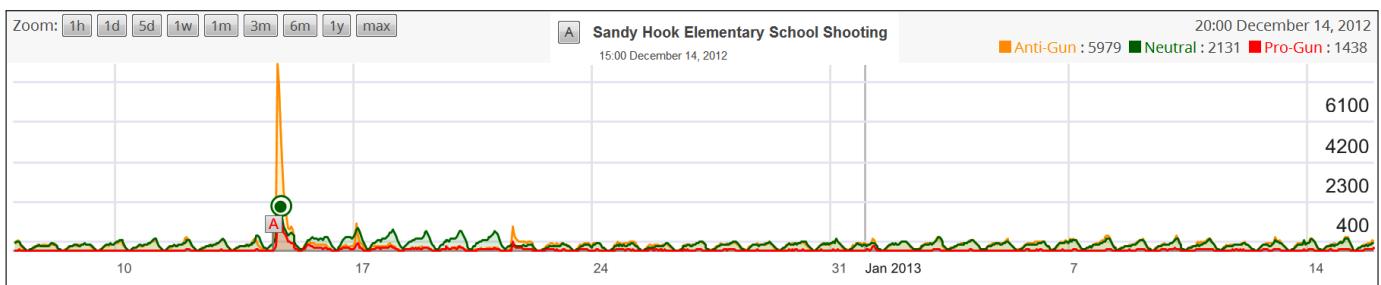
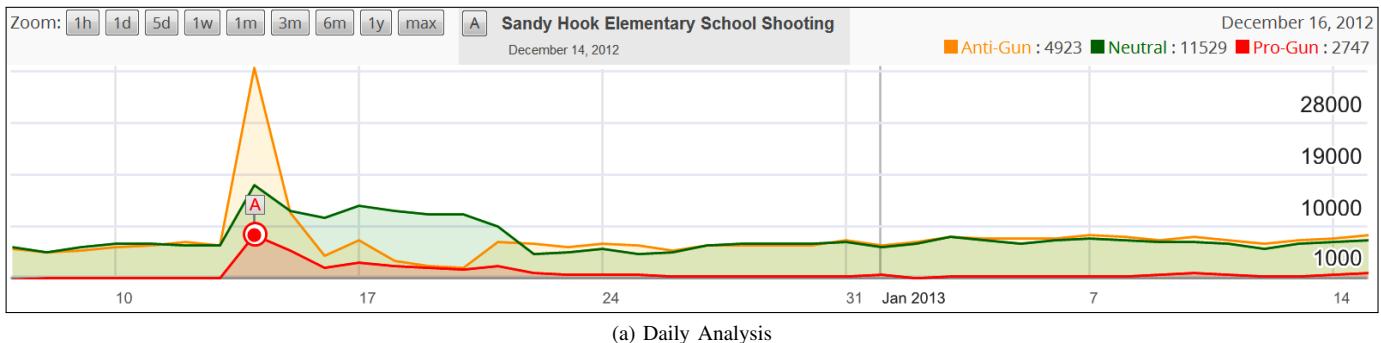


Fig. 5. Line graph of the interactive web application

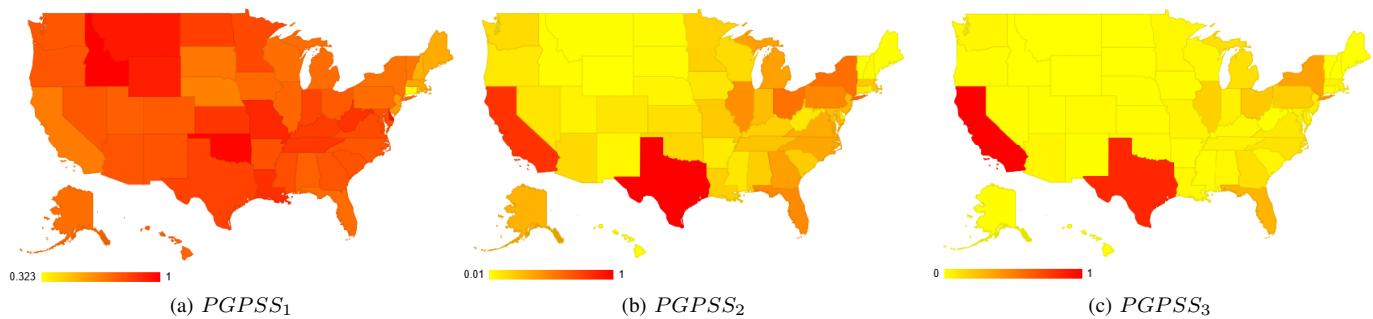


Fig. 6. Geographic map with customised PGPSS for December 7 2012 to January 15 2013

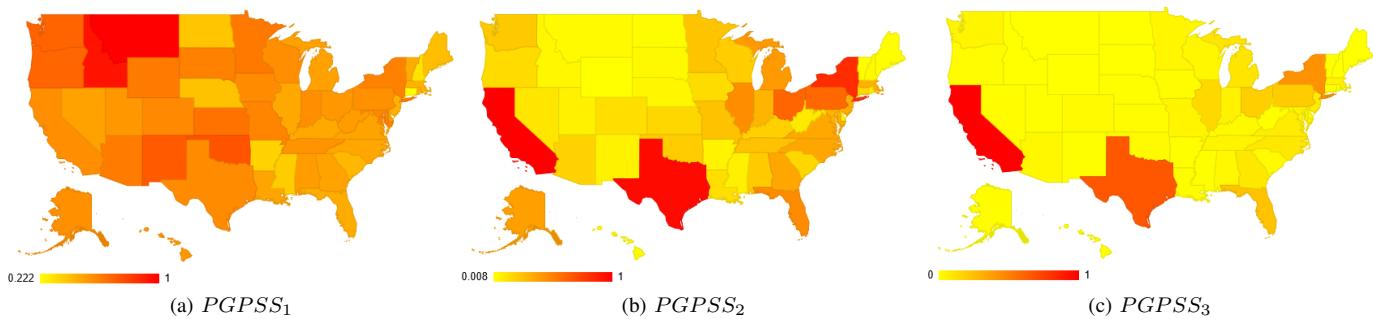


Fig. 7. Geographic map with customised PGPSS for December 13 2012 to December 15 2012

B. Automatic Text Categorisation

Algorithms for automated classification include dictionary-based and machine learning approaches. Dictionary-based approaches [38] [40] compare tweets against lexicons of existing dictionaries. By counting frequencies of the words matched with the chosen dictionary, a tweet is assigned a sentiment based on the prevalent emotion in the text. This approach is disadvantageous in that the acquired lexicon is domain independent and therefore it is impossible for a dictionary-based algorithm to capture different contexts. Another drawback of this approach is that it has to ignore the informal language used in tweets, which could probably stand for strong emotions. For example, words with repeated letters such as ‘nooooooo’ convey a stronger emotion than ‘no’. To overcome these disadvantages, machine learning based approaches provide the possibility of enhanced utility.

Machine learning approaches can be incorporated in building classifiers and can produce accuracy comparable to human experts [41]. However, the best classifier depends on the specific case study it is employed for; SVM and NB are popularly used owing to their average accuracy of above 80% regardless of features selected [7]. Other algorithms such as Conditional Random Field (CRF) and Maximum Entropy (ME) are found to be less effective for tweets analysis [13] [7].

In addition to the aforementioned standard machine learning techniques, adaptive classifiers have been built to improve performance. For example, the NBSVM, an SVM classifier trained with NB features, has better performance than SVM [21]. Similarly, an interpolation method [34] can be employed to modify the NB classifier with a generative model of words given semantic concepts. Hybrid classification has been proposed as an effective technique to combine a rule-based classification and machine learning approach [42].

C. Challenges

Although functional techniques have been developed in the area of using machine learning for sentiment analysis, there continues to be three challenges that we seek to address in this paper. Firstly, there is a lack of evaluation of various machine learning approaches in the same Twitter analysis context. For example, the so-called best classifiers, which were trained for categorising positive and negative text, demonstrated poor performance with three classes (positive, negative and neutral) [7]. Meanwhile, approaches such as Neural Network (NN) and Boosting in sentiment analysis are scarcely investigated. Hence, we believe it is essential to evaluate different approaches in the same context in order to identify the best approach in any given context.

Secondly, an important social issue, namely gun violence, has not been investigated for understanding public sentiment. Twitter analysis has been employed for understanding complex phenomena in the areas of politics [38], disaster management [33] and public health [43]. Gun violence is an important and challenging topic in public health and understanding

the temporal and geographic patterning of pro-gun and anti-gun sentiments is important in terms of framing and timing political initiatives aimed at reducing gun related deaths. This is particularly true in terms of mass shootings and in this paper, we seek to use machine learning based Twitter analysis to explore pro-gun and anti-gun sentiments in relation to the Sandy Hook School shootings.

Thirdly the use of machine learning techniques for the analysis of large data sets is in general still confined to the academic discipline of Computer Science (CS). In seeking to bridge the disciplines of Computer Science and Public Health in producing this paper we are seeking to further the prospects of such trans-disciplinary research by promoting the benefits that machine learning can offer.

Hence, in this study, the above three challenges related to the evaluation of machine learning techniques, the application of Twitter analysis to gun violence and the accessibility of machine learning techniques for non-CS experts are addressed through the Sandy Hook Elementary School shootings case study. Tweets expressed public sentiment in terms of pro-gun or anti-gun attitudes were examined over forty days. These sentiments were extracted from tweets using machine learning techniques and visualised.

VII. CONCLUSIONS

The public health implications are firstly methodological. The disciplines of Computer Science and Public Health are brought together on the important and emotive issue of reactions to a mass school shooting which is a useful exercise that yields multidisciplinary dividends.

Furthermore in this paper, we have evaluated a number of machine learning approaches and identified those most suitable to classifying public sentiment towards gun violence in light of the Sandy Hook school shooting. We have shown that it is possible to analyse a large body of social media data using machine learning in a reliable and replicable way by employing a methodology to collect, train and classify tweets. Public sentiment is captured for different time scales and by state can be interactively visualised on an online tool available at <http://www.gunsontwitter.com>.

Sentiment analysis shows a peak of anti-gun feeling on the day of the Sandy Hook school shooting which quickly falls to pre-event levels. More surprisingly the analysis shows a peak of pro-gun sentiment on the day of the shooting that is sustained at an elevated level for a number of days. These findings suggest that a discernible minority of the US population see high levels of public gun ownership as part of the solution rather than part of problem. This is in spite of the objective evidence that the US with its generally liberal gun ownership rules suffers a uniquely high incidence of school shootings [44] and that high level of public firearms ownership increase firearm related deaths both of the public [45] and of law enforcement officers [46].

More work is required to understand the complexities of public opinion on this important issue but social media analysis appears to have a useful contribution to make. In the future,

we aim to apply machine learning techniques to additional gun violence related events. This will enhance our knowledge of public reaction and delineate a time period during which preventative political action can be instigated.

REFERENCES

- [1] S. Sedensky, "Report of the State's Attorney for the Judicial District of Danbury on the Shootings at Sandy Hook Elementary School and 36 Yogananda Street, Newtown, Connecticut on December 14, 2012, State of Connecticut," November 2013.
- [2] J. M. Shultz, G. W. Muschert, A. Dingwall, and A. M. Cohen, "The Sandy Hook Elementary School Shooting as Tipping Point," *Disaster Health*, vol. 1, no. 2, pp. 65–73, 2013.
- [3] S. Li, G. Zhou, Z. Wang, S. Y. M. Lee, and R. Wang, "Imbalanced Sentiment Classification," in *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pp. 2469–2472, 2011.
- [4] P. Zhang, "Model Selection via Multifold Cross Validation," *The Annals of Statistics*, pp. 299–313, 1993.
- [5] M. Gesmann and D. de Castillo, "googleVis: Interface Between R and the Google Visualisation API," *The R Journal*, vol. 3, pp. 40–44, December 2011.
- [6] E. Koulopis, T. Wilson, and J. Moore, *Twitter Sentiment Analysis: The Good the Bad and the OMG!*, pp. 538–541. AAAI Press, 2011.
- [7] A. Go, R. Bhayani, and L. Huang, "Twitter Sentiment Classification Using Distant Supervision," *CS224N Project Report, Stanford*, pp. 1–12, 2009.
- [8] V. D. Nguyen, B. Varghese, and A. Barker, "The Royal Birth of 2013: Analysing and Visualising Public Sentiment in the UK Using Twitter," in *Proceedings of the IEEE International Conference on Big Data*, pp. 46–54, 2013.
- [9] E. Dimitriadou, K. Hornik, F. Leisch, D. Meyer, A. Weingessel, and M. F. Leisch, "The e1071 Package," *Department of Statistics (e1071), TU Wien*, 2006.
- [10] T. P. Jurka, L. Collingwood, A. E. Boydston, E. Grossman, and W. van Atteveldt, "RTextTools: Automatic Text Classification via Supervised Learning," *R package version 1.3.9*, 2012.
- [11] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*. Springer, 2013.
- [12] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs Up?: Sentiment Classification Using Machine Learning Techniques," in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pp. 79–86, Association for Computational Linguistics, 2002.
- [13] A. Pak and P. Paroubek, "Twitter as a Corpus for Sentiment Analysis and Opinion Mining," in *Proceedings of the Seventh conference on International Language Resources and Evaluation*, 2010.
- [14] T. P. Jurka, "maxent: An R Package for Low-memory Multinomial Logistic Regression with Support for Semi-automated Text Classification," *The R Journal*, vol. 4, no. 1, pp. 56–59, 2012.
- [15] B. Ripley, "Classification and Regression Trees," *R package version 1.0-35*, 2005.
- [16] A. Peters and T. Hothorn, "ipred: Improved Predictors," *R package version 0.9-3*, 2013.
- [17] A. Liaw and M. Wiener, "Classification and Regression by randomForest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.
- [18] J. Tuszyński, "caTools: Tools: Moving Window Statistics, GIF, Base64, ROC AUC, etc," *R package version 1.17*, 2014.
- [19] I. Basheer and M. Hajmeer, "Artificial Neural Networks: Fundamentals, Computing, Design, and Application," *Journal of Microbiological Methods*, vol. 43, no. 1, pp. 3–31, 2000.
- [20] W. N. Venables and B. D. Ripley, *Modern Applied Statistics with S*. New York: Springer, fourth ed., 2002.
- [21] S. Wang and C. D. Manning, "Baselines and Bigrams: Simple, Good Sentiment and Topic Classification," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pp. 90–94, Association for Computational Linguistics, 2012.
- [22] R. Bivand and C. Rundel, "rgeos: Interface to Geometry Engine - Open Source (GEOS)," *R package version 0.3-6*, 2014.
- [23] E. J. Pebesma and R. S. Bivand, "Classes and Methods for Spatial Data in R," *R News*, vol. 5, pp. 9–13, November 2005.
- [24] R. Brownrigg, "maps: Draw Geographical Maps," *R package version 2.3-7*, 2014.
- [25] R. Bivand and N. Lewin-Koh, "maptools: Tools for Reading and Handling Spatial Objects," *R package version 0.8-30*, 2014.
- [26] J. M. Metzl and K. T. MacLeish, "Mental Illness, Mass Shootings, and the Politics of American Firearms," *American Journal of Public Health*, vol. 105, no. 2, pp. 240–249, 2015.
- [27] P. Burnap, M. L. Williams, L. Sloan, O. Rana, W. Housley, A. Edwards, V. Knight, R. Procter, and A. Voss, "Tweeting the Terror: Modelling the Social Media Reaction to the Woolwich Terrorist Attack," *Social Network Analysis and Mining*, vol. 4, no. 1, pp. 1–14, 2014.
- [28] G. Lotan, E. Graeff, M. Ananny, D. Gaffney, I. Pearce, and danah boyd, "The Arab Spring— The Revolutions Were Tweeted: Information Flows during the 2011 Tunisian and Egyptian Revolutions," *International Journal of Communication*, vol. 5, no. 0, 2011.
- [29] R. Procter, F. Vis, and A. Voss, "Reading the Riots on Twitter: Methodological Innovation for the Analysis of Big Data," *International Journal of Social Research Methodology*, vol. 16, no. 3, pp. 197–214, 2013.
- [30] B. Pang and L. Lee, "Opinion Mining and Sentiment Analysis," *Foundations and Trends in Information Retrieval*, vol. 2, pp. 1–135, 2008.
- [31] A. Esuli and F. Sebastiani, "Sentiwordnet: A Publicly Available Lexical Resource for Opinion Mining," in *Proceedings of the International Conference on Language Resources and Evaluation*, vol. 6, pp. 417–422, 2006.
- [32] S. Mohammad, C. Dunne, and B. Dorr, "Generating High-coverage Semantic Orientation Lexicons from Overtly Marked Words and a Thesaurus," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2*, pp. 599–608, 2009.
- [33] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors," in *Proceedings of the 19th international conference on World Wide Web*, pp. 851–860, 2010.
- [34] H. Saif, Y. He, and H. Alani, "Semantic Sentiment Analysis of Twitter," in *Proceedings of the 11th International Conference on The Semantic Web - Volume Part I*, pp. 508–524, 2012.
- [35] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau, "Sentiment Analysis of Twitter Data," in *Proceedings of the Workshop on Languages in Social Media*, pp. 30–38, 2011.
- [36] A. Bifet and E. Frank, "Sentiment Knowledge Discovery in Twitter Streaming Data," in *Proceedings of the 13th international conference on Discovery Science*, pp. 1–15, 2010.
- [37] D. Davidov, O. Tsur, and A. Rappoport, "Enhanced Sentiment Learning Using Twitter Hashtags and Smilies," in *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pp. 241–249, 2010.
- [38] A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpe, "Election Forecasts with Twitter: How 140 Characters Reflect the Political Landscape," *Social Science Computer Review*, p. 0894439310386557, 2010.
- [39] J. Bollen, H. Mao, and X. Zeng, "Twitter Mood Predicts the Stock Market," *Journal of Computational Science*, vol. 2, no. 1, pp. 1–8, 2011.
- [40] S. Asur and B. A. Huberman, "Predicting the Future with Social Media," in *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pp. 492–499, 2010.
- [41] F. Sebastiani, "Machine Learning in Automated Text Categorization," *ACM computing surveys*, vol. 34, no. 1, pp. 1–47, 2002.
- [42] R. Prabowo and M. Thelwall, "Sentiment Analysis: A Combined Approach," *Journal of Informetrics*, vol. 3, no. 2, pp. 143–157, 2009.
- [43] M. J. Paul and M. Dredze, "You Are What You Tweet: Analyzing Twitter for Public Health," in *Proceedings of the 5th International AAAI Conference on Weblogs and Social Media*, pp. 265–272, 2011.
- [44] N. Böckler, T. Seeger, P. Sitzer, and W. Heitmeyer, *School Shootings: International Research, Case Studies, and Concepts for Prevention*, ch. School Shootings: Conceptual Framework and International Empirical Trends, pp. 1–24, 2013.
- [45] M. Siegl, Y. Negussu, S. Vanhoi, J. Pleskunas, C. S. Ross, and C. King 3rd, "The Relationship Between Gun Ownership and Stranger and Nonstranger Firearm Homicide Rates in the United States, 1981–2010," *American Journal of Public Health*, vol. 104, no. 10, pp. 1912–1919, 2014.
- [46] D. I. Swedler, M. M. Simmons, F. Dominici, and D. Hemenway, "Firearm Prevalence and Homicides of Law Enforcement Officers in the United States," *American Journal of Public Health*, vol. 105, no. 10, pp. 2042–2048, 2015.

A Hybrid Approach to Population Construction For Agricultural Agent-Based Simulation

Peng Chen*, Tom Evans†, Michael Frisby‡ Eduardo Izquierdo* and Beth Plale*

*School of Informatics and Computing, †Department of Geography, ‡Indiana Statistical Consulting Center
Indiana University Bloomington

Email: chenpeng, evans, mfrisby, edizquie, plale@indiana.edu

Abstract—An Agent Based Model (ABM) is a powerful tool for its ability to represent heterogeneous agents which through their interactions can reveal emergent phenomena. For this to occur though, the set of agents in an ABM has to accurately model a real world population to reflect its heterogeneity. But when studying human behavior in less well developed settings, the availability of the real population data can be limited, making it impossible to create agents directly from the real population. In this paper, we propose a hybrid method to deal with this data scarcity: we first use the available real population data as the baseline to preserve the true heterogeneity, and fill in the missing characteristics based on survey and remote sensing datasets; then for the remaining undetermined agent characteristics, we use the Microbial Genetic Algorithm to search for a set of values that can optimize the *replicative validity* of the model to match data observed from real world. We apply our method to the creation of a synthetic population of household agents for the simulation of agricultural decision making processes in rural Zambia. The result shows that the synthetic population created from the farmer register can correctly reflect the marginal distributions and the randomness of survey data; and can minimize the difference between the distribution of simulated yield and that of the observed yield in Post Harvest Survey (PHS).

I. INTRODUCTION

Spatial agent-based modeling (ABM) has been proven to be beneficial to agricultural economics for its ability to represent interactions amongst heterogeneous actors, and to fully take into account the spatial dimension of agricultural activities [1]. While an actor (agent) can be an individual farmer, it is typical to consider the household as the basic unit of analysis in agricultural modeling [2], [3], [4]. An agent representing a household has characteristic variables (e.g. wealth, labor supply, preferences) and spatial locations (cells/patches). The key to good agricultural agent based modeling is to construct agents that can truly reflect the characteristics of a real population of households.

However, data about populations is often limited to census data, and in some cases administrative (farmer) register data, which refers to information collected at the district level about which farmers are planting which crops and in which fields in a growing season. This information generally contains a limited set of characteristic variables and thus is insufficient for creating the agents. One way to deal with this insufficient information (i.e., missing agent characteristic variables) is to integrate multiple available data sources and to derive from empirical data. A valuable practice is to use the available census data (and in some cases aggregated administrative

register data), combined with remote sensing data, and project that onto the creation of a synthetic population dataset, which is then used to construct agents [3], [4], [5], [6]. While this approach may allow heterogeneities between subgroups of the population to occur, it cannot capture the heterogeneities at individual (household) level. For instance, preserving the structure (sizes) and heterogeneities of subgroups results in synthetic data that may be statistically equivalent to real population (census or aggregated register data) [2], but still lack the household level granularity needed for agent-based simulations.

In this paper, we propose a novel method of simulating synthetic population data based on available real population data (i.e., individual level farmer register data), household survey data, and remote sensing data. The real population data serves as the baseline for the synthetic population. The variability needed within the missing variables, and the relationships between missing variables and known variables are both learned from the survey data, and then used to simulate the missing variables. To assign spatial locations to the synthetic population, the agricultural land data generated from remote sensing is disaggregated into raster form and allocated to the synthetic population.

While researchers in Geography often want to derive as many variables as possible from empirical data, there could always be variables needed that has no relevant real population data or survey data available. In other words, our proposed data simulation method can not derive variables that do not exist in the survey data. In these cases, it is possible to use an optimization process to derive a set of values for the missing variables that improve the *replicative validity* [7] of the model; that is, aiming to minimize the difference between the data generated by the simulation and data previously acquired from the real system. Nevertheless, classical optimization tools such as regression may not be effective in finding a suitable combination of missing variables due to the inherent complexity of the interactions within the model [8]. For this reason, Genetic Algorithms (GAs) have been previously used for model calibration [9], [8], [10], [11] with good results.

In this paper, we apply the microbial genetic algorithm [12] to the calibration of missing variables of household agents, and demonstrate that it can be used in conjunction with the data simulation method to create a synthetic population of agents. There are two challenges when applying a GA to an

agent-based model: 1) how to design a fitness function that can consider the behaviors of all the agents; and 2) how to handle the stochasticity in the simulation run. We address the former by using Kullback–Leibler divergence [13] that measures the distance between two distributions; and the latter by exposing the random number seed in the model as a parameter to be calibrated by GA.

We test our hybrid approach on an agricultural agent-based model that we developed for the Monze District, Zambia, an area that is approximately 1,866 square miles in size, with 53,491 households. Each agent in the model represents a household with characteristic variables (e.g., number of household members and area of cultivation) and decision making rules (based on the variable values). Available data includes: 1) an extensive household survey conducted in Southern Province of Zambia, 2) district-level farmer registry data and 3) the Post Harvest Survey (PHS) data. The extensive survey data was collected through surveys of 330 households and includes information on household size from which labor supply and food demand are calculated. The farmer registry is compiled by regional agricultural extension officers and consists of a census of all small-scale farmers in a particular district with basic attributes, such as the total area of the farm and the total area under cultivation in a particular year. The Post Harvest Survey data is the mechanism that the Zambian government uses to assess end of season crop production (i.e. crop yield). The PHS is a household-level survey conducted with a sample of households in the country each year.

Agent-based models are highly sensitive to how the initial set of agents are created within a model (i.e. how many agents and with what attributes) and the decision algorithms that govern agent behavior in the model. It is rare to have an empirical dataset that tells a modeler exactly how many agents there should be in a model and what attribute are needed s in order to represent a real-world scenario. In the case of our agricultural system in Zambia, we specifically need to know how many farmers there are in a particular area and how large their farms are in order to model farm-level agricultural production. No single dataset provides this information. Thus we have developed a hybrid data integration process drawing on the datasets above to initialize our farmer agents. Our hybrid draws on household attributes (household size) from our extensive survey data, land in cultivation from the farmer registry data and end of season agricultural yield from the Post-Harvest Survey data. From these three datasets, we are able to integrate these salient characteristics to create a set of farmer agents that enables us to run simulations for alternative climate scenarios.

We demonstrate that our approach is effective in integrating household survey data with the farmer register data, and in deriving an optimized set of values for agent variables based on PHS. It is our belief that the administrative register data provides the structure and heterogeneities of the real population at the individual level, which has never been done before to our best knowledge. The result shows that the simulated data can reflect the marginal distributions and the randomness

from the survey data, and that the set of values optimized for the missing variables can produce simulated production close to the observed PHS.

The remainder of the paper is organized as follows. Section II reviews related work. Section III introduces the proposed simulating method and Section IV describes the application of microbial genetic algorithm to agent-based model. Section V demonstrates the experiment of proposed hybrid approach on the Zambia agent-based model. We conclude the paper in Section VI with future work.

II. RELATED WORK

The existing work on creating household agents in ABMs for agricultural analysis [3], [4], urban planning [5] and urban disaster management [6] focused on decomposing aggregated demographic/administrative data. In environmental modeling, methods that create agents from survey data are often called parameterisation [14] and agent typology [15]. For example, Ralha et al. [16] use survey/sample data to create the agent typologies and then create a population of agents according to the distribution of agent types. In addition, there is research that maps disaggregated census data (like household level records) to various agent types [17], using techniques such as Regression Tree [18]. However, none of these methods leverages the relations from survey data to filling in missing variables in available real population data like our approach does. To our best knowledge, we are the first to integrate the real population data into the agent creation process.

Many agent-based models built in environmental science have a module that allocates land to agents. For example, Gaube et al. [19] and Ralha et al. [16] use a density/probability map to place households; Schouten et al. [20] and Murray-Rust et al. [21] introduce an auction/competition mechanism to allocate land to households. In our proposed method, we consider the spatial location an important variable of household agent and develop a land allocation algorithm that aims at forming natural farmer communities when placing the household agents.

Data imputation [22] focuses on missing variables inside a dataset. Our problem is different in that the administrative register data is a complete dataset, but it lacks some variables that are important to the agent-based simulations. Besides, most data imputation techniques work great when there is only a small percent of data missing, but in our case, the amount of missing data is two orders of magnitude larger than that of the known data. Lastly, data imputation is mostly followed by data analysis. Thus the design of data imputation techniques usually aims at supporting data analysis. However, in Agent Based Modeling (ABM), the set of agents is used to model the real world population.

Synthetic data generation and *data simulation* are terms usually referred to the creation of large population from a small data – whether it is an aggregated dataset [2] and/or a sample data [23], [24], [25]. Data generation/simulation based on aggregated data usually means to decompose the data while maintaining the same marginal distribution on each

dimension. Frazier and Alfons [26] utilize the linear regression model plus random errors, which is particularly similar to our method. However, instead of using aggregated data, our method leverages the large administrative register and the individual level survey data, which can better preserve the heterogeneity and the randomness.

Genetic Algorithms (GAs) can automatically search a parameter space, and thus they have been used to calibrate agent-based models [9], [8], [10], [11]. Calvez and Hutzler [9] summarize the specific difficulties related to the use of GA approach with agent-based models: the choice of fitness function, the stochasticity, and the computational cost. In this work, for agent variables that cannot be deducted from the available real world data, we apply the microbial GA [12] to find a set of values that can produce simulation outcome that is close to real world observations. We use the distance between the simulated outcome and real world observations as the fitness score; and we set the random number seed in the agent-based model and expose the random number seed as a parameter in the chromosome of GA to handle its stochasticity.

III. SIMULATION OF SYNTHETIC POPULATION

This section introduces our proposed method that simulates the agent variables from individuals' register data, survey data, and remote sensing data.

A. Simulating Household Characteristics from Survey Data and Real Population Data

The first step in the data simulation is to create a generalized linear model based on the relationship between a set of observed independent variables and an observed response variable in the survey data – while the same independent variables are known in the register data, the response variable is a missing variable in the register data. Generalized linear models [27] are extensions of traditional regression models that allow the mean to depend on the independent variables through a link function, and the dependent variable to be any member of a set of distributions called the exponential family (e.g., Normal, Poisson, Binomial). By using the generalized linear model we make two assumptions:

- 1) That there exists a strong correlation between the independent variables and the dependent variable(s). This can be tested with Likelihood Ratio Test (LRT) on a fitted model.
- 2) That the dependent variable(s) can be modeled using one of the distributions in the exponential family. This can be verified using a goodness of fit test.

For instance, when there is only one independent variable (X) and the dependent variable has a Poisson distribution, the fitted model can be represented as:

$$\log(E(Y|X)) = a + b * X \quad (1)$$

where Y denotes the dependent variable, $E(Y|X)$ denotes the expected value of Y given X , and a, b are the coefficients estimated during the model fitting process.

Once the generalized model is fitted, we apply it to the independent variables in the farmer register to predict the mean values of the missing variables. We then use the predicted mean value to randomly create values based on the distribution in the exponential family that we chose.

B. Simulating Household Spatial Locations by Allocating Agricultural Land to Household Agents

Spatial interactions amongst agents require the exact spatial locations of agents to be known. However, the spatial information contained in the real population data is generally given in aggregated form (e.g., by zones). To solve that, we take the remote sensing data that is classified into agricultural and non-agricultural land, disaggregate it into raster, and allocate the agricultural cells/patches to households.

We developed an algorithm shown in Fig. 1, to allocate the agricultural cells to households. Our land allocation algorithm first chooses a number of seed households in the procedure ALLOCATE_LAND_TO_HH. For each seed household H , it randomly selects an unallocated farmland patch SP and then invokes ALLOCATE_MANY that assigns to SH with P and extra unallocated farmland patch(es) within the maximum search radius s of P . Within ALLOCATE_MANY, ALLOCATE_ONE is invoked to actually allocate one farmland to a household, and mark the adjacent farmland as *tentative*. Once all seed households are assigned with enough farmland, ALLOCATE_LAND_TO_HH will continue to allocate farmland to the non-seed households. For each non-seed household SH , it finds a *tentative* unallocated farmland TSP and invoke ALLOCATE_MANY(SH, TSP). In this way, the households should be located close to each other to form communities.

IV. CALIBRATING AGENT VARIABLES WITH GENETIC ALGORITHM

Genetic Algorithm (GA) is a heuristic search that mimics the process of natural selection. It belongs to the larger class of evolutionary algorithms (EAs), which generate solutions to optimization problems using techniques inspired by natural evolution. While there are different variants of GA, the common underlying idea is the same: given a population of individuals and a fitness function, the properties of the individuals are mutated and altered in each generation and the best fitted individuals are preserved to the next generation to evolve an optimized solution. The Microbial Genetic Algorithm is a minimal GA that has the same functionality and efficacy as the standard GAs, but is simple to code and tune. We choose to implement the Microbial GA instead of using other off-the-shelf software for the simplicity and a better understanding of the calibration process.

The most creative and challenging parts of programming a GA are usually the problem-specific aspects, that is, the design of chromosome (a set of properties for each individual in the population) and its mutation/alternation process, and the fitness function (the fitness score is usually the objective value in the optimization problem being solved).

```

1: procedure ALLOCATE_ONE( $H, P$ )  $\triangleright H$ : household,  $P$ : patch
2:    $A \leftarrow$  area of farmland needed by  $H$ 
3:   if  $A >$  area of  $P$  then
4:      $occupiedRatio(P) \leftarrow 1$   $\triangleright P$  fully occupied by  $H$ 
5:   else
6:      $occupiedRatio(P) \leftarrow (A - 1)$   $\triangleright P$  partially occupied by  $H$ 
7:   end if
8:    $N \leftarrow$  neighbor farmland (in radius  $r$ ) of  $P$   $\triangleright r$  a global parameter of allocation radius
9:    $status(N) \leftarrow$  tentative seed patches
10: end procedure

11: procedure ALLOCATE_MANY( $H, P$ )  $\triangleright H$ : household,  $P$ : patch
12:   Invoke allocate_one( $H, P$ )
13:   repeat
14:      $searchRadius \leftarrow 700m$   $\triangleright$  starting from threshold value
15:      $SP \leftarrow$  randomly selected unoccupied farmland within  $searchRadius$  of  $P$ 
16:     if  $SP$  is not NULL then
17:       Invoke allocate_one( $H, SP$ )
18:     else
19:        $searchRadius \leftarrow (searchRadius + 100m)$   $\triangleright$  Expand the search area
20:     end if
21:     until  $H$  has been assigned enough farmland  $\vee$   $searchRadius == s$   $\triangleright s$  is global parameter of maximum search radius
22: end procedure

23: procedure ALLOCATE_LAND_TO_HH
24:    $i \leftarrow 1$   $\triangleright$  id of current HH to be allocated
25:   repeat
26:      $SH \leftarrow$  the  $i$ th household
27:      $status(SH) \leftarrow$  seed household
28:      $SP \leftarrow$  a randomly selected patch
29:     Invoke allocate_many( $SH, SP$ )
30:      $i \leftarrow (i + 1)$ 
31:   until  $i == numSeed$   $\vee$  no unoccupied land  $\triangleright numSeed$  is global parameter of total number of seed households created during initialization
32:   repeat
33:      $SH \leftarrow$   $i$ th household
34:      $TSP \leftarrow$  randomly selected patch so that  $status(TSP) ==$  tentative seed patch  $\wedge$   $occupiedRatio(TSP) == 0$ 
35:     Invoke allocate_many( $SH, TSP$ )
36:      $i \leftarrow (i + 1)$ 
37:   until  $i == numHHS$   $\vee$  there is no unoccupied land  $\triangleright numHHS$  is total number of households
38: end procedure

```

Fig. 1. Algorithm to allocate agricultural cells to households. Comments are denoted by right-pointing triangle.

The chromosome could be composed of properties that each represents a missing agent variable. There are different types of missing variables and thus each has to be treated differently:

- 1) For nominal variables, such as *soilType*, we represent them as integers, and mutate them randomly into any other possible values.
- 2) For continuous variables, such as *ratioOfLocalMaize*, we represent them as doubles, and mutate them using a Gaussian number generator.
- 3) For variables that follow a certain distribution, we expose its parameters as doubles and mutate them using a Gaussian number generator. For example, we assume that *plantingDate* follows a normal distribution and choose to fix its mean while changing its standard deviation.

Note that the mutation/alternation has to be within the value space of each variable, and if it results in a value that falls below or goes above the boundaries, we simply replace it with the boundary values.

Two simulation runs of the same agent-based model can generally bring slightly different results, due to the stochasticity of the model and the simulator. One solution is to simulate each model several times to improve the evaluation of the fitness function, which however greatly increases the number of simulations and thus the time to run the simulations. Calvez and Hutzler [9] address this issue by estimating the fitness of each model with one simulation at most generations, while simulate the model several times at each n generation. While their method reduces the number of simulations, it still needs to simulate one model several times. To completely avoid it, we propose to fix the random number seed in the agent-based model (and the platform) and expose it as a parameter to the genetic algorithm. In this way, the model becomes deterministic while the genetic algorithm can factor in the stochasticity of the model when searching for the best combination of parameters.

Lastly, agent-based systems or simulations are dynamic and often characterized by emergent phenomena, which complicates the measure of the fitness function. What has to be measured strongly influences the characteristics of the models obtained by the genetic algorithm. If the fitness function is not carefully chosen, the resulting models will be optimized for that specific fitness function. Since the data generated from agent-based model can be collected at individual level (e.g., the yield of each household agent) or aggregated level (e.g., total crop production), we could evaluate the fitness based on the individual level data or the aggregated data. For example, we can measure the distance between the simulated average yield of household agents and the observed average yield from PHS (aggregate level); or we can measure the difference between the distribution of simulated household yields and that of observed household yields in PHS (individual level). In this paper, we propose to use the Kullback–Leibler divergence to measure the difference between the distribution of simulated data and the distribution of observed data. This is because we

want the agents' behaviors to accurately reflect the variance in real households' crop production.

V. APPLICATION TO ZAMBIA FOOD SECURITY ABM

A. Zambia Crop Decision-making ABM

We apply our technique to the initialization of a spatially explicit ABM of intra-seasonal agricultural decision-making of households for the entire Monze District in Zambia. In our model, household agents make decisions biweekly based on a utility maximization approach [5] within the context of local institutional regimes (i.e., wards). The goal is to use this model to identify how climate change impacts adaptive capacity.

B. Survey Data and Farmer Register Data Cleaning

The original survey data and farmer register data are stored in Excel spreadsheet. We start with writing Python code to auto-correct the formatting errors and typos, and to extract the information into a MySQL database. However, additional cleansing has to be done to prepare the data for simulation. Cultivated area (*CultArea*) is expressed in hectares out to two decimal places in the survey data, but is frequently rounded in the farmer register. The rounding could be because people tend to answer with rounded values, which is called *response heaping* and has been extensively observed and studied [28]. To address this problem, we decided to round the variable of *CultArea* in both the survey data and the farmer register data. The rounding policy is described below:

- 1) If the value of *CultArea* is less than 1ha, we round it up to 1ha. This is to avoid rounding small values of *CultArea* to zero, since the data is collected from smallholder farmers (that each owns small-based plots of land).
- 2) For any other value of *CultArea*, we round it to the nearest integer value.

Incorrect records can exist in both the survey data and the register data. We identified and removed one record (out of 184) in the survey data with a *CultArea* of 80ha, which is far larger than the others (maximum 12ha, see Figure 2). From the register data, we removed records that have *CultArea* larger than *FarmArea* – the total area owned by the farmer's household. Since the survey data after cleaning has *CultArea* ranging from 0ha to 12ha, and there are only 107 records (out of 53597) in the register data that have *CultArea* larger than 12ha, we decided to remove the 107 records from the register data. This makes sure that the generalized linear model trained from survey data is applied to the same range of independent variables in the register data.

Another concern with the survey data is whether it is an actual random draw from the population data. To test that, we perform a two-sample Kolmogorov-Smirnov test on the rounded variables of *CultArea* using the function 'ks.test' in R [29]. The result gives a p-value of 0.1629, which suggests that the probability distributions of rounded *CultArea* in the survey data and in the register data are consistent – the reason for this p-value being not significant might be the response heaping problem that we mentioned before. Fig. 3 shows that

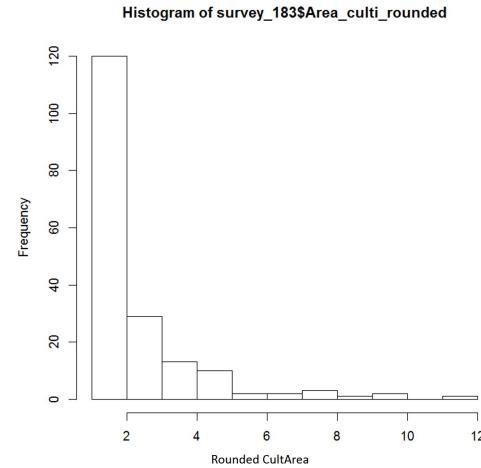


Fig. 2. Histogram of the rounded values of 'Area_culti' in the survey data.

they have similar Empirical Cumulative Distribution Functions (ECDFs).

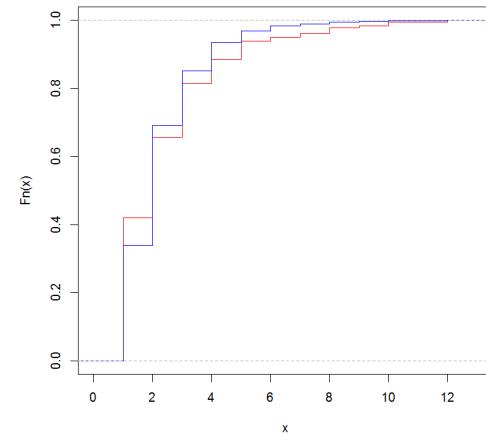


Fig. 3. Comparison of the ECDFs (red: the rounded variable *CultArea* from survey data; blue: the rounded variable of *CultArea* from the register data).

C. Remote Sensing Data

The Monze land cover dataset was generated from the spectral classification of medium-resolution Landsat 5 Thematic Mapper (TM) data. Multi-temporal data were utilized in the analysis, with scenes acquired on September 29, 2008 and May 24, 2009 (nominal scene center of Path: 072 and Row: 071). Land cover within the study area was classified into five primary categories: forest, cropland, savanna, settlement/urban, and water with an emphasis on the accurate delineation of cropland.

An ISODATA algorithm was initially applied to the multi-temporal data to segment pixels into natural clusters reflecting the underlying structure of the data. Clusters were randomly sampled and sample locations were coded with the appropriate land cover label using high resolution imagery. Spectral signatures were extracted at sample locations and statistically clustered into primary category subgroups, ranging from 3-10

subgroups for non-water categories. Subgroup signatures were then used to parameterize a maximum likelihood classifier.

An accuracy assessment of the subgroup thematic map indicated which subgroups, or strata, of primary categories exhibited high commission error. Logit models paired with multi-spectral derivatives were adopted to correct class confusion within high-error strata. Pixels within targeted strata were reclassified based on the predicted probabilities of membership to a particular primary category. In total, seven strata directly affected the accuracy of cropland and were reclassified. Overall accuracy for the five primary categories was 88.18%. Reclassification of select strata reduced the spatial extent of the initial cropland class by over 53% and reduced error of omission to 12.1% and error of commission to 9.8%.

D. Household Characteristics Simulation

The household size (variable *HHSIZE* in the survey data) is the number of members in a household. This integer data can be modeled as a Poisson distribution as determined by a goodness-of-fit test (function *goodfit* in R) on the values of *HHSIZE* in the survey data (p-value 0.056). See Fig. 4 for the histogram and a rootogram of the observed and fitted values.

We then used a generalized linear model (specifically *glm* method in R) to fit using the variables *CultArea* (rounded) and *HHSIZE* from the survey data.

$$\text{Log}(E(HHSIZE|CultArea)) = 1.70118 + 0.06279 * \text{CultArea} \quad (2)$$

The fitted model can be visualized with the survey data (Fig. 5).

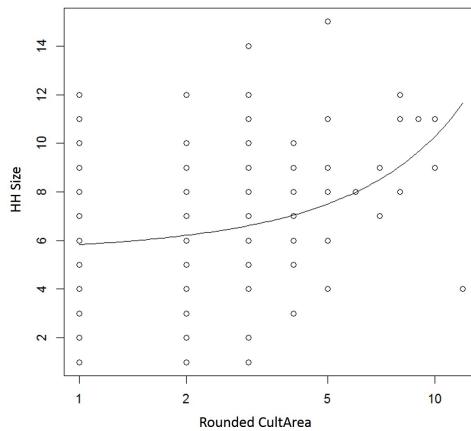


Fig. 5. Visualization of the fitted model using the survey data. Note that the X-axis is in log scale.

Using R's *summary* method on the fitted model we can see the results of Likelihood Ratio Test (LRT). LRT compares the fitted full model with the restricted model where the independent variable *CultArea* is omitted. The p-value of 1.98e-06 suggests that the dependent variable *HHSIZE* is strongly related to the independent variable *CultArea*. Note that while there might be other predictors of *HHSIZE* and

the generalized model can have more than one predictor, *CultArea* is the only predictor we have in our input data, and our goal is to simulate the missing variables as best as we can.

To simulate the values of *HHSIZE* in the register data, we first used the fitted model to predict the value of *HHSIZE* for each value of *CultArea* in the register data. Then we used the predicted values as the parameter *lambda* (mean value) in Poisson distribution to randomly generate the simulated values of *CultArea* (here we use R's *rpois* method). The simulated data can be plotted together with the survey data (Fig. 6). It can be seen that the simulated data points (blue) appear like "expanding" from the original survey data (red).

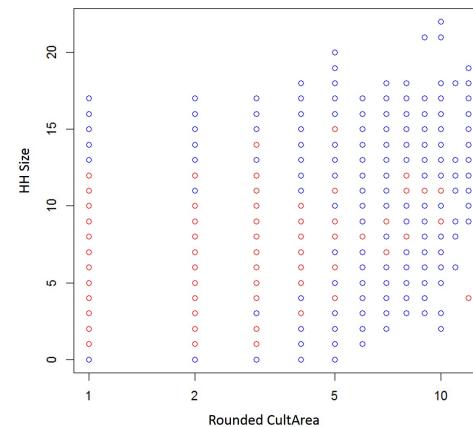


Fig. 6. Overlaid visualization of the simulated data (blue) and the survey data (red). Note that the X-axis is in log scale.

To verify that the simulated data has the same marginal distribution as the survey data, we run the two-sample Kolmogorov-Smirnov test. The resulting p-value of 0.999 shows that the two do have the same probability distribution. This can also be seen from the overlaid Empirical Cumulative Distribution Function (ECDF) in Fig. 7.

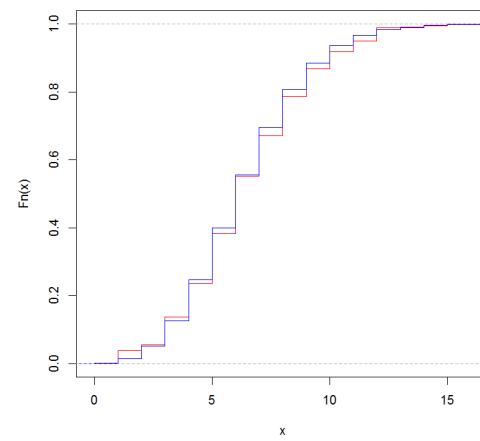


Fig. 7. Comparison of the ECDFs of the variable HHSIZE (red: survey data; blue: simulated data).

However, it is not expected for the simulated data to have the

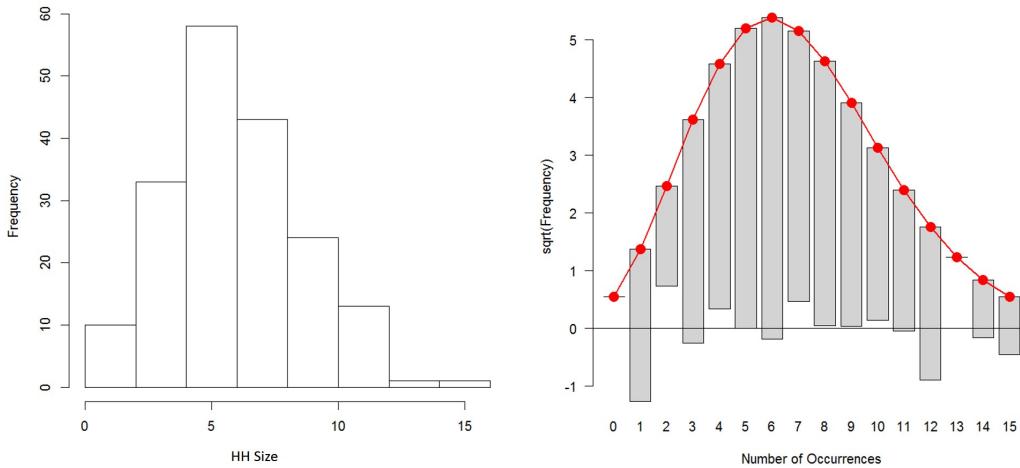


Fig. 4. On the left is a histogram of variable *HHSize* in the survey data and on right is rootogram of the observed and fitted values.

same multivariate distribution as the survey data, as the register data has a different distribution of values of *CultArea*. This can be shown by overlaid visualization of the kernel density estimates (Fig. 8). We also ran a kernel density comparison test on those two distributions to confirm that they are different (the R function *kde.test* generates a p-value of 0).

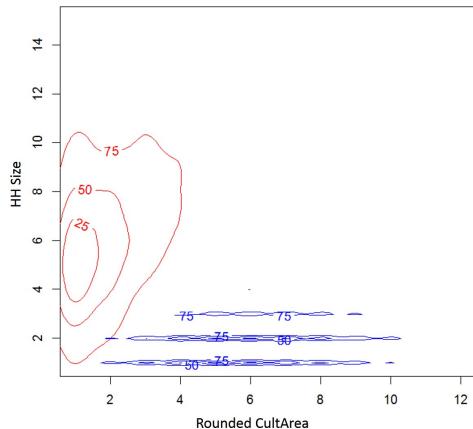


Fig. 8. Overlaid visualization of the kernel density estimates for the simulated data (blue) and the survey data (red).

E. Household Spatial Location Simulation

Next, we add the exact spatial locations to the simulated population of households by allocating to them the agricultural cells/patches generated from remote sensing data. The register data has the aggregated spatial information that tells the ward name of each household. We break down the entire land cover raster of Monze District based on wards, and then run the land allocation algorithm for each ward. Fig. 9 shows the results for one ward.

F. Remaining Missing Variables Calibrated by Microbial GA

Finally, we calibrate all the missing variables whose values could not be determined in previous steps, using the Microbial

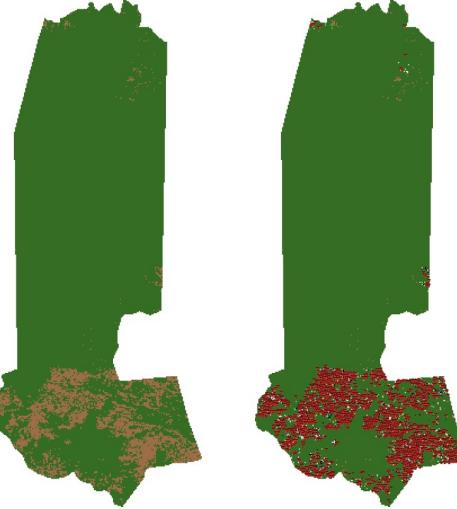


Fig. 9. Results of land allocation in one ward of Monze District, Zambia. Left: agricultural land (brown) and non-agricultural land (green); Right: agricultural land is allocated to households (red).

Genetic Algorithm. Each chromosome is composed of four properties: *soilType* (integer, 0–14), *ratioOfLocalMaize* (double, 0–1), *plantingDateStandardDeviation* (double, 0.001–0.167), and *randomSeed* (integer). Among those properties, *soilType* and *ratioOfLocalMaize* are direct representations of agent variables — there are 15 types of soil in the model, and there are two types of maize (hybrid or local). The possible planting dates within a growing season are every other weeks from middle October to end of January — eight options in total. Finally, we assume that the discrete probability of planting dates follows a normal distribution, that is, we assume most data points fall into 0–1 on the x-axis which is equally divided into eight intervals (eight options by the temporal order), and use the area of distribution on each interval as the probability of planting dates falling into that option. We fix the mean value to be the middle of the

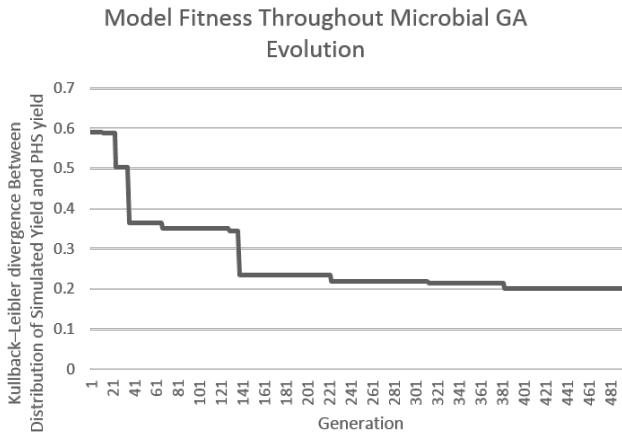


Fig. 10. Model fitness throughout Microbial GA evolution with a population size of 50 and a deme size of 25.

planting date options, and expose its standard deviation as *plantingDateStandardDeviation*. To ensure that most data points in the normal distribution fall between 0–1, we limit the maximum standard deviation to 0.167 (the approximation of $0.5/3$). Thus *plantingDateStandardDeviation* ranges from 0.001 (which has to be larger than zero) to 0.167 (both are inclusive).

We randomly created a population of 50 chromosomes, and evaluated the fitness of each chromosome by running the simulation and measure the Kullback–Leibler divergence between the distribution of simulated yield and that of the observed yield in PHS, for growing season 2011–2012. We calculated the fitness score for each chromosome and stored it into a cache table and updated the score only when that chromosome is mutated/altered, so that each time when a comparison was made between two chromosomes, we can look up the cache table to avoid recalculating the fitness scores.

The deme size in the microbial GA is used to maintain a trivial geography. Spector and Klein [30] chose the deme size arbitrarily and claim that “trivial geography will often provide benefits with a range of deme size and that the choice of deme size is not critical.” In our experiment, we set the deme size to be 50% of the population size, which is 25, and run the Microbial GA for 500 generations. Figure 10 shows that the fitness score (Kullback–Leibler divergence) decreases as the population evolves in the Microbial GA.

Figure 11 shows that the simulated yield has a distribution similar to that of the observed yield from PHS.

Figure 12 shows the distribution of fitness scores, and it can be seen that the entire population tends to converge to the optimal (minimal) fitness score. What is more interesting is to see if there are different sets of values that can all produce close to optimal fitness scores. Table I shows the details of the chromosomes that fall into the first bin (0.2–0.4). It can be seen that:

- 1) There are several different *soilType* that can produce the optimal results;

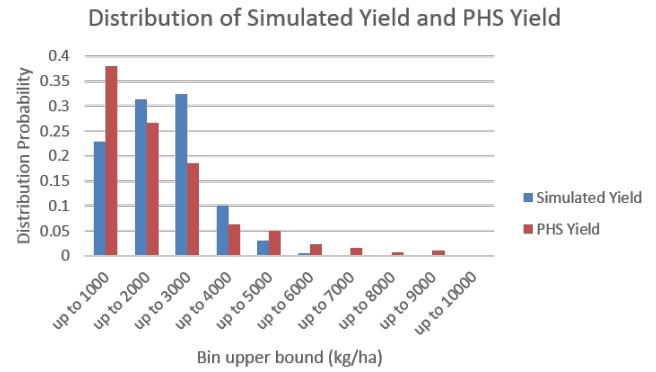


Fig. 11. The comparison between simulated yield distribution and the observed yield distribution from PHS.

Histogram of Fitness Scores after 500 Generations in Microbial GA

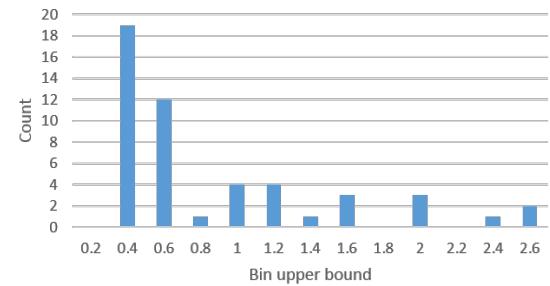


Fig. 12. The distribution of fitness scores (KullbackLeibler divergence) after 500 generations in Microbial Genetic Algorithm.

- 2) *ratioOfLocalMaize* tends to converge to 0.57;
- 3) *plantingDateStandardDeviation* converged to 0.167;
- 4) The random number seed can be very different.

To summarize, we applied the Microbial GA to calibrate the missing variables that cannot be derived in previous steps, and the result shows good matching in distributions of simulated data and observed data.

G. Discussion

An agent in ABM is generally characterized by lots of parameters which together determine the global dynamics of the model. Thus when calibrating these parameters, the search space is huge and Genetic Algorithms (GAs) are good at dealing with the large dimensionality of this problem. There is no mathematical equation that can anticipate the dynamics of an agent-based model without executing it, and thus the computation of the fitness function requires the execution of simulation, which implies a high computational cost. Although more computationally intensive than traditional nonlinear estimation techniques, the GA is capable of accurately finding optimum parameter sets and providing additional information about the search space (Table I). While Monte Carlo exper-

TABLE I
THE SETS OF PARAMETERS THAT CAN PRODUCE THE BEST FITNESS SCORES.

score	soilType	ratioOfLocalMaize	plantingDateStandardDeviation	randomSeed
0.20116529	WI_LVLS007	0.572238345	0.167	-6.53E+08
0.214395283	WI_LVLS007	0.504317117	0.167	-1.45E+09
0.218892992	WI_GLBW752	0.598754994	0.167	1.73E+09
0.218977268	WI_GLBW752	0.598754994	0.167	2.04E+09
0.224874894	WI_GLBW752	0.687669638	0.167	7.33E+08
0.227640214	WI_GLBW752	0.557108806	0.167	9.41E+08
0.236832655	WI_GLBW752	0.598754994	0.161571945	9.29E+08
0.247164101	WI_LVLS007	0.879742596	0.167	-4.08E+08
0.272959905	WI_LVLS007	0.926399603	0.167	-1.41E+09
0.293541824	WI_GLBW752	0.879742596	0.167	-4.61E+08
0.299738564	WI_GLBW752	0.879742596	0.167	-7.63E+08
0.317545137	WI_LVLS007	1	0.167	1.62E+09
0.323698096	WI_LVLS007	1	0.167	4.99E+07
0.326112368	WI_LVLS007	1	0.167	-6.53E+08
0.326112368	WI_LVLS007	1	0.167	-6.53E+08
0.329442109	WI_LVLS007	1	0.167	1.92E+09
0.344025581	WI_PHCF014	1	0.167	-5.85E+08
0.351703556	WI_CMZR003	1	0.167	8.21E+08
0.358091846	WI_PHCF014	1	0.167	1.62E+09

iments could be conducted to generate similar information about the search space, GA is much more efficient as it is integrated with the optimization process. In addition to the computational cost, two other difficulties of using GA in ABM are the choice of fitness function and the stochasticity of simulation run. In Section V-F, we demonstrate that our proposed method can successfully address these two problems.

VI. CONCLUSION AND FUTURE WORK

In this paper we propose a hybrid method that can create a synthetic population to reflect the structure and heterogeneities of real farmer households, and to optimize the *replicative validity* of the model matching data already acquired from the real system (retrodiction). While existing research generally does not use real population data directly in creating synthetic population, due to its lack of information, we demonstrate its applicability in agent-based modeling by integrating other data sources with the real population data. For agent variables whose values cannot be determined due to lack of data, we propose to use Genetic Algorithm (e.g., the Microbial GA) to search for a set of values that can match the model to production survey data (e.g., PHS). We expose the random number seed of the model and the platform as a property of the chromosome to be determine by GA, and we evaluate the fitness based on the distribution of simulated yield using the Kullback–Leibler divergence. We have applied the method to our food security agent-based model in Zambia. The result shows that the synthetic population generated using our method can reflect the marginal distributions of aggregated survey data, and the distribution of simulated yield is close to that of the observed yield.

The next step is to use the synthetic population as the basis and continue developing the Zambia agent-based model to study the interactions between household agents (e.g., labor sharing) and the impact of climate change on food security. We will evaluate whether or not the generated synthetic population

can achieve good *predictive validity* – the model matches data before data is acquired from the real system. Finally, there are other parameter search methods such as Reinforcement Learning, and we will compare them with the GA method.

ACKNOWLEDGMENT

The authors thank Sean Sweeney at Indiana University for generating the landcover dataset from the remote sensing data. The research is supported in part by the National Science Foundation under grants BCS1026776 and SES-1360463, and by the Pervasive Technology Institute at Indiana University.

REFERENCES

- [1] T. Berger, “Agent-based spatial models applied to agriculture: a simulation tool for technology diffusion, resource use changes and policy analysis,” *Agricultural economics*, vol. 25, no. 2, pp. 245–260, 2001.
- [2] R. Moeckel, K. Spiekermann, and M. Wegener, “Creating a synthetic population,” in *Proceedings of the 8th International Conference on Computers in Urban Planning and Urban Management (CUPUM)*, 2003.
- [3] T. P. Evans and H. Kelley, “Multi-scale analysis of a household level agent-based model of landcover change,” *Journal of Environmental Management*, vol. 72, no. 1, pp. 57–72, 2004.
- [4] H. Kelley and T. Evans, “The relative influences of land-owner and landscape heterogeneity in an agent-based model of land-use,” *Ecological Economics*, vol. 70, no. 6, pp. 1075–1087, 2011.
- [5] R. J. Beckman, K. A. Baggerly, and M. D. McKay, “Creating synthetic baseline populations,” *Transportation Research Part A: Policy and Practice*, vol. 30, no. 6, pp. 415–429, 1996.
- [6] D. Felsenstein, A. Y. Grinberger, and M. Lichter, “Dynamic agent based simulation of an urban disaster using synthetic big data,” in *Workshop on Big Data and Urban Informatics*, 2014.
- [7] K. G. Troitzsch, “Validating simulation models,” in *18th European Simulation Multiconference. Networked Simulations and Simulation Networks*, 2004, pp. 265–270.
- [8] O. B. Espinosa, “A genetic algorithm for the calibration of a micro-simulation model,” *arXiv:1201.3456*, 2012.
- [9] B. Calvez and G. Hutzler, “Automatic tuning of agent-based models using genetic algorithms,” in *Multi-agent-based simulation VI*. Springer, 2005, pp. 41–57.
- [10] Z. Y. Wu, T. Walski, R. Mankowski, G. Herrin, R. Gurrieri, and M. Tryby, “Calibrating water distribution model via genetic algorithms,” *Proc. AWWA IMTech, Kansas City, Mo*, 2002.

- [11] A. E. Mulligan and L. C. Brown, "Genetic algorithms for calibrating water quality models," *Journal of environmental engineering*, vol. 124, no. 3, pp. 202–211, 1998.
- [12] I. Harvey, "The microbial genetic algorithm," in *Advances in artificial life. Darwin Meets von Neumann*. Springer, 2009, pp. 126–133.
- [13] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.
- [14] T. Iwamura, E. F. Lambin, K. M. Silvius, J. B. Luzar, and J. M. Fragoso, "Agent-based modeling of hunting and subsistence agriculture on indigenous lands: Understanding interactions between social and ecological systems," *Environmental Modelling & Software*, vol. 58, pp. 109–127, 2014.
- [15] D. Valbuena, P. H. Verburg, and A. K. Bregt, "A method to define a typology for agent-based analysis in regional land-use research," *Agriculture, Ecosystems & Environment*, vol. 128, no. 1, pp. 27–36, 2008.
- [16] C. G. Ralha, C. G. Abreu, C. G. Coelho, A. Zaghetto, B. Macchiavello, and R. B. Machado, "A multi-agent model system for land-use change simulation," *Environmental Modelling & Software*, vol. 42, pp. 30–46, 2013.
- [17] A. Smajgl, D. G. Brown, D. Valbuena, and M. G. Huigen, "Empirical characterisation of agent behaviours in socio-ecological systems," *Environmental Modelling & Software*, vol. 26, no. 7, pp. 837–844, 2011.
- [18] A. Smajgl and E. Bohensky, "Behaviour and space in agent-based modelling: Poverty patterns in east kalimantan, indonesia," *Environmental Modelling & Software*, vol. 45, pp. 8–14, 2013.
- [19] V. Gaube and A. Remesch, "Impact of urban planning on household's residential decisions: An agent-based simulation model for vienna," *Environmental Modelling & Software*, vol. 45, pp. 92–103, 2013.
- [20] M. Schouten, T. Verwaart, and W. Heijman, "Comparing two sensitivity analysis approaches for two scenarios with a spatially explicit rural agent-based model," *Environmental Modelling & Software*, vol. 54, pp. 196–210, 2014.
- [21] D. Murray-Rust, C. Brown, J. Van Vliet, S. Alam, D. Robinson, P. Verburg, and M. Rounsevell, "Combining agent functional types, capitals and services to model land use dynamics," *Environmental Modelling & Software*, vol. 59, pp. 187–201, 2014.
- [22] T. D. Pigott, "A review of methods for missing data," *Educational research and evaluation*, vol. 7, no. 4, pp. 353–383, 2001.
- [23] A. Alfons, S. Kraft, M. Templ, and P. Filzmoser, "Simulation of synthetic population data for household surveys with application to eu-silc," Research Report CS-2010-1, Department of Statistics and Probability Theory, Vienna University of Technology. URL <http://www.statistik.tuwien.ac.at/forschung/CS/CS-2010-1complete.pdf>, Tech. Rep., 2010.
- [24] T. Arentze, H. Timmermans, and F. Hofman, "Creating synthetic populations: approach and empirical results," in *Proceedings of the AET European Transport Conference*, 2001.
- [25] ———, "Creating synthetic household populations: problems and approach," *Transportation Research Record: Journal of the Transportation Research Board*, 2015.
- [26] T. J. Frazier and A. Alfons, "Generating a close-to-reality synthetic population of ghana," *Social Science Research Network (SSRN)* 2086345, 2012.
- [27] P. McCullagh, "Generalized linear models," *European Journal of Operational Research*, vol. 16, no. 3, pp. 285–292, 1984.
- [28] A. L. Holbrook, S. Anand, T. P. Johnson, Y. I. Cho, S. Shavitt, N. Chávez, and S. Weiner, "Response heaping in interviewer-administered surveys is it really a form of satisficing?" *Public Opinion Quarterly*, vol. 78, no. 3, pp. 591–633, 2014.
- [29] R. Ihaka and R. Gentleman, "R: a language for data analysis and graphics," *Journal of computational and graphical statistics*, vol. 5, no. 3, pp. 299–314, 1996.
- [30] L. Spector and J. Klein, "Trivial geography in genetic programming," in *Genetic programming theory and practice III*. Springer, 2006, pp. 109–123.

A parallel microsimulation package for modelling cancer screening policies

Andreas Karlsson*, Niten Olofsson^{†‡}, Erwin Laure[§] and Mark Clements*

* Department of Medical Epidemiology and Biostatistics, Karolinska Institutet, Stockholm, Sweden

& Swedish e-Science Research Centre (SeRC)

Email: andreas.a.karlsson@ki.se, mark.clements@ki.se

[†] Department of Neuroscience, Karolinska Institutet, Stockholm, Sweden

[‡] Combitech AB, Stockholm, Sweden

Email: niten.olofsson@combitech.se

[§] Center for High Performance Computing, Royal Institute of Technology, Stockholm, Sweden

& Swedish e-Science Research Centre (SeRC)

Email: erwinl@pdc.kth.se

Abstract—Microsimulation with stochastic life histories is an important tool in the development of public policies. In this article, we use microsimulation to evaluate policies for prostate cancer testing.

We implemented the microsimulations as an R package, with pre- and post-processing in R and with the simulations written in C++. Calibrating a microsimulation model with a large population can be computationally expensive. To address this issue, we investigated four forms of parallelism: (i) shared memory parallelism using R; (ii) shared memory parallelism using OpenMP at the C++ level; (iii) distributed memory parallelism using R; and (iv) a hybrid shared/distributed memory parallelism using OpenMP at the C++ level and MPI at the R level.

The close coupling between R and C++ offered advantages for ease of software dissemination and the use of high-level R parallelisation methods. However, this combination brought challenges when trying to use shared memory parallelism at the C++ level: the performance gained by hybrid OpenMP/MPI came at the cost of significant re-factoring of the existing code.

As a case study, we implemented a prostate cancer model in the microsimulation package. We used this model to investigate whether prostate cancer testing with specific re-testing protocols would reduce harms and maintain any mortality benefit from prostate-specific antigen testing. We showed that four-yearly testing would have a comparable effectiveness and a marked decrease in costs compared with two-yearly testing and current testing.

In summary, we developed a microsimulation package in R and assessed the cost-effectiveness of prostate cancer testing. We were able to scale up the microsimulations using a combination of R and C++, however care was required when using shared memory parallelism at the C++ level.

I. INTRODUCTION

Monte Carlo simulation is an important tool in the development of public policies. If the system is too complex, then it is often easier to simulate the system rather than to solve it analytically. *Microsimulation* is a simulation approach that operates at the level of individual units. In econometrics and the health sciences, those units are usually individual people and the microsimulation generates stochastic life histories. Agent-based models also focus on individuals, but they tend

to focus on emergent behaviours and interactions, while microsimulations tend to focus on independent individuals [1].

Microsimulation models can be classified as static models, where behaviours are constant over time, or dynamic models, which allow for changing behaviours over time [2]. Microsimulation models can be further classified by whether they model for continuous or discrete time, and whether the model formulation is either event-oriented, where events are modelled separately, or process-oriented, where a sequence of events is modelled within a process.

As a motivating example, we are using microsimulation to develop policies related to prostate cancer screening¹. Prostate-specific antigen (PSA) testing to screen for prostate cancer is common in many Western countries, however the potential harms from PSA testing may outweigh any benefits from reduced prostate cancer mortality. In Figure 1, we describe the use of a microsimulation model to predict progression of prostate cancer for the purpose of planning policies for prostate cancer screening. The microsimulation predicts life-lines under different screening scenarios for large populations (e.g. 10^7 to 10^8 individuals) and the population statistics are then used to inform public policy decision makers. Figure 1 illustrates life histories where prostate cancer is detected early through screening and late through symptoms.

From a recent diagnostic trial (STHLM3), we showed that a new prostate cancer test (denoted the STHLM3 Model, or S3M) could reduce the number of less advanced cancers and appreciably reduce the number of false positive tests while maintaining sensitivity for more advanced prostate cancers [3]. The questions for decision makers are now: How should one plan for cancer screening? Should prostate cancer testing be organised? Can harms be reduced by testing based on risk-stratification where re-testing frequency or referral to biopsy is based on the predicted risk? At what price is the S3M test cost-effective? And how should the S3M be used? By utilising

¹For this article, we define cancer “screening” in the general sense of a screening test in a population which is asymptomatic for cancer, rather than specifically organised screening.

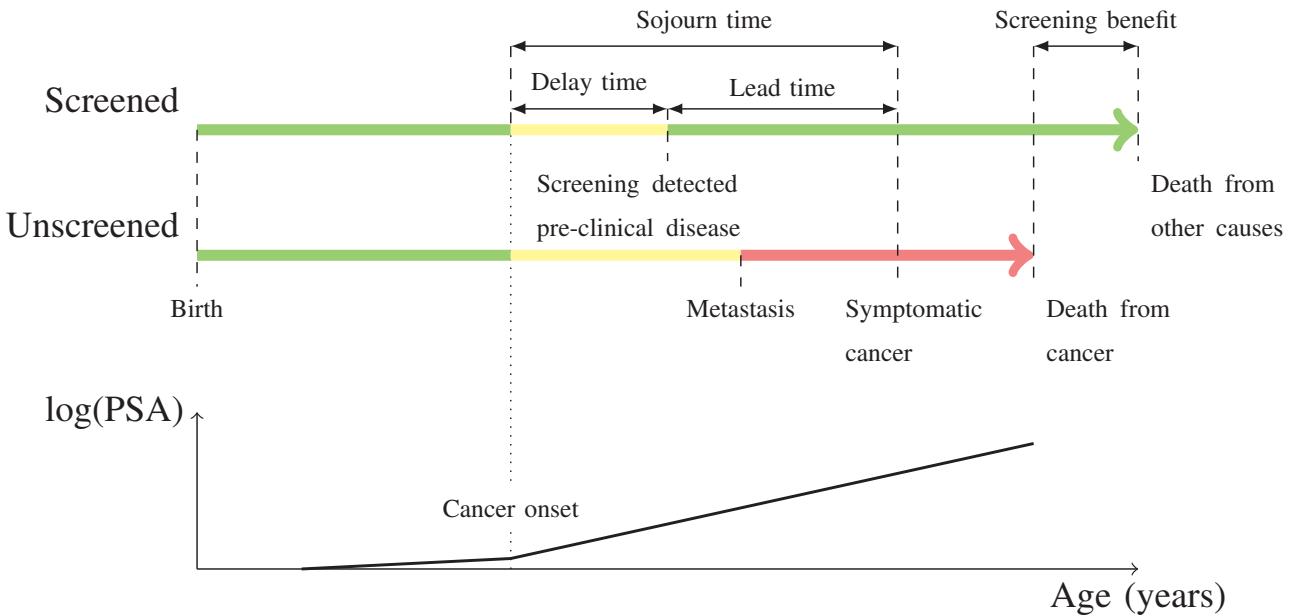


Fig. 1. Simplified schema for the progression of prostate cancer and its relationship with prostate-specific antigen (PSA) as an observable biomarker. We display two example life-lines, where the first life-line experiences early detection screening and cure whereas the second life-line does not. The time intervals shown in our two example life-lines are: *Sojourn time*, which is the duration of disease detectable by screening before clinical symptoms; *Lead time*, which is the time between detection by screening and when the cancer would otherwise have become clinically detected; and *Delay time*, which is the time between when the cancer is first screen-detectable and when the cancer is screen detected.

health registries and clinical trials (Figure 2), microsimulations can provide answers to these questions to support the decision making process. Once the microsimulation model is calibrated, predictions comparing a range of public policies can be performed.

utilities and costs for screening intervention k , such that:

$$\text{Effectiveness}_k = \frac{1}{n} \sum_{i=1}^n \int_0^\infty \frac{dU_{ik}(t)}{(1+\delta)^t}$$

$$\text{Costs}_k = \frac{1}{n} \sum_{i=1}^n \int_0^\infty \frac{dC_{ik}(t)}{(1+\delta)^t}$$

where we simulate for n individuals with index i , with individual-based cumulative utilities $U_{ik}(t)$ and costs $C_{ik}(t)$ at time t , with discounting δ (e.g. $\delta = 0.03$).

It can be computationally expensive to run and calibrate microsimulations for a large population. As cancer events are rare, there is a need to simulate for a large population of individuals (e.g. 10^7 to 10^8 individuals). The models can be calibrated using methods such as the Nelder-Mead algorithm [4] or approximate Bayesian computations [5], which are slow to converge and typically need 500 to 10,000 iterations. With a microsimulation on 10^8 individuals, one run takes approximately 1.5 hour. Iterated 500 times on a single core, the calibration would take approximately one month. In order for the model to be calibrated under different assumptions, results need to be available within hours.

With the pervasive availability of multi-core architectures, massive computational power is today available not only in large-scale supercomputers, but also in smaller clusters or on the cloud. To exploit this computational power, however, simulations need to be explicitly parallelised. Several methods exist, including shared and distributed memory parallelisation

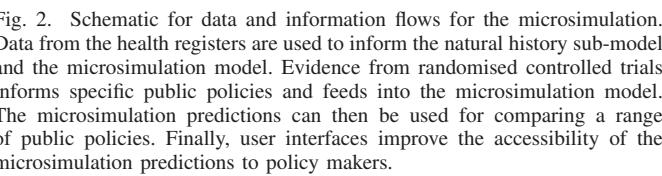


Fig. 2. Schematic for data and information flows for the microsimulation. Data from the health registers are used to inform the natural history sub-model and the microsimulation model. Evidence from randomised controlled trials informs specific public policies and feeds into the microsimulation model. The microsimulation predictions can then be used for comparing a range of public policies. Finally, user interfaces improve the accessibility of the microsimulation predictions to policy makers.

Cancer screening policies may be based on *cost-effectiveness analysis*, where the expected lifetime costs and health utilities for different policies are compared. The cost-effectiveness from the microsimulation can be described using

using OpenMP or MPI [6], [7], but it is not always straightforward to parallelise applications efficiently.

Our objectives are three-fold: (i) to develop a parallel microsimulation package; (ii) to assess whether the package scales well using different forms of parallel computing; and (iii) to apply the microsimulation package to model screening for prostate cancer.

The remainder of the paper is organised as follows. A discussion of related work is given in Section II. In Section III, we describe our microsimulation package and assess its efficiency using different forms of parallelism. Section IV describes a case study, where we model prostate cancer screening to assess the cost-effectiveness of different screening scenarios. Section V concludes with a brief summary and a discussion of future work.

II. RELATED WORK

A recent review of dynamic microsimulation models is provided by Li and O'Donoghue [2]. There are few general tools for dynamic, continuous time microsimulation modelling. One example is the closed source MODGEN microsimulation language developed by Statistics Canada [8]. The Openm++ platform is a recent open source re-implementation of MODGEN [9].

Most discrete-event simulation (DES) frameworks and libraries (e.g. OMNeT++, ns-3, adevs, SimX, and JAMES-II [10]–[14]) focus on large and process-oriented simulations, rather than lightweight, event-oriented libraries as required for our purposes. A discrete time formulation, as used by many dynamic microsimulation models (e.g JAMSIM, SESIM [15], [16]), is less efficient than continuous time for modelling rare events, such as cancer death.

There are few DES libraries available in R [17]. They include: **MicSim**, which implements simple microsimulations that are useful for teaching [18]; **simmer** for a specific set of process-oriented simulations, where the model is specified in R using a C++ core [19]; and **MILC**, which defines a natural history model for lung cancer. R is also a common choice for post-processing of microsimulation traces, with general tools provided by the **Biograph** package and specific tools provided by, for example, the **JAMSIM** and **MicCore** packages [15], [20], [21].

Many of the microsimulation models for cancer screening have been supported by the Cancer Intervention and Surveillance Network (CISNET). CISNET has taken a multiple-model approach, where the different models share data inputs to predict the effects of screening and treatment on health outcomes. This approach has been applied to several cancer sites [22], [23]. CISNET has provided extensive model documentation, but model code has not been widely disseminated.

There is a lack of open source microsimulation models for cancer screening. Moreover, there is a paucity of tools for scaling microsimulations to high performance computing (HPC) or Big Data frameworks such as Hadoop, Spark or Flink.

III. THE MICROSIMULATION PACKAGE

Algorithm 1 describes the microsimulation in terms of an event-oriented discrete-event simulation. Simulation inputs include person attributes (e.g. year of birth), simulation parameters (e.g. testing frequencies by age) and random number seeds. Events are scheduled by insertion into the event queue. As the events occur during the simulation, the person attributes and simulation report may be updated. This is then repeated over J persons where the initialised person attributes are varied to create the appropriate random effects within the simulated cohort.

```

input : Simulation parameters
output: Simulation report
report  $\leftarrow \{\}$ ;
for  $j \leftarrow 1$  to  $J$ ;           // iterate over people
do
    initialise: begin
        time  $\leftarrow 0$ ;
        queue  $\leftarrow$  EventQueue();
        set person attributes;
        schedule natural history events;
        schedule screening events;
    end
    while queue is not empty do
        event  $\leftarrow$  pop(queue);
        time  $\leftarrow$  event.time;
        handle event: begin
            update person attributes;
            schedule new events;
            write to report;
        end
    end
    return report;

```

Algorithm 1: Algorithmic description of a single microsimulation run.

We implemented this algorithm as an R package. R has become the *de facto* programming language for statisticians and it is also a popular language for data scientists. The microsimulation package is open source and released under a GPL licence [24]. For speed, the simulation engine and model were specified in C++, with pre- and post-processing in R, using the Rcpp package to seamlessly pass data structures between R and C++ [25]. R packages are easy to disseminate and can include C++ source code or binaries.

We used some publicly available C++ libraries. The C++ core uses the SSIM library for discrete-event simulation [26]. The SSIM library is lightweight, providing classes for processes and events and static methods for the simulation. The library supports both event-oriented simulations, where processes respond to event messages, and process-oriented simulations using co-routines. We extended the library to support the removal of events using a predicate, and implemented a richer message API based on the OMNeT++ simulator.

We used common random numbers where each stochastic process (e.g. developing or progressing with disease) is assigned a random number stream. Using the `RngStreams` library, a random number stream is divided into sub-streams and each individual assigned a sub-stream [27]. The use of random sub-streams is a recommended variance reduction strategy for microsimulations [28]. In the baseline implementation, the R and C++ code are closely coupled, where the non-uniform random numbers from R use the uniform random numbers from C++.

We also included an in-simulation reporting infrastructure that splits follow-up by age, collecting details on person-time, utilities, costs and the number of events.

The C++ code for the package is currently C++98 compliant and uses the Boost libraries [29]. `Rcpp` is used to pass data structures between R and C++. In addition to these C++ libraries, the baseline R package includes 3500 lines of C++ code and 1000 lines of R code.

A. Parallelisation

Following a map-reduce programming model [30], we split a simulation, mapped the subsets to different processes, and then collated the reports from each process.

We investigated four forms of parallelism: (i) shared memory parallelism using R, where we coded for the simulation report reductions in R with no changes to the C++ code; (ii) shared memory parallelism using OpenMP at the C++ level, where we needed to substantially re-factor the C++ code; (iii) distributed memory parallelism using MPI, where we coded for reductions and set up the MPI environment in R with no changes to the C++ code; and (iv) a hybrid shared/distributed memory parallelism using OpenMP at the C++ level and MPI at the R level.

- (i) A simple implementation using the `mclapply` function in R's `parallel` package. Each thread used its own instance of the `Rcpp` API to run the computationally intensive part in C++ as illustrated in Figure 3.

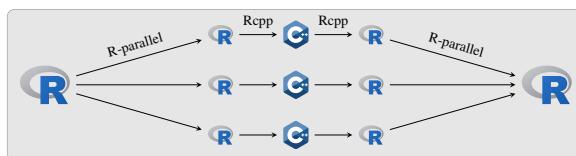


Fig. 3. Implementation utilising the shared memory parallelisation functionality via `mclapply` in R's `parallel` package.

- (ii) We also implemented shared memory parallelisation on the C++ side, using OpenMP (Figure 4).

The re-factoring of the the C++ code for OpenMP included the following. First, R is single threaded, so all random number generation had to be done on the C++ side. In particular, our baseline implementation used R for non-uniform random numbers. Both the `Boost.Random` library and the C++11/C++14

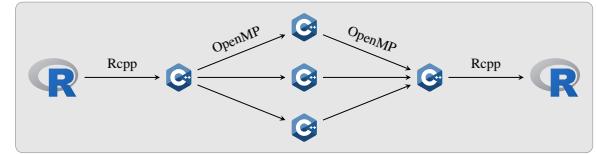


Fig. 4. A second shared memory implementation using OpenMP parallelism on the C++ side.

standard libraries provide non-uniform random numbers; we provided an interface between these libraries and the `RngStreams` library, and rewrote the model to use these non-uniform random number generators. Moreover, we used separate streams per thread and updated the sub-streams for different individuals. The SSIM DES library, as per most DES libraries, uses static methods for simulation, such that there is only one simulator. The OpenMP solution required that the SSIM library be refactored to use simulator instances. Each simulator instance was given an event queue and associated methods to manipulate that queue.

The reduction step of the simulation reporting also required some effort to parallelise. In the serial version of the code, the result was accumulated in an associative array with scope throughout the whole module file. After each map step, the resulting associative array was incrementally updated. In this reduction step several function calls were made updating the report, where the functions accessed it in the scope of the file and not by a passed reference. We first tried to protect the global report object with a `#pragma omp critical` directive.

After a performance analysis, we realised that the reduction step was the bottle-neck of the simulation. The reduction step was re-factored to first perform a thread-wise reduction in a thread-private report object and then a final reduction to a shared report object.

- (iii) The MPI implementation was done using `Rmpi`, which is an R wrapper for MPI. A consequence of using MPI on the R side was that we could use the reduction step we implemented when using R's `parallel` package and a small speed advantage due the structure of the existing simulation code. All threads, also within a node, were started using MPI as illustrated in Figure 5.

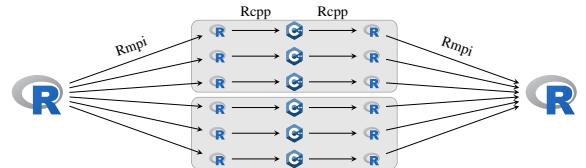


Fig. 5. Implementation using MPI for distributed parallelisation via the `Rmpi` wrapper. All threads, also within a node, were started using MPI.

- (iv) We then combined the OpenMP and the MPI approach to a hybrid solution. The nodes were initiated with MPI and the threads within the node were initiated with OpenMP (Figure 6).

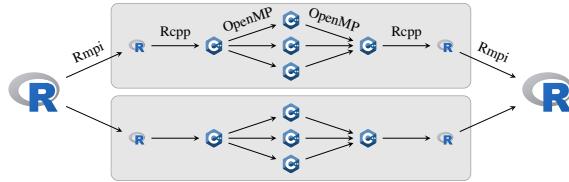


Fig. 6. Hybrid implementation utilising MPI for the outer loop and OpenMP for the inner loop.

This resulted in three reduction steps: (i) a thread private reduction in each thread; (ii) a sequential reduction of those threads within a `#pragma omp critical`; and (iii) another sequential reduction of the MPI nodes on the R-side.

We also investigated the possibility of a hybrid approach using R's parallel package and Rmpi, which would have required considerably less re-factoring. However, we were unable to implement this approach because of package incompatibilities.

B. Performance Results

Our benchmarking of the four implementations was done on a cluster with eight (2x4) 2.2GHz cores per node using a maximum of 16 nodes. The interconnect was a full bisection bandwidth (FBB) Infiniband fabric with a multiple root tree structure. All links were 4xDDR resulting in a per-link bandwidth of 2GB/s. The software versions were OpenMP with gcc version 4.8.1 and the `-O3` optimisation level, R version 3.0.2 and Open MPI version 1.4.1 as the MPI implementation. For the benchmarking, we decided on a simulation size of 10^7 individuals with a moderately complex intervention and detailed reporting. We assessed the performance using the prostate cancer model described later in Section IV.

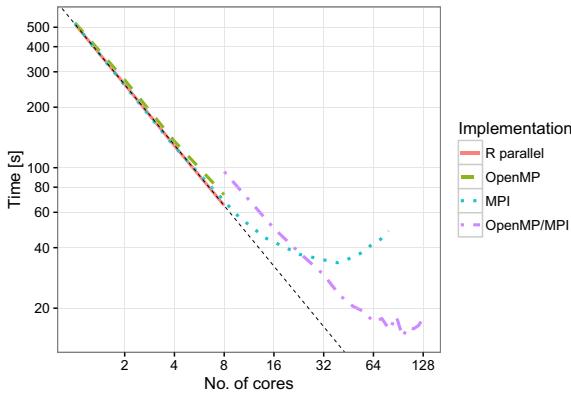


Fig. 7. Execution times on 8 core nodes with implementations using R's parallel package, OpenMP alone, MPI alone and OpenMP/MPI as a hybrid solution. The thin black dashed line represents ideal scaling.

Under these conditions the simulation took more than 500 seconds on a single core (Figure 7). Within a compute node, the R side parallelism scaled well, with a reduction of the total execution time to approximately 70 seconds on eight cores. Across compute nodes, the R side MPI parallelism scaled well when tasks were sufficiently large. As the tasks became smaller, the hybrid OpenMP/MPI solution scaled better. The MPI-only implementation continued to scale up to five nodes after which the simulation time increased. The hybrid OpenMP/MPI solution was slower than using MPI alone for the first three nodes. But, in contrast to using MPI alone, the hybrid continued to scale until approximately 10 nodes, where the execution time was less than 20 seconds. There was an unexpected initial slowdown of the hybrid OpenMP/MPI compared with MPI or OpenMP. This may be partially explained by the hybrid implementation using multiple reduction operations.

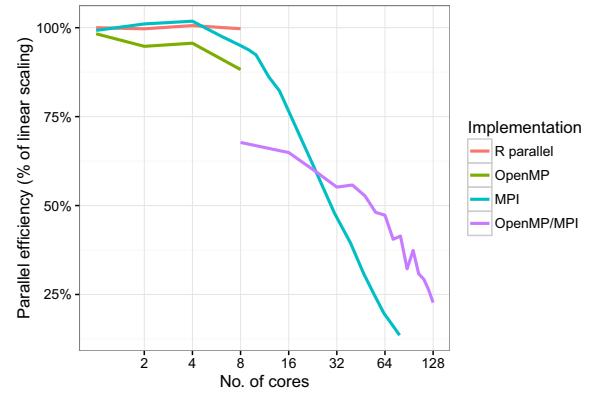


Fig. 8. Efficiency on 8 core nodes with implementations using R's parallel package, OpenMP alone, MPI alone and OpenMP/MPI as a hybrid solution. We observed a slight super-linear speedup for MPI alone on a small numbers of cores.

As can be seen in Figure 8, R's parallel package and MPI alone were the most efficient implementations within a single compute node. We observed a slight super-linear speedup with an efficiency above 100% when using a small numbers of cores. This behaviour may be due to cache effects from increased spatial locality. Alternatively, this behaviour may be due to the smaller tasks using a shorter range of sequential simulation parameters (e.g. birth cohort), which, in turn, reduces the size of the dynamic map in the returned report object. The MPI alone implementation was the most efficient distributed solution for the first three nodes. However, MPI's efficiency decreased rapidly as the number of cores increased and the task sizes became smaller.

Compared with the other implementations, the efficiency for the hybrid OpenMP/MPI solution at one node was comparatively low at approximately 70%. The hybrid solution was more efficient than MPI alone for more than three nodes, and had an efficiency of approximately 25% at 128 cores.

IV. CASE STUDY

Using the microsimulation package, we implemented a prostate cancer model based on a well validated model from the Fred Hutchinson Cancer Research Center [31]; documentation is available on-line [32]. Leveraging extensive Swedish population-based data, we calibrated the model to the Nordic context, incorporating data on 1.4 million PSA tests for 400,000 men living in Stockholm. These data were combined with data on cancer incidence, treatment modalities, mortality and prostate cancer survival [33]. The prostate cancer model is included in the microsimulation package. The simulation runs presented in this case study were comparatively small computational tasks. However, as described in Section I, the preceding model calibrations were larger tasks. As an example, a simple calibration using the Nelder-Mead algorithm took approximately four hours using R-side parallelism on a 24 core node. As we further refined the model by adding more calibration targets, we found that the calibration time increase exponentially, so that using a single node would become infeasible.

For this case study, we compared four screening scenarios: *No screening*, for a hypothetical scenario with no PSA testing and cancer diagnosis only for symptomatic cancers; *2-yearly* and the *4-yearly* scenarios with regular screening at specific intervals between ages 50–69 years; and a *current* scenario, designed to replicate the current PSA testing pattern for men born 1960.

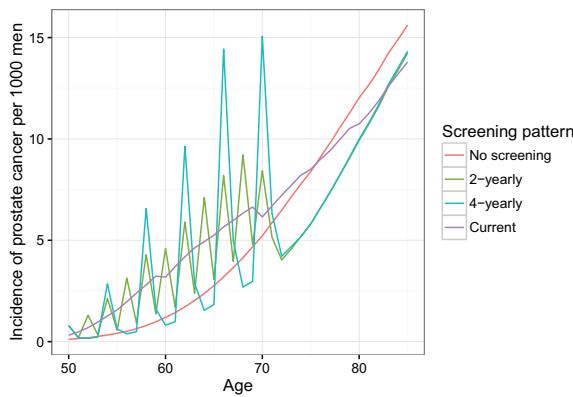


Fig. 9. Annual incidence rates per 1000 men for four screening scenarios. The incidence rates rose sharply with routine 2-yearly and 4-yearly PSA re-testing.

We first described the rate of incident prostate cancers, modelling cancer diagnosis through both symptomatic detection and screening pathways. Compared with *no screening*, the annual cancer incidence rates for routine 2-yearly and 4-yearly testing were higher and spiked during the screening age interval, and then the incidence rates were lower after the screening interval (Figure 9). Incidence rates under *current* screening tended to be higher at older ages than the 2-yearly and 4-yearly testing scenarios, as our model of *current* screening included prostate cancer testing after 70 years of age.

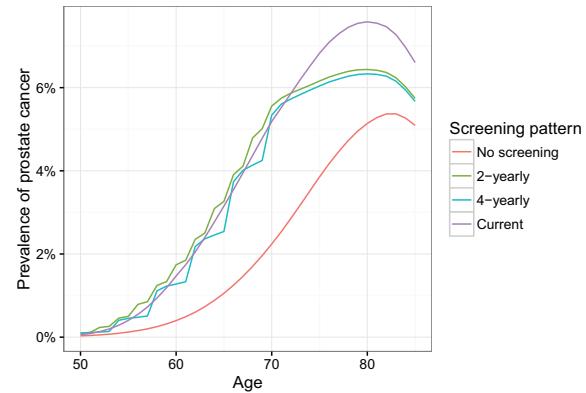


Fig. 10. Prevalence of men diagnosed with prostate cancer for four screening scenarios. Prevalence is defined as men living with a diagnosis of prostate cancer.

We also predicted the prevalence of men living with a prostate cancer diagnosis under these scenarios (Figure 10). Naturally, *no screening* resulted in a lower prevalence, as the asymptomatic men remain undetected, while *2-yearly* and the *4-yearly* testing resulted in an increased prevalence during the screening ages which then dropped well below the prevalence of the *current* testing pattern.

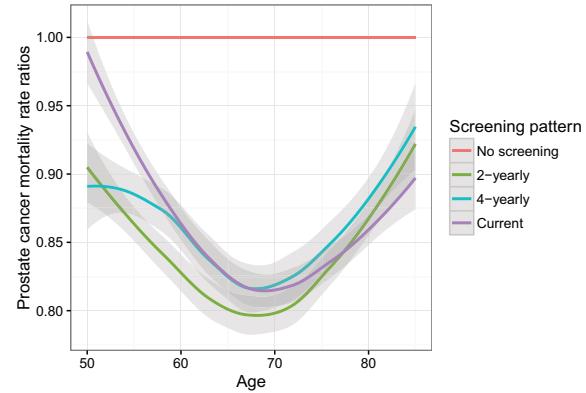


Fig. 11. Prostate cancer mortality rate ratios for the investigated screening patterns with *no screening* as the reference screening pattern.

We then described the predicted effect of the screening scenarios on prostate cancer mortality rates. An ideal screening program would have no false positive tests, no over-diagnosis² and reduce cancer mortality as much as possible. We compared mortality rate ratios of the three screening scenarios relative to *no screening* (Figure 11). Compared with *no screening*, 2-yearly testing had the greatest reduction in prostate cancer mortality through to age 75 years, after which *current* screening had a greater mortality reduction. Notably, 4-yearly screening was assumed to have the greater part of the mortality reduction predicted for 2-yearly testing.

²Over-diagnosis of a cancer can be defined as a screen-detected cancer that would never have been symptomatic prior to death due to another cause.

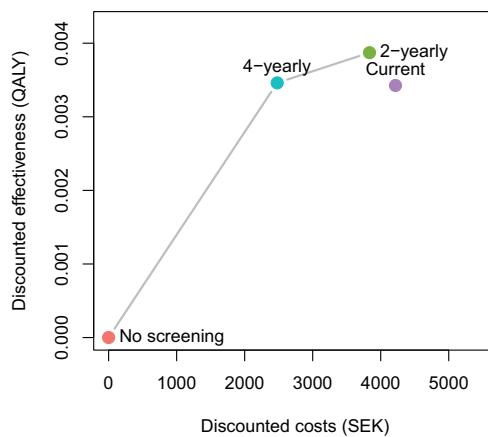


Fig. 12. Cost-effectiveness frontier comparing the expected costs and effectiveness for three prostate cancer screening scenarios relative to no screening. Values were calculated for Sweden taking a societal perspective, with 3% discounting. Costs were reported in Swedish kronor (SEK) and effectiveness was reported using quality adjusted life-years (QALYs).

Finally, we investigated the cost-effectiveness of prostate cancer testing with specific re-testing protocols. For each scenario, we calculated the discounted expected costs and discounted expected effectiveness. We then compared screening scenarios by calculating the incremental cost-effectiveness ratios, defined as the expected cost divided by the change in effectiveness. Effectiveness was measured in terms of quality adjusted life-years (QALYs) and costs were measured in terms of Swedish kronor. Costs and effectiveness were discounted at 3%, and we took a societal perspective. The cost-effectiveness frontier, represented by the grey line in Figure 12, was defined as the most cost-effective testing patterns at different costs. The most frequent testing scenario, with 2-yearly testing, was the most effective and the most costly scenario. The 4-yearly testing pattern offered comparable effectiveness and a marked decrease in costs. Current testing was not on the cost-effectiveness frontier and, compared with 4-yearly testing, was associated with a very similar effectiveness and a marked increase in costs.

V. CONCLUSIONS

In summary, we developed an R package for microsimulation that was closely coupled with C++. The package was used to implement a prostate cancer screening model to assess the cost-effectiveness of prostate cancer testing. In the case study, we showed that four-yearly prostate cancer testing would have a similar effectiveness and a marked decrease in costs compared with two-yearly testing and current testing.

There were a number of advantages provided by closely coupling R and C++: R allowed for easy dynamic scripting, C++ was very fast, and Rcpp provided the glue. Moreover, this combination offered opportunities for using high-level R parallelisation methods, both for shared memory

and distributed systems. However, this combination brought challenges when trying to use shared memory parallelism at the C++ level: the performance gained by the hybrid OpenMP/MPI came at the cost of significant re-factoring of the existing code.

These tools have potential applications to other problem domains, including: screening for other cancer sites; infectious disease transmission; other chronic diseases; and, more generally, social policy. We plan to continue releasing these models as open source, allowing for their use and reuse. However, most existing microsimulation models are not open source, possibly due to the significant time and effort required for developing them. This closed approach to science also limits the dissemination of tools and models. We suggest that a more open scientific approach would strengthen the use of microsimulation in the policy domain.

For the future, we believe that a hybrid shared/distributed memory solution at the R level would be a very useful tool for scientists. We plan to use the prostate cancer model to evaluate the cost-effectiveness of organised testing with novel prostate cancer tests in the Nordic context.

ACKNOWLEDGEMENTS

The authors acknowledge funding support from the Swedish eScience Research Centre, the Nordic Information for Action eScience Center and the Swedish Cancerfonden (CAN 2012/765).

REFERENCES

- [1] M. Boman and E. Holm, *Multi-agent systems, time geography, and microsimulations*. Springer Netherlands, 2004.
- [2] J. Li and C. O'Donoghue, "A survey of dynamic microsimulation models: uses, model structure and methodology," *International Journal of Microsimulation*, vol. 6, no. 2, pp. 3–55, 2013.
- [3] H. Grönberg, J. Adolfsson, M. Aly, T. Nordström, P. Wiklund, Y. Brandberg, J. Thompson, F. Wiklund, J. Lindberg, M. Clements, L. Egevad, and M. Eklund, "Prostate cancer screening in men aged 50–69 years (STHLM3): a prospective population-based diagnostic study," *The Lancet Oncology*, vol. 16, no. 16, pp. 1667 – 1676, 2015. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1470204515003617>
- [4] J. A. Nelder and R. Mead, "A Simplex Method for Function Minimization," *Computer Journal*, vol. 7, pp. 308–313, 1965. [Online]. Available: <http://comjnl.oxfordjournals.org/content/7/4/308>
- [5] S. Tavaré, D. J. Balding, R. C. Griffiths, and P. Donnelly, "Inferring coalescence times from DNA sequence data," *Genetics*, vol. 145, no. 2, pp. 505–518, Feb 1997.
- [6] L. Dagum and R. Menon, "OpenMP: an industry standard API for shared-memory programming," *Computational Science & Engineering, IEEE*, vol. 5, no. 1, pp. 46–55, 1998.
- [7] M. P. Forum, "MPI: A Message-Passing Interface Standard," University of Tennessee, Knoxville, TN, USA, Tech. Rep., 1994.
- [8] Statistics Canada, *MODGEN Version 10.1.0: Developer's Guide*, Statistics Canada, 2012. [Online]. Available: <http://www.statcan.gc.ca/microsimulation/modgen/doc-eng.htm>
- [9] "OpenM++: open source microsimulation platform," 2016, accessed: 2016-05-26. [Online]. Available: <http://ompp.sourceforge.net/>
- [10] A. Varga, "The omnet++ discrete event simulation system," in *In ESM'01*, 2001.
- [11] "NS-3," 2016, accessed: 2016-05-26. [Online]. Available: <http://ns-3.org/>
- [12] J. Nutaro, "adevs (A Discrete Event Simulator) library," 2016, accessed: 2016-05-26. [Online]. Available: <http://web.ornl.gov/~1qn/adevs/>

- [13] S. Thulasidasan, L. Kroc, and S. Eidenbenz, "Developing parallel, discrete event simulations in Python - first results and user experiences with the SimX library," in *4th International Conference On Simulation And Modeling Methodologies, Technologies And Applications, SIMULTECH 2014, Vienna, Austria, August 28-30, 2014*, 2014, pp. 188–194. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=7095018
- [14] J. Himmelsbach and A. M. Uhrmacher, "Plug'n simulate," in *Simulation Symposium, 2007. ANSS '07. 40th Annual*, March 2007, pp. 137–143.
- [15] O. Mannion, R. Lay-Yee, W. Wrapson, P. Davis, and J. Pearson, "JAM-SIM: A Microsimulation Modelling Policy Tool," *Journal of Artificial Societies and Social Simulation*, vol. 15, no. 1, p. 8, 2012.
- [16] L. Flood, F. Jansson, T. Pettersson, T. Pettersson, O. Sundberg, and A. Westerberg, "SESIM III—a Swedish dynamic microsimulation model," *Handbook of SESIM, Ministry of Finance, Stockholm*, 2005.
- [17] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2014. [Online]. Available: <http://www.R-project.org/>
- [18] S. Zinn, "The MicSim Package of R: An Entry-Level Toolkit for Continuous-Time Microsimulation," *International Journal of Microsimulation*, vol. 7, no. 3, pp. 3–32, 2014. [Online]. Available: <https://ideas.repec.org/a/ijm/journl/v7y2014i3p3-32.html>
- [19] B. Smeets and I. Ucar, "simmer: Discrete-Event Simulation for R," 2016, accessed: 2016-05-26. [Online]. Available: <https://cran.r-project.org/web/packages/simmer/index.html>
- [20] F. Willekens, *Multistate Analysis of Life Histories with R*. New York, NY: Springer, 2014.
- [21] S. Zinn, J. Himmelsbach, A. M. Uhrmacher, and J. Gampe, "Building Mic-Core, a specialized M&S software to simulate multi-state demographic micro models, based on JAMES II, a general M&S framework," *Journal of Artificial Societies and Social Simulation*, vol. 16, no. 3, p. 5, 2013.
- [22] C. M. Rutter, A. M. Zaslavsky, and E. J. Feuer, "Dynamic microsimulation models for health outcomes: a review," *Medical Decision Making*, vol. 31, no. 1, pp. 10–18, 2011.
- [23] J. S. Mandelblatt, K. A. Cronin, S. Bailey, D. A. Berry, H. J. De Koning, G. Draisma, H. Huang, S. J. Lee, M. Munsell, S. K. Plevritis *et al.*, "Effects of mammography screening under different screening schedules: model estimates of potential benefits and harms," *Annals of Internal Medicine*, vol. 151, no. 10, pp. 738–747, 2009.
- [24] Microsimulation package for R. Accessed: 2016-05-30. [Online]. Available: <https://github.com/mclements/microsimulation>
- [25] D. Eddelbuettel and R. François, "Rcpp: Seamless R and C++ integration," *Journal of Statistical Software*, vol. 40, no. 8, pp. 1–18, 2011. [Online]. Available: <http://www.jstatsoft.org/v40/i08/>
- [26] SSIM - A Simple Discrete-Event Simulation Library. Accessed: 2016-05-30. [Online]. Available: <http://www.inf.usi.ch/carzaniga/ssim/>
- [27] P. L'Ecuyer, R. Simard, E. J. Chen, and W. D. Kelton, "An object-oriented random-number package with many long streams and substreams," *Operations Research*, vol. 50, no. 6, pp. 1073–1075, 2002. [Online]. Available: <http://dx.doi.org/10.1287/opre.50.6.1073.358>
- [28] N. K. Stout and S. J. Goldie, "Keeping the noise down: common random numbers for disease simulation modeling," *Health Care Management Science*, vol. 11, no. 4, pp. 399–406, Dec. 2008.
- [29] BOOST C++ Libraries. Accessed: 2016-05-30. [Online]. Available: <http://www.boost.org>
- [30] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in *Sixth Symposium on Operating System Design and Implementation*, San Francisco, December 2004.
- [31] R. Gulati, J. L. Gore, and R. Etzioni, "Comparative effectiveness of alternative prostate-specific antigen-based prostate cancer screening strategies: model estimates of potential benefits and harms," *Annals of Internal Medicine*, vol. 158, no. 3, pp. 145–153, Feb. 2013.
- [32] CISNET: Prostate Cancer Model Profiles. Accessed: 2016-05-30. [Online]. Available: <http://cisnet.cancer.gov/prostate/profiles.html>
- [33] J. R. Rider, F. Sandin, O. Andren, P. Wiklund, J. Hugosson, and P. Stattin, "Long-term outcomes among noncuratively treated men according to prostate cancer risk category in a nationwide, population-based study," *Eur. Urol.*, vol. 63, no. 1, pp. 88–96, Jan 2013.

OntoSoft: A Distributed Semantic Registry for Scientific Software

Yolanda Gil, Daniel Garijo, Saurabh Mishra, Varun Ratnakar

Information Sciences Institute
University of Southern California
Los Angeles, CA, USA

gil@isi.edu, dgarijo@isi.edu, saurabhm@usc.edu, varunr@isi.edu

Abstract— OntoSoft is a distributed semantic registry for scientific software. This paper describes three major novel contributions of OntoSoft: 1) a software metadata registry designed for scientists, 2) a distributed approach to software registries that targets communities of interest, and 3) metadata crowdsourcing through access control. Software metadata is organized using the OntoSoft ontology along six dimensions that matter to scientists: identify software, understand and assess software, execute software, get support for the software, do research with the software, and update the software. OntoSoft is a distributed registry where each site is owned and maintained by a community of interest, with a distributed semantic query capability that allows users to search across all sites. The registry has metadata crowdsourcing capabilities, supported through access control so that software authors can allow others to expand on specific metadata properties.

Keywords—software registries, software metadata, scientific software, software catalogs, software repositories

I. INTRODUCTION

The software developed by scientists embodies important scientific knowledge that should be explicitly captured, curated, managed, and disseminated. Software captures mathematical models, statistical analyses, and causal reasoning that are used to generate new results. Scientists recognize the value of sharing software to avoid replicating effort and to inspect and reproduce results from others. In addition, recurring issues of provenance and uncertainty in the context of data could be better addressed with improved treatment of software: one of the best ways to understand data is to look at the software that uses it or generates it.

A major issue for scientific software reuse is the dissemination and documentation of existing codes. Although code repositories already exist and are used by many scientists, they typically contain basic metadata such as authors, license but lack appropriate metadata to facilitate discovery and reuse.

A second major issue in scientific software sharing is the limited sharing of codes used in scientific publications. While the loss of “dark data” in science is well recognized [1], we see an analogous problem in the pervasive loss of “dark software”. Many scientists do not share their software, because they are unaware of its value, or they do not know how, or they are worried about not getting proper acknowledgment, or they do not see its value. Studies show that scientists spend between 60% to 80% of a project’s effort collecting and preparing data

This work was supported by the U.S. National Science Foundation under award ICER-1440323.

before doing new science (e.g., [2]). This would indicate a significant overhead in developing software for data preparation that is only rarely shared and rarely reused. A common concern is the relatively lower quality of such software, since it is typically not written with robustness or generality in mind and scientists do not want their reputations tarnished [3]. Scientists should be aware of the value of their software, and have the means to share it easily and worry-free.

A third major issue for scientific software sharing is adoption and trust. Scientists should be able to recognize whether they can trust software developed by others since they rely on it to do science. Scientists today have no access to the kinds of community ratings that help assess quality and reuse.

This paper reports on OntoSoft, a distributed semantic software registry aimed to improve scientific software stewardship through: 1) an ontology for software metadata, which organizes information about software in metadata categories that are designed with scientists in mind; 2) metadata crowdsourcing capabilities, where software authors can control permissions for others to edit specific metadata entries; and 3) a distributed architecture where each site can be overseen by a community of practice but metadata is exported and can be queried across all sites.

The paper begins discussing relevant work on characterizing software and how software is described in code repositories today. It introduces the requirements for OntoSoft based on feedback from scientists. The main section of the paper describes the architecture of the OntoSoft registry and its implementation, followed by conclusions and future work.

II. SOFTWARE REPOSITORIES AND OTHER RELATED WORK

In this section we discuss related work on capturing, describing, and sharing scientific software. In addition, we discuss the kinds of metadata used in code repositories, both general-purpose and specific to scientific domains. All this work has informed the design of OntoSoft.

A. Capturing Information about Software

The Core Software Ontology (CSO) and the Core Ontology of Software Components (COSC)¹ [4] extend the DOLCE ontology [5] to describe software components and web services. These ontologies were designed to describe large software systems, so their requirements include the

¹ <http://cos.ontoware.org>

accessibility of the software components, middleware services, execution failures, and composition of software. CSO formalizes concepts related to software and data, and includes both software components and services. COSC extends CSO to define software components further, and includes notions such as interaction protocols and taxonomies. There are several major departures from our goals in OntoSoft. First, these ontologies are very formalized and axiomatized based on the *Descriptions* and *Situations* ontology design pattern that captures how states change when actions are executed. This is not an aspect of software that is central to a software catalog. Second, such formalizations would place restrictions on what users must specify, which would require users to understand logic inference in ways that an average scientist would not know. Finally, they focus on complex software systems, rather than in end users who are scientists and need to describe software for other scientists.

Chue-Hong [6] proposes a framework for capturing information about software that promotes reusability. The framework consists of four levels: absolute minimum (L1), useful minimum (L2), pragmatic minimum (L3), and good minimum (L4). Each level contains information that is split into five categories: license (legal constraints), availability (discovery and accessibility), quality (understanding functional and non-functional characteristics), support (communicating with original developers), and incentive (rewarding developers). These levels build on one another, and the framework includes two additional levels at the extremes that represent theoretical minimum (L0, insufficient for reusability) and idealistic minimum (L5, too much required from the developer). For example, in the license category L0 requires that the software has any license, L1 that the license allow reuse, L3 that the license allows modification as well as reuse, and L4 that it is an approved open source license allowing modification and reuse (L2 and L5 are omitted for this category). The framework is only described informally, and is not specified as a model or ontology. The framework is complementary to OntoSoft, in that it could be incorporated as an extension by specifying required properties and values in each of the levels.

WICUS² [7] is an ontology for describing execution requirements of workflows. It has a complementary focus, and could be used in combination with OntoSoft as an extension to specify runtime requirements.

The TIMBUS Ontologies³ [8] propose a model to preserve and ensure the availability of business processes and their computational infrastructure, aligned with enterprise risk and business management. They also propose a semantic approach to describe execution environments of processes, which could be combined with OntoSoft. Even though TIMBUS has studied the applicability of their approach to scientific software, it is focused on business processes.

The Software Ontology (SWO)⁴ is a heavyweight ontology that aims to help describing software used by the curation and

data preservation community in domains ranging from text bioinformatics to social sciences. SWO extends some of the Open Biomedical Ontologies (OBO)⁵, like the Basic Formal Ontology (BFO)⁶ as foundational ontology and the Relations Ontology (RO)⁷ to describe software relationships. It also extends the EDAM Ontology⁸ for adapting different data formats and operations in the bioinformatics domain. However, the level of sophistication in formal logic of the SWO makes it less accessible to users who lack such expertise, as is the case with most scientists. Some of the SWO terms may be aligned with OntoSoft to allow describing further some aspects of software (i.e., software composition) that are outside the initial scope of OntoSoft.

CodeMeta⁹ is a recent effort to map across software metadata across repositories. It includes a mapping to the OntoSoft ontology.

In [9], the authors argue for the use of software security maturity models and apply them to characterize the levels of security and reliability of scientific software. These kinds of models have not been applied to scientific software, so they are not covered in our OntoSoft ontology but could be an extension of it.

B. General Software Repositories

Software repositories are widely used by scientists. They have different features and utility. Although they allow users to describe their software and often use standard conventions for doing so, they do not use an ontology or model to organize the descriptions of the software.

General code repositories are widely used for scientific software. GitHub¹⁰ is a repository that supports version control through the Git infrastructure. GitHub projects have a standard way to specify documentation (through readme files and collaborative Wiki pages) and licenses. BitBucket¹¹ is a private software repository that allows synchronizing through Git and Mercurial repositories. SourceForge¹² is a software repository for open source software projects, where users can provide an overview of the features of their project and point to the supported executables or installers for download.

Other code repositories are more focused on science, and attract contributions in many domains. CRAN¹³ is an archive of code written in the R language. CRAN is built collaboratively, based on a network of ftp and web servers that store up-to-date versions of code and documentation. The only requirement for submitting new code is to fill a short form and provide documentation. Similarly, PyPI¹⁴ is a software repository of Python codes [10], although scientific software tends to be in the SciPy repository [11] described below.

²<http://purl.org/net/wicus>

³<http://timbusproject.net/portal/publications/ontologies>

⁴<http://theswo.sourceforge.net>, <https://softwareontology.wordpress.com/>

⁵<http://www.obofoundry.org/>

⁶<http://www.obofoundry.org/cgi-bin/detail.cgi?id=bfo>

⁷<http://www.obofoundry.org/cgi-bin/detail.cgi?id=ro>

⁸<http://edamontology.org/>

⁹<https://github.com/codemeta>

¹⁰<http://www.github.com>

¹¹<https://bitbucket.org/>

¹²<http://sourceforge.net/>

¹³<http://cran.r-project.org/>

¹⁴<https://pypi.python.org/pypi>

FigShare¹⁵ and Zenodo¹⁶ are repositories for research artifacts (papers, blog posts, datasets, etc.) including software. Both allow users to describe all these artifacts with keywords, assign them DOIs, link to a code repository (e.g. GitHub), and add the corresponding license and publication date. An important feature of these repositories, is that they specify how to cite the software so authors can get credit.

Kaggle¹⁷ is a site that holds competitions for data analytics where companies and government agencies post their data and pose challenges and data mining experts around the world can submit entries to the challenge. Winning solutions have to be documented through a template¹⁸. The template includes details about the machine learning approach taken, such as feature selection and extraction, training procedures, and formation of model ensembles. It also includes the description of inputs and outputs of codes, their functions, their runtime dependencies, and detailed instructions to run them.

Workflow systems use domain-independent languages to describe the software components that are used as steps in the workflows [12,13,14,15]. These languages capture information about how the codes need to be invoked, basic use documentation, and execution information. Workflow repositories, such as myExperiment [16] and CrowdLabs [17], do not use ontologies to capture structured software descriptions of the individual workflow steps.

C. Software Repositories in Science Domains

There are several repositories that have been developed for specific science domains that have different approaches and rationale as well as practical experiences with the perceived benefits and incentives of collecting software metadata. We designed OntoSoft to cover all the metadata that these repositories collect, and more.

The Community Surface Dynamics Modeling System (CSDMS) contains hundreds of codes for models for Earth surface processes [18]. CSDMS also collects a comprehensive set of metadata for software, including authors, programming languages, pointers to code, licenses, and test datasets. It also assigns DOIs to models. Other modeling frameworks in geosciences include the Earth System Modeling Framework (ESMF)¹⁹ and the Computational Infrastructure for Geodynamics (CIG)²⁰. These frameworks support sophisticated model coupling capabilities to run several models in consonance, such as re-gridding to match the model scales and message passing across models to synchronize the processes they each model, which requires standard interfaces. These are aspects not covered by OntoSoft.

In [19] the authors describe the practical experiences with a software repository for astronomy, the Astrophysics Source Code Library (ASCL). A major community driver for this resource is that it gives authors the ability to cite software from the repository, and in addition it collects citations for each

software entry. Each software entry is described with five fields: 1) code name, 2) brief description, 3) authors, 4) URL to download the code, and 5) unique identifier for the software (the ASCL ID). The entry also includes a link to a paper describing the software, and links to papers using the software. Through interactions with the astronomy community, it was found that users prefer to keep any metadata together with the code, that they would rather use ASCL more as a registry than as a repository (jumping from 40 codes for several years to 700 entries in just 3 years), that the type of license or the quality of the code are not as important as having them registered and indexed, and that any information that changes over time (e.g., versions) should not be captured because it is too hard to track. Due to lack of resources, their focus is on metadata that enables identifying the code accurately and without ambiguity. [19] discuss other code repositories in astrophysics that were not so successful due to lack of exposure in the community.

nanoHUB [20] is a science gateway that contains for nanotechnology software and educational materials. A license is required, and approximately one-eighth of the codes use open source licenses. Code providers are encouraged to provide documentation for first time users, test suites, and a citation for the software. A key added value of the repository is the measures of quality of the software, which are contributed by its more than 330,000 users annually. They are tracked through usage statistics and citations. In addition, reviews and questions/answers are associated with each code, as well as wish lists from users. Exposing these indirect measures of quality incentivizes code authors to support their code. Citation is supported, but [20] reports that over a hundred tools have been cited once and only fourteen have been cited ten or more times. HUBzero [21] is the framework underlying nanoHUB, and it has been used to develop web sites in different domains. One such site is the hub for the volcanology community [22]. They recommend to include benchmarks and keep track of popular codes, to document usage limitations and scope of the codes so they are not wrongly utilized, and facilitate integration within a workflow.

SciPy [11] is an open-source library of scientific python code. It includes packages for mathematical computations, plot generation, and publishing of interactive results through iPython [23]. SciPy is only accessible and understandable by python programmers.

Some workflow systems include a significant amount of codes in a domain that can be used as workflow steps, including LONI Pipeline for neuroimaging genomics [24], GenePattern and Galaxy for genomics [25], and Taverna for bioinformatics services [12]. However, the descriptions of these codes are typically only structured in terms of inputs and outputs, and other information is simply text documentation.

III. REQUIREMENTS AND DESIGN OF ONTOSOFT

There are important requirements not met by current software repositories that motivate our design for OntoSoft:

- *A software registry to complement code repositories.*
There are already code repositories that offer important

¹⁵<http://figshare.com/>

¹⁶<https://zenodo.org/>

¹⁷<http://www.kaggle.com/>

¹⁸<https://www.kaggle.com/wiki/WinningModelDocumentationTemplate>

¹⁹<https://www.earthsystemcog.org/projects/esmf/>

²⁰<https://geodynamics.org/cig/>

functionality in terms of collaborative software development, version control, and community support. However, these repositories are not integrated with one another, and their registry aspects (i.e., the metadata that describes the software entries) are not very structured because they are included in readme files and other informal means. As a result, it is hard for a scientist to find software with desired properties, such as finding a software package for k-means, in Java, with a license that allows commercialization, and that takes data in NetCDF format. In addition, many scientists have limited software skills and find code repositories hard to use. They are more amenable to sharing code through a data repository such as Zenodo. There is a need for a registry that would describe software entries with proper metadata, while pointing to a repository to download the software itself.

- *A software metadata vocabulary designed for scientists and scientific software.* The information and documentation that is available in code repositories is typically centered on software installation, rather than on software sharing. Execution information is sometimes extracted automatically (e.g., from virtualization environments). There is little guidance on how to describe software to facilitate the kind of understanding that scientists require in order to trust it and use it to do their science. This includes for example understanding the assumptions made, or the projects or publications that use the software. New approaches for capturing scientific software metadata are needed.
- *A social approach to scientific software documentation.* Users are rarely excited about providing metadata – not for datasets and not for software. In addition, when considering reuse, scientists often want to know what other scientists thought of the software as they tried to use it. This requires opening software documentation beyond software authors. This also liberates the software authors from not only having to deliver the software but also the metadata. It also allows others to specify the metadata as they read through documentation and perhaps also the literature, and reflecting their own experiences with respect to usability and quality of the code. At the same time, authors should have some control over what is said about their software to ensure it is accurate. There is a need to open software metadata to contributions from the community, not just the software authors, while retaining some control to ensure utility and quality.

These requirements stem from informal surveys of scientists done as part of the OntoSoft project²¹, and have driven us to design the OntoSoft distributed semantic registry for scientific software metadata.

IV. THE ONTOsoft DISTRIBUTED SEMANTIC REGISTRY FOR SCIENTIFIC SOFTWARE

The metadata captured by OntoSoft is organized through an ontology, described in [26]. The software metadata properties

specified in the ontology are organized into six major categories based on information that a scientist would seek: 1) identify software, 2) understand and assess software, 3) execute software, 4) get support, 5) do research, and 6) update the software. Properties are marked as important or optional.

Figure 1 highlights the main features of OntoSoft described throughout this section. The user is shown indicators of metadata completeness. Some metadata is imported automatically from software catalogs. For example if the user specifies a GitHub site, OntoSoft will import metadata such as authors, contributors, and license. The user can export the software metadata in HTML, RDF and JSON, so they can include the metadata in their publications or attach it to their software.

A. Distributed Architecture

Each OntoSoft site has access to the content of other repositories, so software entries can be shown to users together with their source. Users can search for software based on semantic metadata properties, and get results for software in any of the OntoSoft sites.

Although OntoSoft is still a prototype under development, there are currently several OntoSoft sites deployed to describe software in different communities like Earth systems modeling, paleoclimatology, and geospace sciences, and environmental omics²². There are currently more than 600 software entries described in OntoSoft.

B. Crowdsourcing Metadata through Access Control Policies

Software authors can open the metadata editing selectively to others. This means that the original software developers are not necessarily responsible for providing all metadata, which can be provided by those who benefit from using the software. Software authors should be able to decide what metadata they are willing to crowdsource. For example, they may want to allow others to edit the metadata about uses of their software, but not metadata about the version releases. In some cases, permission to edit may only be given to selected contributors.

Our approach is to give authors access control mechanisms. OntoSoft extends the W3C WebAccessControl ontology.²³ Authorization is generally implemented using access control lists (ACLs), which consist of a series of access control instructions that either allow or deny permissions (such as read, write, etc.) to specified entries and their attributes. The WebAccessControl ontology incorporates these concepts by defining acl:Authorization, an abstract entity whose properties (acl:accessMode, acl:accessTo, and acl:agent) are defined in an ACL. In the ontology, acl:InformationResource can refer to either a software entry or to a metadata property of a software entry, which allows finer-grained software control.

Figure 2 illustrates this fine-grained access control. Here, a software author has created a software *software1* and defined two specific permissions. Permission *auth1* specifies that user *user1* has read access to any metadata property of *software1*. A second permission *auth2* specifies that user *user2* has write access to the metadata property *hasUseStatistics*. The corresponding access control list entries are:

²¹ <http://ontosoft.org/>

²² <http://www.ontosoft.org/portals/>

²³ <https://www.w3.org/wiki/WebAccessControl>

Software Repository

Describe your software so others can find it.

Software entries from distributed repositories are readily accessible

Semantic search

Automatic import of metadata from other repositories

Metadata properties collected through simple questions

Crowdsourcing of metadata through access control permissions

Comparison matrix of software entries

Metadata exported in HTML, RDF, JSON

Metadata completion highlighted

Metadata properties organized into categories that make sense to scientists

Fig. 1. An overview of the user interface of the OntoSoft editor for software metadata properties, highlighting its main features as a distributed architecture, its scientist-centered design, and its metadata crowdsourcing approach.

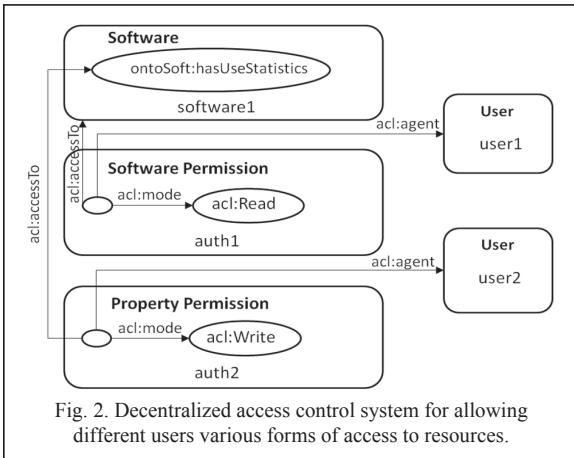
```
[acl:accessTo <http://www.ontosoft.org/portal/software/software1>;
  acl:mode acl:Read;
  acl:agent <http://www.ontosoft.org/portal/users/user1>]
[acl:accessTo <http://www.ontosoft.org/portal/software#hasUseStatistics>;
  acl:mode acl:Write;
  acl:agent <http://www.ontosoft.org/portal/users/user2>]
```

OntoSoft uses Discretionary Access Control (DAC) [27], which allows the owner of a resource to determine which users can access it. Users are given one of three roles for each software entry, which are used to set permissions: software owners, software editors, and metadata editors. A *software owner* has complete access to all the aspects of a software entry, and can make others editors of the entire entry or editors of specific metadata properties. Software owners have

permissions to delete the entry, add or remove software editors, add or remove property editors, and edit any properties. A *software editor* has permission to edit all the metadata properties of a software entry. A *metadata editor* has permission to update properties they have been granted write access to for a given software entry. Figure 1 shows in a pop-up window how permissions can be granted to users. Each OntoSoft site can be configured with defaults for new entries.

C. Querying Distributed Software Registries

Each OntoSoft site has a public API that can be used to retrieve a list of all software entries and all the metadata for a particular software entry. The results can be in different serializations (e.g., JSON, RDF/XML). Each OntoSoft site



shows its own software entries by default. Each site can be configured to index the software entries from other sites, aggregating all their data so it can be efficiently queried. This allows users to view, filter, and compare entries from other sites. When a user is logged into a site and selects a software entry from another site, the user is redirected to the page of that software entry in the site in which the entry was created. With this approach, the information is decentralized and each community is responsible for their own software entries while enabling others to search and compare contents from their site.

V. CONCLUSIONS

OntoSoft is a semantic registry for capturing scientific software metadata that provides the means to crowdsource software metadata while granting software owners control over who can modify their entries. OntoSoft has a distributed architecture, so that different communities can run their own sites while keeping them all interconnected. This enables a distributed query capability so that users can search software entries across different OntoSoft sites. We plan to enable cross-site user registration and access control so that users can login with the same credentials and access rights across sites. Future work also includes improving integration with existing code repositories, and developing a recommender system.

ACKNOWLEDGMENTS

We would like to thank other members of the OntoSoft project, including Scott Peckham, Chris Mattmann, Kaijian Xu, Erin Robinson, and Chris Duffy. We would also like to thank the many early adopters of OntoSoft for their feedback and comments on this work, and in particular Cedric David, Leslie Hsu, Anna Kelbert, and Sandra Villamizar.

REFERENCES

- [1] Heidorn, P.B. "Shedding Light on the Dark Data in the Long Tail of Science." *Library Trends*, Vol. 57, No. 2, Fall 2008.
- [2] Garijo, D.; Alper, P.; Belhajjame, K.; Corcho, O.; Gil, Y.; and Goble, C. Common Motifs in Scientific Workflows: An Empirical Analysis.. *Future Generation Computer Systems*, . 2013.
- [3] Barnes, N. Publish your computer code: It is good enough. *Nature* 467, 753, 2010. doi:10.1038/467753a
- [4] Oberle D., Lamparter S., Grimm S., Vrandecic D., Staab S., Gangemi A. "Towards Ontologies for Formalizing Modularization and Communication in Large Software Systems." *Journal of Applied Ontology*, Vol. 1, No. 2, 2006.
- [5] Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., Schneider, L. "Sweetening Ontologies with DOLCE." *Proceedings of the 13th Conference on Knowledge Engineering and Knowledge Management (EKAW)*, 2002.
- [6] Chue-Hong, N. "Minimal information for reusable scientific software." Presented at the Second Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE2), 2014. figshare. <http://dx.doi.org/10.6084/m9.figshare.1112528>.
- [7] Santana-Perez, I., Pérez-Hernández, M. "Towards Reproducibility in Scientific Workflows: An Infrastructure-Based Approach" *Scientific Programming*, vol. 2015, 2015.
- [8] Mayer, R. Miksa, T. and A. Rauber. Ontologies for describing the context of scientific experiment processes, in: 10th International Conference on e-Science, 2014.
- [9] Heiland, R., Thomas, B., Welch, C. and C. Jackson. "Towards a Research Software Security Maturity Model." Presented at the First Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE1), 2013, <http://arxiv.org/abs/1309.1677>
- [10] Terrel, A. "Sustaining the Python Scientific Software Community." Presented at the First Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE1), 2013. figshare. <http://dx.doi.org/10.6084/m9.figshare.791565>
- [11] Jones E., Oliphant E., Peterson P., et al. SciPy: Open Source Scientific Tools for Python, 2001, <http://www.scipy.org/> [Accessed 2015-03-13].
- [12] Missier, P., Soiland-Reyes, S., Owen, S., Tan, W. et al. Taverna, reloaded. In 22nd International Conference on Scientific and Statistical Database Management (SSDBM), Heidelberg, Germany, 2010.
- [13] Ludaescher, B., Altintas, I., and Berkley, C., Higgins, D., Jaeger, E., et al. "Scientific workflow management and the Kepler system." *Concurrency and Computation: Practice and Experience*. Vol 18, 2006.
- [14] Gil, Y., Ratnakar, V., Kim, J., Gonzalez-Calero, P., Groth, P., Moody, J. and E. Deelman. "Wings: Intelligent Workflow-Based Design of Computational Experiments." *IEEE Intelligent Systems*, 26(1). 2011.
- [15] Gil, Y. "Intelligent Workflow Systems and Provenance-Aware Software. Proceedings of the Seventh International Congress on Environmental Modeling and Software, 2014.
- [16] De Roure, D., Goble, C. and R. Stevens. "The design and realizations of the myExperiment Virtual Research Environment for social sharing of workflows". *Future Generation Computer Systems*, 25 (561-567), 2009.
- [17] Mates, P., Santos, E., Freire, J. and C.T. Silva. CrowdLabs: Social analysis and visualization for the sciences. 23rd International Conference on Scientific and Statistical Database Management, 2011.
- [18] Peckham, S. D., Hutton, E.W.H. and Norris, B. "A component-based approach to integrated modeling in the geosciences: The design of CSDMS." *Computers and Geosciences*, 53, 2013.
- [19] Shamir, L., Wallin, J.F., Allen, A., Berriman, B., Teuben, P., Nemiroff, R.J., Mink, J., Hanisch, R.J., K. DuPrie. "Practices in source code sharing in astrophysics." *Astronomy and Computing*, Vol 1, 2013.
- [20] Zentner, L., Zentner, M., Farnsworth, V., et al. "nanoHUB.org: Experiences and Challenges in Software Sustainability for a Large Scientific Community." *Journal of Open Research Software*, 2(1), 2014.
- [21] McLennan M. and R. Kennell. "HUBzero: A Platform for Dissemination and Collaboration in Computational Science and Engineering." *Computing in Science & Engineering*, 12(2), 2010.
- [22] Patra, A., Jones, M., Gallo, S., et al "Role of Online Platforms, Communications and Workflows in Developing Sustainable Software for Science Communities." Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE2), 2014. figshare. <http://dx.doi.org/10.6084/m9.figshare.1112569>.
- [23] Pérez, F. and B.E. Granger. IPython: A System for Interactive Scientific Computing, *Computing in Science and Engineering* 9(3), 2007.
- [24] Torri, F., Clark, A.P. Dinov, I. Zamanyan, A. et al. Next generation sequence analysis and computational genomics using graphical pipeline workflows. *Genes*, 3:545-575. 2012.
- [25] B. Giardine et al. Galaxy: A platform for interactive large-scale genome analysis. *Genome Research*, 15(10):1451-1455, 2005.
- [26] Gil, Y.; Ratnakar, V.; and Garijo, D. OntoSoft: Capturing Scientific Software Metadata. In *Proceedings of the Eighth ACM International Conference on Knowledge Capture*, 2015.
- [27] Barkley, J. Comparing simple role based access control models and access control lists. *Proceedings of the Second ACM Workshop on Role-Based Access Control*, 1997.

A Secure Data Enclave and Analytics Platform for Social Scientists

Yadu N. Babuji, Kyle Chard, Aaron Gerow, Eamon Duede

Computation Institute

University of Chicago and Argonne National Laboratory

{yadunand,chard,gerow,eduude} @uchicago.edu

Abstract—Data-driven research is increasingly ubiquitous and data itself is a defining asset for researchers, particularly in the computational social sciences and humanities. Entire careers and research communities are built around valuable, proprietary or sensitive datasets. However, many existing computation resources fail to support secure and cost-effective storage of data while also enabling secure and flexible analysis of the data. To address these needs we present CLOUD KOTTA, a cloud-based architecture for the secure management and analysis of social science data. CLOUD KOTTA leverages reliable, secure, and scalable cloud resources to deliver capabilities to users, and removes the need for users to manage complicated infrastructure. CLOUD KOTTA implements automated, cost-aware models for efficiently provisioning tiered storage and automatically scaled compute resources. CLOUD KOTTA has been used in production for several months and currently manages approximately 10TB of data and has been used to process more than 5TB of data with over 75,000 CPU hours. It has been used for a broad variety of text analysis workflows, matrix factorization, and various machine learning algorithms, and more broadly, it supports fast, secure and cost-effective research.

I. INTRODUCTION

Data is fast becoming a crucial, defining, asset for researchers. Entire fields, including those new to computational practices, are quickly embracing data-driven research. However, the increasing scale and complexity of analysis and the fact that datasets are often proprietary, or sensitive, creates unique new challenges. The centrality of data has inspired new processes that are designed for specific datasets, which typically results in tightly coupled environments that discourage reusability and agility. To support the needs of data-driven research, we developed CLOUD KOTTA¹, a unique cloud-based framework that enables the secure and cost-effective management and analysis of large, potentially sensitive datasets.

To address the growing reliance on data in research (particularly in the social sciences and humanities) scholars are increasingly replacing on-premise infrastructure with cloud-based solutions such as those offered by Amazon Web Services (AWS). This trend is not difficult to explain: cloud platforms provide high reliability, availability, and download performance without encumbering researchers with managing on-site infrastructure. The adoption of cloud-based services has also afforded new avenues for exploration. For example, when storage is co-located with elastic computing capacity with which data can be analyzed, aggregated, and integrated

on-demand, researchers can take bigger risks and explore new analyses more flexibly. CLOUD KOTTA enables this kind of agility across fluid groups while also ensuring scalability, security, and data provenance.

Cloud-based infrastructure also has the advantage of helping centralize disparate teams. For example, given the sensitivity, value, and size of many datasets, it is often not feasible to replicate and download entire datasets for analysis on local computers or clusters. In many cases, circuitous approval procedures are necessary to gain access to data, and users must adhere to strict data-use agreements. Migrating data from the environment where it is hosted adds further complexity in this respect, and cloud-based strategies can exacerbate these challenges. So, when it comes to cloud-based infrastructure, it is important for researchers to develop a unified strategy. CLOUD KOTTA offers a strategy that is cost-effective, secure with respect to data policies, offers sustainable short- to long-term storage, and is scalable for demanding workloads.

CLOUD KOTTA is designed to address the requirements of two canonical use cases: managing community datasets securely and providing scalable compute resources. The first use case is motivated by a growing need for researchers to make valuable datasets available to certain research communities. This might be required by funding agencies or institutions, though, it is generally helpful to share data when establishing new research groups. The most important requirements for this use case are that CLOUD KOTTA be:

- **Secure:** Data should be stored securely and accessible only to authorized and authenticated users.
- **Scalable:** Data can be large, the storage system should scale to the data.
- **Reliable:** Data should be stored reliably, using backups in case of failure or corruption.
- **Available:** Data should be available to geographically distributed users.
- **High performance:** Large amounts of data should be moveable easily and quickly for analysis, download, and archival.
- **Cost-effective:** Costs associated with data storage should be minimized to encourage use.

The second use case is motivated by a large-scale movement towards data-intensive research. As data sizes grow and analyses become more computationally intensive, the

¹ Available at https://github.com/yadudoc/cloud_kotta.

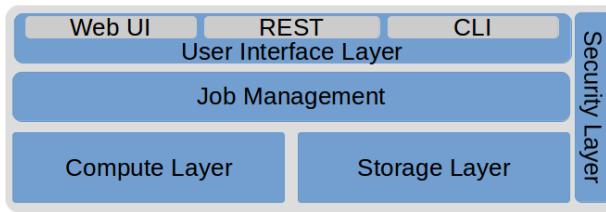


Fig. 1. Logical Architecture of CLOUD KOTTA

requirements often exceed the computational capabilities of individual researchers. As such, researchers need to be able to scale analyses from individual computers to distributed, parallel modes of computing. To address these priorities, CLOUD KOTTA should be:

- **Secure:** Authorizations should control what data can be analyzed and which analyses should be isolated.
- **Scalable:** Analyses should scale to data, exploit parallelism where possible, and leverage large scale computing infrastructure for efficient performance.
- **Cost-effective:** Costs should be comparable or lower than using local compute resources.
- **Easy to use:** Interfaces should make it simple to access the underlying infrastructure.

II. ARCHITECTURE & IMPLEMENTATION

The CLOUD KOTTA architecture is depicted in Fig. 1. The entire system is comprised of a web interface and REST API for accessibility; a set of event-based management and monitoring software to ensure reliable job execution; a compute layer that provides cost efficient compute resources; a fast and cost-effective storage layer; and an extensible and customizable security fabric that permeates all of the above components.

CLOUD KOTTA is designed to be deployed on *Amazon Web Services* (AWS), the ecosystem of its intended users. Where possible, CLOUD KOTTA leverages existing cloud services as they are scalable, reliable, secure, and cost-effective. The entire CLOUD KOTTA system is open source and can be deployed using a reproducible *CloudFormation* configuration which can be further customized to match target workloads.

A. User Interface

CLOUD KOTTA offers three interfaces: a web interface, a REST API, and a command line interface (CLI) accessible from the login node. This range of interfaces supports broad usage scenarios, enabling intuitive web access for web-based users and advanced programmatic and CLI support to facilitate customizable and automated invocation by technical users.

The supported interfaces support the same operations including, for example, browse datasets, upload new data, view and download results from previous analyses, submit and manage analyses. As with the entire CLOUD KOTTA architecture, the interfaces are secured, restricting access to authenticated, authorized users. The general architecture is centered around data stored in AWS Simple Storage Service (S3) buckets.

Users can browse accessible data that they are permitted to access in S3, and they can also upload files to their own private S3 buckets. Once uploaded, files are available to be specified as inputs to submitted jobs. To support dynamic sharing scenarios, such as emailing colleagues the results of an analysis, CLOUD KOTTA provides support to construct short-term, anonymous URLs.

Submitting an analysis requires a description of the application (scripts, executables, etc), a list of inputs (S3, external URLs), a list of output files to be saved, and a maximum wall-time. In addition to supporting jobs with arbitrary executables, applications can be templated to create pipelines with simplified user interfaces. That is, other users wishing to reuse an analysis can simply complete a customized form with the required parameters. Users submit jobs via web forms in the Web interface, or by specifying the job as a JSON file for the CLI and REST interfaces.

B. Storage Layer

CLOUD KOTTA's storage layer uses a mix of AWS storage services that provide different guarantees regarding access time, durability, and availability with different cost models. The types of storage used by CLOUD KOTTA are:

- **Elastic Block Storage (EBS):** a high performance block storage model that can be mounted as a file system on an EC2 instance.
- **S3 standard:** a reliable object store that provides high performance access via HTTP(S).
- **S3 infrequent access:** an object store with reduced storage cost at the expense of availability.
- **Glacier:** an archival storage model that provides high durability at a low price with high data retrieval times.

CLOUD KOTTA utilizes a caching model that is implemented using automated data life-cycle policies that manage data migration between storage tiers based on access patterns. Frequently accessed data resides on S3, as it is fast and highly available, whereas data that is accessed infrequently is moved to Glacier, as it provides durable, low cost storage at the expense of longer retrieval time. Fig. 2 illustrates the storage tiers used in CLOUD KOTTA's data model. Transferring data to lower tiers helps minimize the cost associated with providing high availability. The primary store for data in CLOUD KOTTA is S3. When data is analyzed, it can either be staged directly from S3 to ephemeral instance storage or EBS (which is subsequently mounted by an instance). Similarly, archived data stored in Glacier or S3 infrequent access buckets is staged to S3 when needed before being staged for analysis. Outputs are staged back to S3, guaranteeing durability.

CLOUD KOTTA's data model has important advantages over a static storage configuration. While EBS provides low-latency access, it is more than three times as expensive as S3-Standard and, in addition, must be mounted as a file system on a live machine to access data. By storing data in S3, a small overhead is incurred to stage data for compute, however, this latency is nominal in most cases and comprises a fraction of the time it takes to provision and execute a job. In addition to the cost

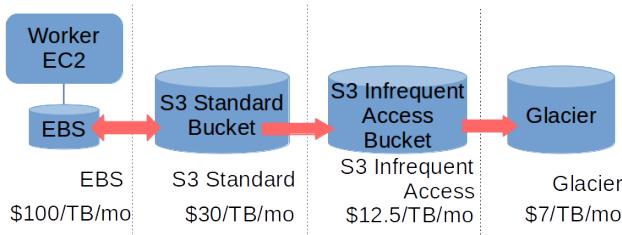


Fig. 2. Storage tiers in CLOUD KOTTA and the heuristics used to minimize storage costs. Some costs may differ across regions and configurations.

benefits, S3 provides support for rich access control features, as well as the ability to publish and access data directly via HTTP.

C. Compute Layer

The analyses for which CLOUD KOTTA is designed often comprise independent, long running, loosely coupled jobs. To support this class of workload, CLOUD KOTTA offers a scalable compute layer built upon elastic pools of Elastic Compute Cloud (EC2) instances.

EC2 is a virtualized computing environment in which users can lease virtual machines (VM) with varying computational resources. Instances are organized by region and *Availability Zone* (AZ). Regions represent different geographic locations whereas AZs are datacenters located within a specific region. Pricing models differ between AZs and, due to the independent failure models, users can achieve high reliability by distributing applications across AZs. EC2 instances are provisioned according to a market model in which users pay for the resources consumed.

CLOUD KOTTA can be configured to use two different EC2 market models. **On-demand** instances are offered at a fixed hourly price where instances live until they are terminated by a user. **Spot** instances are offered using a dynamic price model where users “bid” a maximum hourly price and instances are terminated by AWS if the market price exceeds the user’s bid. Spot markets are typically a fraction of the on-demand price.

CLOUD KOTTA is designed to support two classes of workloads: short development jobs requiring quick responses, with minimal compute resources, and longer running production tasks that are computationally intensive, but more tolerant of delays. To meet the needs of these two asymmetric workloads, CLOUD KOTTA offers two independent pools of compute resources. The development pool is comprised of on-demand instances, with at least one instance accessible at all times. To minimize the cost of executing computationally intensive, long running, yet delay tolerant production workloads, we utilize spot instances. CLOUD KOTTA relies on an automated bidding model to provision resources across AZs (to avoid price fluctuations in an AZ). Administrators can configure the bidding model to use static or policy-based bid prices (some fraction of the equivalent on-demand price, for example).

While using spot instances can significantly reduce costs, instance revocations are inevitable. To mitigate the problems

associated with instance termination, CLOUD KOTTA manages queues that ensure jobs that do not complete are resubmitted to the queue and executed again on a new instance. By provisioning spot instances on-demand, CLOUD KOTTA can meet the demands of what are often sporadic and bursty workloads while also helping minimize costs [1]. CLOUD KOTTA currently uses a pre-defined EC2 instance type for each of its queues. In future work, we will integrate CLOUD KOTTA with cost-aware provisioning [2], [1] and profiling [3] approaches to improve the selection of instance types based on cost and execution time.

D. Job Management

CLOUD KOTTA uses a job management layer to control the execution of arbitrary user analyses on the compute layer. User submitted tasks include the input files, execution scripts, and output files. Users must also choose if the task is a development or production job. When tasks are submitted, the description is stored in a database such that it can be accessed by the job management layer and the instances executing the analysis. To execute the task, the job management layer determines the user’s access permissions, associates the user’s role with the description, and places the job in the appropriate queue.

We leverage a queue model as it provides a reliable method for distributing jobs across a pool of EC2 instances. Worker nodes (EC2 instances) poll the queue for waiting tasks. If a task is available the worker moves it from a pending queue to the active queue. This active queue is used to manage execution and ensure that no tasks are lost. The worker retrieves the task description from the database and begins execution. In the case where spot instances are used, we must account for unreliability of the underlying infrastructure. To do so, the job management layer includes a monitoring service that periodically checks instance health (e.g., for termination or other failures). If instances are terminated, the monitoring service will resubmit the task to the pending queue. Throughout execution, the worker node writes job status markers to the database. This information provides worker statistics (CPU, I/O and RAM utilization) and job progress, both of which are accessible to the user to monitor job execution. Upon completion, the worker will stage output data to S3, and update the database with the completion code of the task.

E. Security

The final layer of CLOUD KOTTA is the security fabric that permeates the system. CLOUD KOTTA uses Amazon’s OAuth 2 model (*Login with Amazon*) for authentication. Users are able to login using their Amazon credentials and CLOUD KOTTA is therefore not responsible for managing user passwords. Before being granted access to the system, users must first be registered in CLOUD KOTTA’s database and given an appropriate role. When users login using the OAuth 2 workflow, CLOUD KOTTA is given a short-term delegated access token. This token can be used to retrieve information about the user from Amazon as well as to use services as the user.

CLOUD KOTTA is built around a role-based access control model in which users are assigned roles, for example *kotta-public-only* and *kotta-read-WOS-private*, where *WOS* refers to the private Web of Science dataset. Policies associated with roles define permissions for specific resources (e.g., data access in S3). Given that all access is controlled by roles, worker nodes must assume a role before they can access restricted data. Other CLOUD KOTTA services are also given appropriate privileges by internal roles such as *web-server*, *task-executor*. These roles, unlike user roles, have access to the internal database, queues, notification systems and are capable of controlling scaling functionality. This role based access model limits validity of credentials to a small window limiting the risk of exposing valuable long-lived credentials. To adhere with the principle of least privilege, CLOUD KOTTA users are initially given no roles or privileges. They are incrementally granted permissions when required.

Data authorization and access control is implemented on S3 buckets. By default, access is not permitted unless it is explicitly granted via a policy. S3 buckets are associated with policies that prescribe permissions. Policies are then associated with roles that give permissions to users. The data stored on S3 buckets are server-side encrypted and accessible only from a Virtual Private Cloud (VPC) Endpoint. This guarantees that traffic between the S3 bucket and the compute instances remain private. Output data is also stored in S3. It is initially created as a private object only accessible to the creator. As CLOUD KOTTA is used by collaborating groups of users, it is important that data can be shared. To provide this capability, we use short-term signed URLs, like those used by other Cloud services (e.g., Google Drive and DropBox) that provide short term access to the holder of the URL.

CLOUD KOTTA implements a strong security model between instances and other services. The compute layer is hosted within a private subnet enclosed within a VPC. This ensures that compute instances are not directly accessible via the internet. Worker nodes are associated with a *task-executor* role that has few privileges (e.g., it cannot access any data). However, this is a trusted role that can be used to change to a user role for a short period of time. This is crucial as it enables a worker node, executing on behalf of a user, the ability to inherit the user's role and therefore access any data needed by the job. This approach ensures that even a running task is only able to access data for which the user is authorized to access. After staging data, the worker returns to the *task-executor* role to execute the job.

Finally, CLOUD KOTTA records every action performed by the system to ensure that data access and usage can be thoroughly audited. This information is recorded in a database such that administrators can export an audit log for any dataset, user, or service.

III. USAGE & APPLICATIONS

CLOUD KOTTA has been deployed and used over the past six months by a range of computational social scientists.

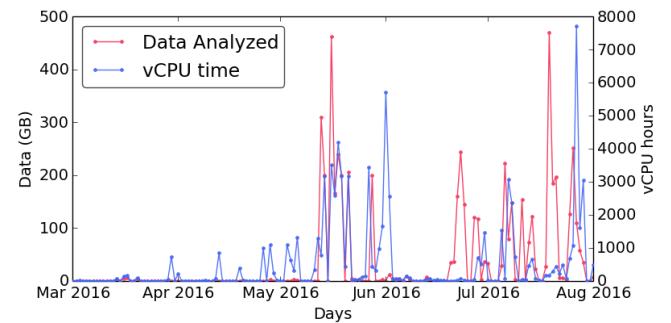


Fig. 3. Monthly usage of CLOUD KOTTA since deployment.

Active use cases for CLOUD KOTTA include text analysis, semantic word embedding, matrix factorization, optical character recognition, and social network analysis. Here we describe usage and illustrate some representative analyses.

A. Usage

CLOUD KOTTA has been used to develop, test and run a broad range of analytics on an array of datasets. CLOUD KOTTA currently manages datasets that collectively total nearly 10TB of data. It has been used for varied development cycles ranging from researchers running off-the-shelf tools on sample data, to teams of programmers developing and testing new methods on large, proprietary datasets. Throughout its development, CLOUD KOTTA was designed to speed up development and analysis cycles in a secure and flexible manner, while reducing costs induced by disparate and often idle compute resources. To-date, CLOUD KOTTA has been used to process over 5TB of data with over 75,330 CPU-hours. As our implementation of CLOUD KOTTA has solidified and become more robust, usage has grown (see Figure 3). The observed usage patterns affirm the choice of elastic cloud computing infrastructure as both data access and compute usage are particularly sporadic with peaks of over 7,000 compute hours in a single day and other days with none.

B. Applications

CLOUD KOTTA currently hosts a number of proprietary and sensitive datasets that have been used for a variety of workloads. CLOUD KOTTA has been primarily used to develop and run time-intensive text analyses, large scale matrix factorization, and optical character recognition (OCR). Other use cases have also been deployed with CLOUD KOTTA, but these three exemplify cases for which CLOUD KOTTA was designed: they are exploratory, large scale, and require private data that is shared among a strict set of users.

1) *Text Analysis*: One of the first use cases that was developed to run on CLOUD KOTTA is a tool-chain for text analysis. The analysis, designed to run on large collections of text, consists of four phases, each with separate inputs. The jobs consist of a series of pre- and post-processing scripts and wrapping natural language models. For these jobs, data is normalized and divided into logical bins, submitted to semantic analysis and post-processed to provide organized output.

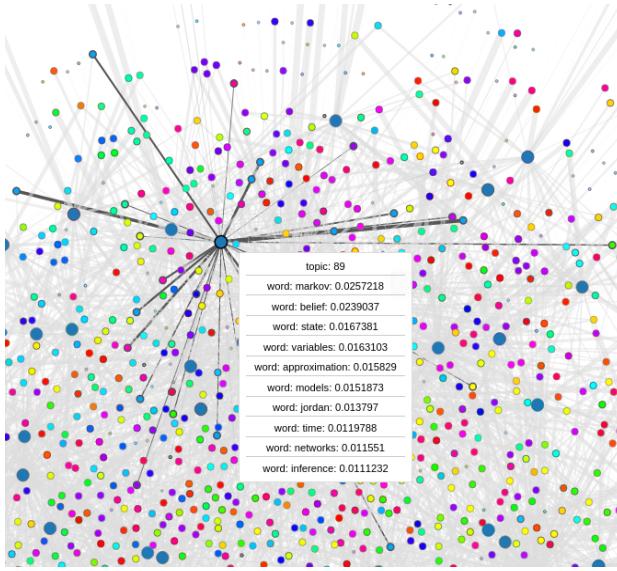


Fig. 4. Interactive analytics based on the analysis of researchers' publications. Shown is the author-to-topic network highlighting connections via topic #89, about Markov models.

The semantic model – the central analysis – includes doc2vec, word2vec [4] and various probabilistic topic models [5], [6]. These tend to be memory and compute intensive.

This workflow has unique challenges that can discourage exploration and slow development. It tends to involve free parameters at various phases, the scaling profile of many semantic models is unpredictable and the veracity of results is often measured qualitatively, making exploratory runs crucial. CLOUD KOTTA allowed our users to efficiently explore the parameter space to optimize a models' specification. One particular example is the Author-Topic (AT) Model [5], a probabilistic graph model that fits distributions of words, documents and authors to topics observed in texts. With CLOUD KOTTA, a multi-dimensional grid search was performed to assess the quality of various specifications attenuating the number of topics, and various fitting parameters that affect the interpretability of topics. Each run took approximately three days and 16GB of RAM. The final outputs of these runs were used to develop an interactive platform for researchers (Fig. 4) to generate explore commonalities and to propose future collaborators based on their existing work [7].

2) *Matrix Factorization*: Another use case where CLOUD KOTTA has been used was in developing a new method of multiple imputation (MI). When faced with lossy data, researchers often use MI to fill in missing values. Traditional MI establishes a missingness pattern on a given response which is then used in a regression with non-missing responses as parameters. The "multiple" aspect of MI is that after being imputed, new values can be used to increase the accuracy of imputing still-missing values. MI is cross-validated to assess stability and provide error-bounds. As a result, MI is computationally intensive and imposes strong statistical assumptions. A more fundamental weakness of parametric MI, however,

is that it disregards latent structure in the response matrix. Low rank and low norm matrix factorization offer alternatives to parametric MI that can exploit patterns throughout the response structure [8].

In developing and testing low rank and low norm alternatives to MI, CLOUD KOTTA was used to run a range of tests simultaneously. These models were implemented in *Julia*, a relatively new scientific programming language. Because there is a stochastic component to low rank and low norm models, cross-validation on a held-out set of data was used to evaluate results over many random folds. Once the models were developed, CLOUD KOTTA was able to execute a large batch of validation jobs to provide pooled results. In these sets, a single run on a 2,500 by 122 matrix used 32 cores on a single instance and 100 GB of RAM for 10 hours. This made CLOUD KOTTA's ability to run multiple jobs in parallel a crucial feature over the course of development.

3) *Optical Character Recognition*: A third application for which CLOUD KOTTA has been used in production is OCR. OCR is the process of extracting text, figures, tables, and other features from rasterized images of documents. OCR is effectively a kind of object recognition that relies on trained models of character classification – models that are computationally intensive. CLOUD KOTTA was used to run OCR software on over 10 thousand grant proposals and scholarly texts. Processing these documents required 20 hours using 10, 32-core instances and 75GB of RAM. With CLOUD KOTTA, what would have taken over a month on personal hardware, was finished in a single day.

IV. RELATED WORK

Computational social science communities are investigating a broad range of approaches for hosting proprietary datasets and conducting scalable analytics. For example, researchers have used hybrid cloud models [9], extended common tools to analyze data at scale [10], and developed environments for securely analyzing data in controlled VMs [11]. CLOUD KOTTA is unique in its support for a wide range of data, a general architecture that accommodates many use cases, and by its automated, scalable storage and analytics environments.

Many scientific communities now have a broad range of data repositories available for storing and accessing different types of data (e.g., biomedicine [12], climate [13], and astronomy [14]). Systems are typically developed around a static data repository that requires significant administrative overhead to populate, curate and manage. Each has independent identity management sub-systems that have been developed to control access to data. But more importantly, most existing systems are static, isolated data environments, that provide minimal management capabilities separate from compute resources. Science gateways [15] aim to bridge this gap by abstracting the complexity of using large scale computing infrastructure. These systems typically provide access to shared datasets (e.g., in a repository) and resources through high level interfaces (workflows, portals, etc.). Examples of

commonly used gateways include CyberGIS [16] for geoscience and iPlant [17] for ecology. Most science gateways are built on more traditional High Performance Computing (HPC) infrastructure. However, recent work has focused on cloud-based solutions [18], [19]. CLOUD KOTTA acts as a fabric on which next generation data repositories and cloud-hosted gateways could be developed in a domain-agnostic setting.

CLOUD KOTTA can deploy customized, cloud-based clusters similar to a number of other systems. For example, Cloud-Man [20] and StarCluster [21] allow users to deploy clusters for hosting and executing workflows. These systems, and others, are designed to aid the creation of clusters for semi-permanent usage. Other systems, such as Globus Galaxies [19] and Makeflow [22], enable on-demand and elastic cluster provisioning in response to workload. CLOUD KOTTA is unique, however, in its use of commodity AWS services and its broad focus on providing a framework for secure data storage and analysis.

V. SUMMARY

CLOUD KOTTA provides a secure and scalable data enclave and analytics environment for computational and data-drive social sciences. It addresses a gaping hole in the current infrastructure available to researchers, providing a model via which, even resource limited researchers can gain access to scalable data storage, elastic computing capacity, and cutting edge analysis algorithms without deploying and operating their own infrastructure. Moreover, it provides these capabilities while also optimizing performance and cost using automated data management and compute provisioning techniques.

In the six months since deployment CLOUD KOTTA has quickly grown to host a dozen private datasets (e.g., IEEE, Web of Science, and ACM) as well as several public datasets (e.g., US patents and Wikipedia). It has been used by dozens of researchers, students, and teachers to perform a wide variety of text analysis, machine learning, image recognition, and network analysis algorithms.

Our future work focuses on building an ecosystem around CLOUD KOTTA by developing a suite of data analytics frameworks from which users can more easily conduct analyses. We will continue to engage social scientists to add datasets to the system while looking to extend its capabilities to other disciplines. We are particularly interested in further developing algorithms for improving data lifecycle and compute management to better meet the needs of users with respect to performance, time, and cost.

ACKNOWLEDGMENTS

The authors thank Nandana Sengupta, Nathan Bartley, and Cha Chen for developing applications on CLOUD KOTTA . This research was supported by grants from the John Templeton Foundation to the Metaknowledge Research Network, IBM for Computational Creativity, and a gift from Facebook.

REFERENCES

- [1] R. Chard, K. Chard, K. Bubendorfer, L. Lacinski, R. Madduri, and I. Foster, "Cost-aware cloud provisioning," in *Proceedings of the 11th International Conference on e-Science*, August 2015, pp. 136–144.
- [2] R. Chard, K. Chard, K. Bubendorfer, L. Lacinski, R. Madduri, and I. Foster, "Cost-aware elastic cloud provisioning for scientific workloads," in *Proceedings of the 8th International Conference on Cloud Computing (CLOUD)*, June 2015, pp. 971–974.
- [3] R. Chard, K. Chard, B. Ng, K. Bubendorfer, A. Rodriguez, R. Madduri, and I. Foster, "An automated tool profiling service for the cloud," in *Proceedings of the 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, May 2016, pp. 223–232.
- [4] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.
- [5] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth, "The author-topic model for authors and documents," in *Proceedings of the 20th conference on Uncertainty in artificial intelligence*. AUAI Press, 2004, pp. 487–494.
- [6] J. Zhang, A. Gerow, J. Altosaar, J. Evans, and R. J. So, "Fast, flexible models for discovering topic correlation across weakly-related collections," in *Proceedings of Empirical Methods in Natural Language Processing*, 2015.
- [7] A. Gerow, B. Lou, E. Duede, and J. Evans, "Proposing ties in a dense hypergraph of academics," in *Social Informatics*. Springer, 2015, pp. 209–226.
- [8] M. Udell, C. Horn, R. Zadeh, and S. Boyd, "Generalized low rank models," *arXiv preprint arXiv:1410.0342*, 2014.
- [9] S. Abramson, W. Horka, and L. Wisniewski, "A hybrid cloud architecture for a social science research computing data center," in *Proceedings of the 34th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, June 2014, pp. 45–50.
- [10] M. A. Saleem, B. Varghese, and A. Barker, "Bigexcel: A web-based framework for exploring big data in social sciences," in *Proceedings of the IEEE International Conference on Big Data (Big Data)*, Oct 2014, pp. 84–91.
- [11] J. Zeng, G. Ruan, A. Crowell, A. Prakash, and B. Plale, "Cloud computing data capsules for non-consumptive use of texts," in *Proceedings of the 5th ACM Workshop on Scientific Cloud Computing (ScienceCloud)*, 2014, pp. 9–16.
- [12] M. Mailman, M. Feolo, Y. Jin, M. Kimura, K. Tryka, R. Bagoutdinov, L. Hao, A. Kiang, J. Paschall, L. Phan, N. Popova, S. Pretel, L. Ziyabari, M. Lee, Y. Shao, Z. Wang, K. Sirotnik, M. Ward, M. Kholodov, K. Zbicz, J. Beck, M. Kimmel, S. Shevelev, D. Preuss, E. Yaschenko, A. Graeff, J. Ostell, and S. Sherry, "The NCBI dbGaP database of genotypes and phenotypes," *Nature Genetics*, vol. 39, no. 10, pp. 1181–1186, 2007.
- [13] "National Climatic Data Center (NCDC)," <http://www.ncdc.noaa.gov/>, web site. Accessed: May, 2016.
- [14] "SIMBAD Astronomical Database," <http://simbad.u-strasbg.fr/simbad/>, web site. Accessed: May, 2016.
- [15] N. Wilkins-Diehr, "Special issue: Science gateways common community interfaces to grid resources," *Concurrency and Computation: Practice and Experience*, vol. 19, no. 6, pp. 743–749, 2007.
- [16] Y. Liu, A. Padmanabhan, and S. Wang, "CyberGIS gateway for enabling data-rich geospatial research and education," in *Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER)*, Sept 2013, pp. 1–3.
- [17] D. Stanzione, "The iPlant collaborative: Cyberinfrastructure to feed the world," *Computer*, vol. 44, no. 11, pp. 44–52, Nov 2011.
- [18] W. Wu, H. Zhang, Z. Li, and Y. Mao, "Creating a cloud-based life science gateway," in *Proceedings of the 7th IEEE International Conference on e-Science*, Dec 2011, pp. 55–61.
- [19] R. Madduri, K. Chard, R. Chard, L. Lacinski, A. Rodriguez, D. Sulakhe, D. Kelly, U. Dave, and I. Foster, "The Globus Galaxies platform: delivering science gateways as a service," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 16, pp. 4344–4360, 2015.
- [20] E. Afgan, D. Baker, N. Coraor, H. Goto, I. M. Paul, K. D. Makova, A. Nekrutenko, and J. Taylor, "Harnessing cloud computing with galaxy cloud," *Nature Biotechnology*, vol. 29, pp. 972–974, 2011.
- [21] "StarCluster," <http://star.mit.edu/cluster/>, web site. Accessed: May, 2016.
- [22] M. Albrecht, P. Donnelly, P. Bui, and D. Thain, "Makeflow: A portable abstraction for data intensive computing on clusters, clouds, and grids," in *Proceedings of the 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies*. ACM, 2012, pp. 1:1–1:13.

Reproducibility in Computer Vision: Towards Open Publication of Image Analysis Experiments as Semantic Workflows

Ricky J. Sethi
 Fitchburg State University
 Fitchburg, MA 01420
 rickys@sethi.org

Yolanda Gil
 USC Information Sciences Institute
 Marina del Rey, CA 90292
 gil@isi.edu

Abstract—Reproducibility of research is an area of growing concern in computer vision. Scientific workflows provide a structured methodology for standardized replication and testing of state-of-the-art models, open publication of datasets and software together, and ease of analysis by re-using pre-existing components. In this paper, we present initial work in developing a framework that will allow reuse and extension of many computer vision methods, as well as allowing easy reproducibility of analytical results, by publishing datasets and workflows packaged together as linked data. Our approach uses the WINGS semantic workflow system which validates semantic constraints of the computer vision algorithms, making it easy for non-experts to correctly apply state-of-the-art image processing methods to their data. We show the ease of use of semantic workflows for reproducibility in computer vision by both utilizing pre-developed workflow fragments and developing novel computer vision workflow fragments for a video activity recognition task, analysis of multimedia web content, and the analysis of artistic style in paintings using convolutional neural networks.

I. INTRODUCTION

The limited reproducibility of scientific research is a growing cause for concern, sometimes described as a “reproducibility crisis” [1], [2], [3], [4]. Lack of reproducibility may lead to unreliable comparisons to other work, insufficient verification of the state-of-the-art, inadvertent errors in publications, and a general inability to efficiently build upon previous work.

The importance of reproducible computational research has come to the forefront in computer vision, as evidenced by premier conferences like Computer Vision and Pattern Recognition (CVPR) requiring reviewers to comment on the reproducibility of papers¹, and the International Conference on Image Processing (ICIP) having round tables on reproducibility [5]. In the closely related field of machine learning, prominent members have argued for the need to share software and datasets to facilitate experimentation and learning [6]. In order to combat the growing problems with reproducibility, some researchers have suggested computational solutions for specific areas [7], [8], [9]; but these are often ad hoc programs written in specific languages that do not accommodate

well applications which combine pre-existing, heterogeneous software such as computer vision.

An elegant solution to the problem of creating standardized, reproducible research is to utilize a scientific workflow system [10]. Sharing workflows in addition to data and software is part of the recently proposed Reproducibility Enhancement Principles (REP) [11]. Scientific workflows have been used for reproducible and reusable research in many science fields but have not been previously applied to computer vision. Computer vision presents a unique challenge due to its inherently multi-disciplinary methodologies, widely heterogenous codebases, and dearth of pre-existing foundational computer vision workflows.

This paper describes a collection of workflows for computer vision applications, which we have developed in the WINGS semantic workflow system [12], [13]. The paper illustrates the use of these workflows for reproducible computer vision research in three case studies: activity recognition in video, multimedia analysis of web site content, and analysis of artistic style in paintings. The paper also discusses how fragments of the workflows can be reused for new tasks, not only to save effort but to build on well-tested and proven analysis methods.

In addition to reproducibility and efficiency, our goal is to enable non-experts to reuse these workflows. In particular, we are interested in allowing geoscientists to analyze images collected in the field [14], art scholars to do computational analysis of artistic style [15], and allowing students with limited or no programming background to learn the capabilities offered by image analysis [16].

We begin with an overview of the workflows that we have created. We then describe the use of these workflows in the three case studies to demonstrate reproducibility and reuse. We conclude with a summary of the benefits of workflows to support reproducible research.

II. SEMANTIC WORKFLOWS IN WINGS FOR COMPUTER VISION TASKS

Computational workflows capture an end-to-end analysis composed of individual analytic steps as a dependency graph that indicates dataflow links as well as control flow links

¹<http://tab.computer.org/pamtc/archive/cvpr2010/submission/>

among steps. They represent complex applications as a dependency network of individual computations linked through control or data flow. Each workflow step is a *software component* that can be implemented in a different language from others, each with one or more data inputs and one or more data outputs.

A *workflow fragment* consists of one or more components, together with their data inputs and outputs. A *full workflow* combines multiple workflow fragments connected together by having one or more data outputs of one fragment becoming data inputs of another fragment. An example is shown in Figure 1, where several fragments have been combined to create a full workflow to analyze the movement over time of groups of objects in a video.

Several workflow systems have been developed with a variety of capabilities, and used in many areas of science [10]. However, none have been applied to computer vision. In our work, we use the WINGS semantic workflow system as it has three key features to support reproducibility: semantic constraints to validate a workflow for data [13], abstractions to represent classes of components [12], and the ability to publish workflows as web objects following linked open data principles [17]. A detailed discussion on how WINGS supports reproducibility and validation with examples from clinical omics can be found in [18].

An overview of the collection of workflow fragments for computer vision tasks we have created is shown in Table I. Some of the fragments were built by us, but most are built using existing open source packages that are popular in computer vision. We have fragments built with OpenCV for image filtering, segmentation, and edge detection. We also have fragments for Latent Dirichlet Allocation (LDA), some of them implemented in the Mallet software. We also include fragments that build on the recently released Google TensorFlow machine learning framework for deep learning. Such pre-defined workflow fragments make complex analytics expertise readily available to new users, who can compose them to create new full workflows. The components that make up workflow fragments can be written in heterogeneous languages: e.g., some components are in Java, others in MATLAB, and still others in C++, but the language of choice is irrelevant as the components are integrated into the workflows without reliance upon their individual implementation idiosyncrasies. This is possible because each individual program is converted into a workflow component via a short wrapper shell script (usually 3-5 lines of code).

WINGS adds semantic constraints to the workflows to express the requirements of the different workflow components. Users can export these workflows and make them available as part of a workflow library like the WINGS standard repository so that other researchers can directly utilize any single component (or the entire component collection) in their own workflows by simply importing those web objects. The exported workflows can also be adapted by adding or changing any component. Thus, these *workflows web objects* allow full reproducibility of the original experiments by examining or

executing them online or downloading and importing them locally.

We show both the re-use of existing workflow fragments from the WINGS standard repository as well as the development of novel computer vision workflow fragments in the analysis of three fundamental computer vision tasks: a video activity recognition application, an analysis of multimedia web site content, and the analysis of artistic style using convolutional neural networks. Workflows for all these tasks are covered in detail in the following sections.

III. CASE STUDY: VIDEO ACTIVITY RECOGNITION TASK

For the video activity recognition task, we utilize the the *Group Transition Ratio*, G_{tr} , from [19] to quantitatively define a group as well as the Atomic Group Actions. The G_{tr} is defined as $G_{tr} = \frac{L}{\lambda}$, where λ is the mean free path and L is the characteristic length. The characteristic length is usually a convenient reference length that is a constant of a given configuration.

We use the G_{tr} to identify when a collection of objects can be considered as separate individuals, a group, or a crowd. In addition, we use the time variance of the G_{tr} to determine *when* a collection of objects transitions between being individuals, groups, or crowds. We implemented the G_{tr} in MATLAB as a component in the WINGS workflow system and tested it against the Atomic Group Actions dataset [20]. The workflow is shown in Figure 1. The Evaluation component represented there is a composite representation of multiple statistical evaluation components from Table I. The re-use of pre-created workflow fragments like the statistical fragments shown there allowed for a rapid development cycle in addition to providing the ability to export the entire analysis and dataset as web objects for reproducibility.

We first show some qualitative results for the G_{tr} analysis shown in Figure 1 that were generated automatically from the workflow. In Figure (2), we show how the G_{tr} varies with time to characterize the action in a video; results for all categories of the G_{tr} graph along with representative frames from the

TABLE I
WORKFLOW FRAGMENTS CREATED FOR COMPUTER VISION TASKS.

Category	Workflow Fragments
Computer Vision	OpenCV components (Optical Flow, Kalman tracker, Mixture of Gaussians, Particle filter, etc.), N-Cuts, PhaseSpace, G_{tr} , <i>RelativeVelocity</i> , <i>RelativeDistance</i> , Image Extractor, Background/Foreground Extraction, Neural Algorithm for Artistic Style (Lua/Torch), Neural Algorithm for Artistic Style (TensorFlow)
General Machine Learning	K-Means, Latent Dirichlet Allocation, Mallet, libSVM, Caffe, Convolutional Neural Networks (Lua/Torch), TensorFlow, Adam Optimizer, Recurrent Neural Networks
Statistical Evaluation	Confusion Matrices, Heatmaps, Precision-Recall Curves, ROC Curves, AUC Curves, Equal Error Rate, F-Measure

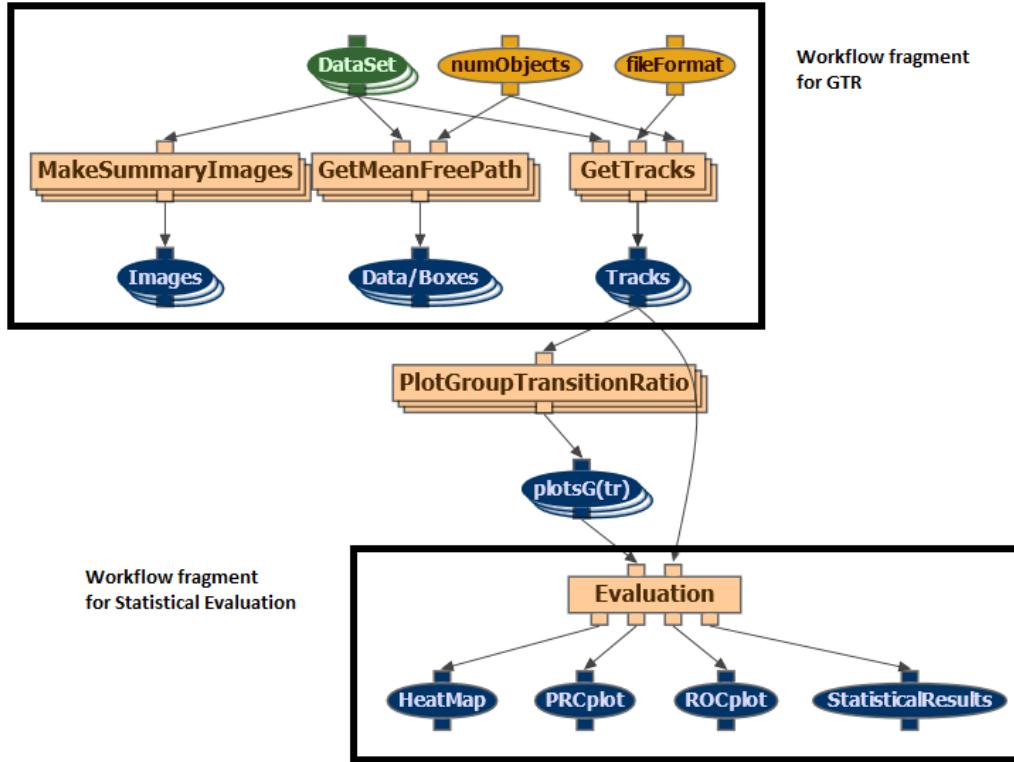


Fig. 1. Workflow for the Group Transition Ratio (G_{tr}) Model for the Video Activity Recognition Task.

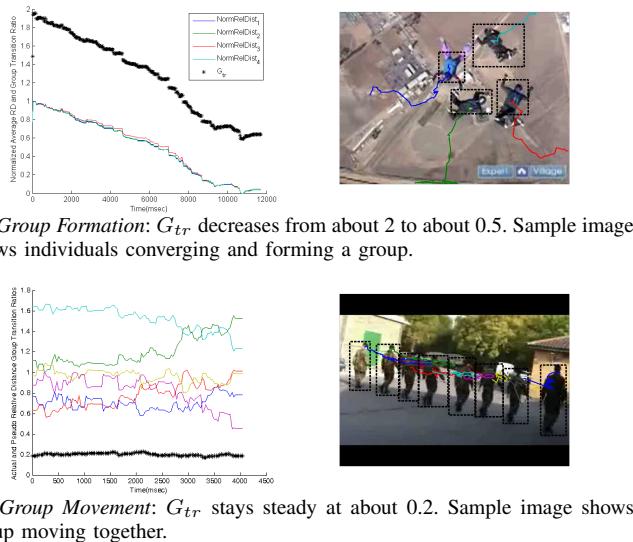


Fig. 2. Workflow Generated Results for the Video Activity Recognition Task.

video are overlaid to show the action. The G_{tr} decreases with time for *group formation* in (a), increases with time for *group dispersal*, and stays the same for *group movement* in (b). The thresholds were empirically determined automatically by the workflow to be $G_{tr} \lesssim 0.1$ for crowds, $0.1 \lesssim G_{tr} \lesssim 1.5$ for

groups, and $G_{tr} \gtrsim 1.5$ for individuals. More details about this task are described in [20].

IV. CASE STUDY: MULTIMEDIA ANALYSIS OF WEB CONTENT

We demonstrate the utility of image analysis workflow fragments by combining them with pre-existing text analysis workflow fragments in order to analyze multimedia content of web sites that contain ads for services in order to detect human trafficking. This project analyzes posts on various sites on the Internet in order to determine if the subject of that post may be a victim of human trafficking and alert law enforcement. Using both the text and image content of posts helps to make a stronger determination of whether or not the subject of the post was trafficked. The combined workflow is described in [21], we focus here on the image analysis aspects.

The initial development of the project had progressed to creating a crawler, which downloads posts from various posting sites, and an extractor, which extracts the text and images and stores them in a database. However, there had been no substantial analysis of the posts in this nascent project. We extended the project to examine both the text of the post (using the text analytics workflow fragments we had already developed, described in [22]), as well as the associated images (using the image analysis workflow fragments we developed); a final determination about trafficking of the subject of the post was made by fusing the results of the text and image analyses

via a fusion workflow fragment we developed. Thus, the re-use and re-purposing of workflow fragments allowed a multimedia analysis spanning data domains of text and image analysis, including the fusion of their results in the final determination.

The fragments of the workflow that focus on computer vision are shown in Figure 3. The initial image analysis is done via N-cuts followed by an unsupervised Mallet LDA analysis and a supervised analysis using SVM in a bag-of-words model.

The original development of the HTD crawler/extractor had taken several months; the conversion to WINGS took roughly two days. The extension of the other components took approximately one day, saving effort estimated to be on the order of 300 man-hours of work. Summary results showed an Equal Error Rate of 0.37 and F-Measure of 0.47 for the fusion module that combines image and text analysis results. More details about this workflow can be found in [21].

V. CASE STUDY: ANALYSIS OF ARTISTIC STYLE USING CONVOLUTIONAL NEURAL NETWORKS

In this section, we show the use of workflows for image analysis and processing of paintings. We implemented the newly released and very popular neural algorithm for artistic style developed in [23]. In order to allow maximal flexibility in the use of the neural algorithm of artistic style [23], we developed two separate implementations as shown in the two workflow fragments shown in Figure 4, which uses the Lua scripting language with the Torch machine learning framework, and Figure 5, which uses Python and Google’s TensorFlow machine learning framework. Both of these use fragments for pre-processing that were also developed in Table I.

The Neural Algorithm of Artistic Style uses deep neural networks (specifically, a Convolutional Neural Network, CNN) to separate the style and content of an image. It designates one image as a style image and one as target image. It then extracts the style from the style image and applies it to the content of the target image to create a new image in the style of the style image.

For example, Figure 6 shows the target image of a scene from Tubingen as presented in the original paper [23]. The algorithm then extracts the style from the Starry Night painting of Van Gogh. This style is applied to the Tubingen image to create a new image of Tubingen in the style of Van Gogh. Similarly, they extract the style of Munch using his painting, The Scream, and apply this to the Tubingen scene, as well.

We reproduce these results in Figure 6 using the workflow fragments shown in Figure 4.

CNNs pass filters of shared weights across overlapping image patches to learn convolutions, also called feature maps. Following the example of the paper, we utilize the publicly available and trained VGG network. The method for determining the style is based on texture extractions using correlations between feature-maps within a layer. The final reconstructed image is created by first randomly generating a white-noise image and then using gradient descent for the

optimization. In the workflows in Figures 4 and 5, we use Adam Optimization, an algorithm for first-order gradient-based optimization. Finally, the loss function in the article is a weighted sum of the style and content mean squared error loss functions and is used to generate the final new image. We can re-use the Adam Optimization module in the workflow in Figure 5 if we did not want to use the optimization provided in TensorFlow.

VI. CONCLUSION

The inability to reproduce results of computational algorithms is a significant issue in science. This paper describes a collection of workflow fragments for computer vision and their use in four different image analysis tasks. By using the power of semantic workflows, we provide a framework to export workflows and their associated datasets as web objects, thus allowing for easy and full reproducibility of research. Using such a workflow framework allows for quick deployment and comparison of various approaches and algorithms. It also eases development by incorporating heterogeneous codebases, re-using components and providing standard implementations of popular components, and allowing for easy extension of pre-existing workflows. We are already using some of these workflows to teach data science to non-programmers [16]. In the future, we plan to use these workflows to allow non-experts to apply state-of-the-art computer vision methods. This includes allowing geoscientists to analyze images collected in the field [14] and art scholars to do computational analysis of artistic style [15].

ACKNOWLEDGMENTS

This research was supported in part by the US National Science Foundation (NSF) with grants ICER-1440323 and ICER-1632211, in part under grant CNS-1019343 to the Computing Research Association for the CIFellows Project, and in part under the National Endowment for the Humanities (NEH) Grant under Award HD-248360-16. We would like to thank Taylor Alarcon, Catherine A. Buell, Alyssa Deng, Hyunjoo Jo, Andrew Philpot, and William P. Seeley for their discussions and their assistance in developing some of the workflow fragments.

REFERENCES

- [1] P. Vandewalle, J. Kovacevic, and M. Vetterli, “Reproducible research in signal processing,” *IEEE Signal Processing Magazine*, vol. 26, no. 3, pp. 37–47, May 2009.
- [2] D. L. Donoho, A. Maleki, I. U. Rahman, M. Shahram, and V. Stodden, “Reproducible Research in Computational Harmonic Analysis,” *Computing in Science & Engineering*, vol. 11, no. 1, pp. 8–18, Jan. 2009.
- [3] S. Fomel and J. F. Claerbout, “Reproducible Research,” *Computing in Science & Engineering*, vol. 11, no. 1, pp. 5–7, 2009.
- [4] R. De Veaux and et al., “Curriculum guidelines for undergraduate programs in data science,” *Annual Review of Statistics, forthcoming*, 2016.
- [5] “ICIP Reproducibility Round Table,” in <http://www.icip2011.org/index.php/Main/RoundTable>, 2011.
- [6] S. Sonnenburg, M. L. Braun, and et al., “The need for open source software in machine learning,” 2007.

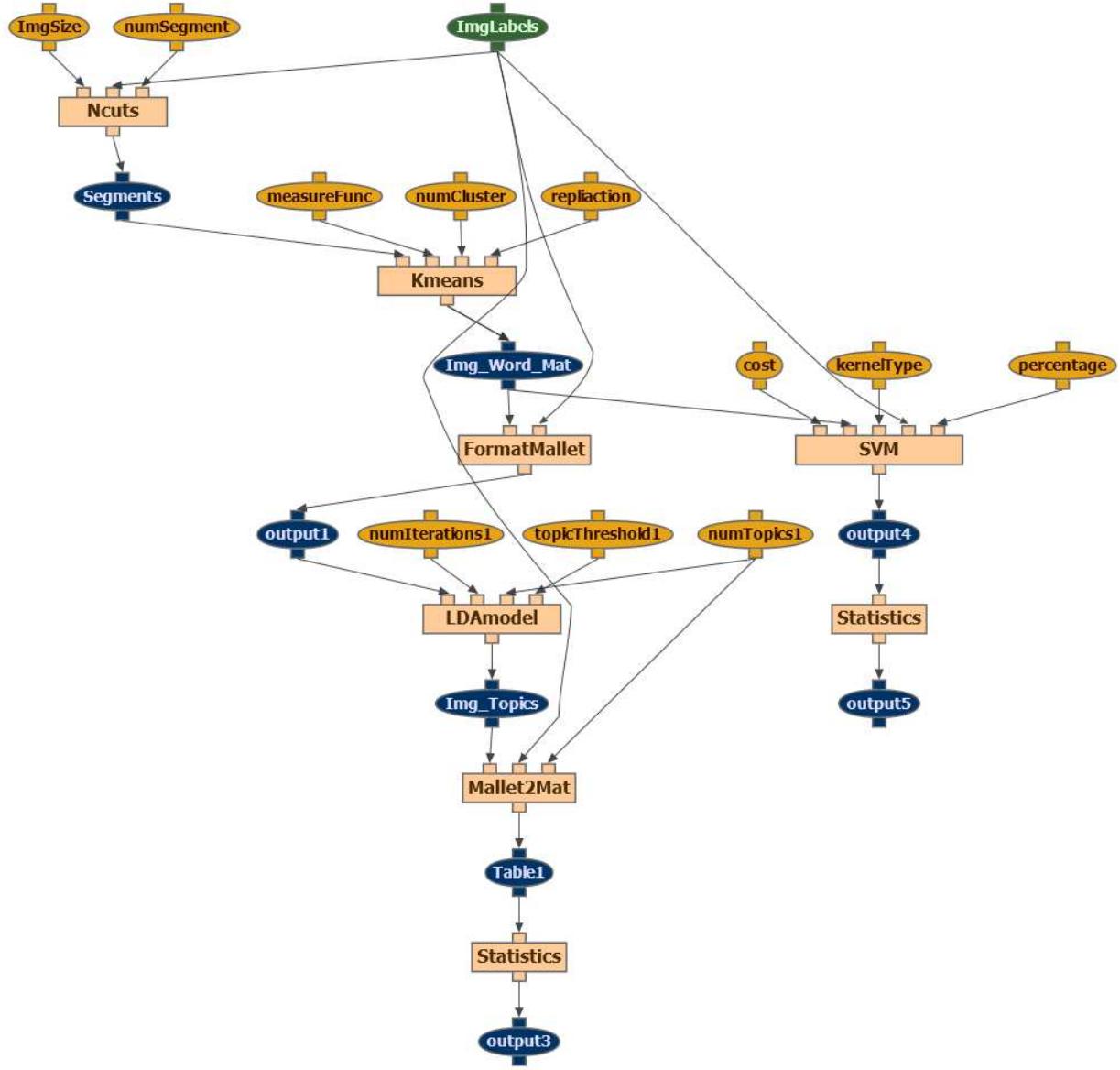


Fig. 3. Workflow for the Multimedia Web Content Analysis Task.

- [7] R. D. Peng and S. P. Eckel, "Distributed Reproducible Research Using Cached Computations," *Computing in Science & Engineering*, vol. 11, no. 1, 2009.
- [8] R. LeVeque, "Python tools for reproducible research on hyperbolic problems," *Computing in Science & Engineering*, pp. 19–27, 2009.
- [9] J. Buckheit, "Wavelab and reproducible research," *LECTURE NOTES IN STATISTICS*, 1995.
- [10] I. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields, *Workflows for e-Science: Scientific Workflows for Grids*, I. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields, Eds. Springer-Verlag, 2006.
- [11] V. Stodden, M. McNutt, D. H. Bailey, E. Deelman, Y. Gil, B. Hanson, M. A. Heroux, J. P. Ioannidis, and M. Taufer, "The reproducibility enhancement principles for disclosing computational methods," *Science*, forthcoming, 2016.
- [12] Y. Gil, V. Ratnakar, J. Kim, P. A. Gonzalez-Calero, P. Groth, J. Moody, and E. Deelman, "Wings: Intelligent workflow-based design of computational experiments," *IEEE Intelligent Systems*, vol. 26, no. 1, 2011.
- [13] Y. Gil, P. A. Gonzalez-Calero, J. Kim, J. Moody, and V. Ratnakar, "A semantic framework for automatic generation of computational workflows using distributed data and component catalogs," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 23, no. 4, 2011.
- [14] Y. Gil and S. Pierce, "Final report of the 2015 nsf workshop on intelligent systems for geosciences," 2015, national Science Foundation, Arlington, VA.
- [15] R. J. Sethi, C. A. Buell, W. P. Seeley, and Y. Gil, "Democratizing the visual stylometry of art: Analysis of artistic style through computational workflows," *Communications of the ACM*, forthcoming, 2016.
- [16] Y. Gil, "Teaching big data analytics skills with intelligent workflow systems," in *Proceedings of the Sixth Symposium on Educational Advances in Artificial Intelligence (EAAI), colocated with the National Conference of the Association for the Advancement of Artificial Intelligence (AAAI)*, Phoenix, AZ, 2016.
- [17] D. Garjio, Y. Gil, and O. Corcho, "Abstracting, linking, publishing, exploiting: Workflow sharing and reuse as linked open data," *Future Generation Computer Systems*, forthcoming, 2016.
- [18] C. L. Zheng, V. Ratnakar, Y. Gil, and S. K. McWeeney, "Use of semantic workflows to enhance transparency and reproducibility in

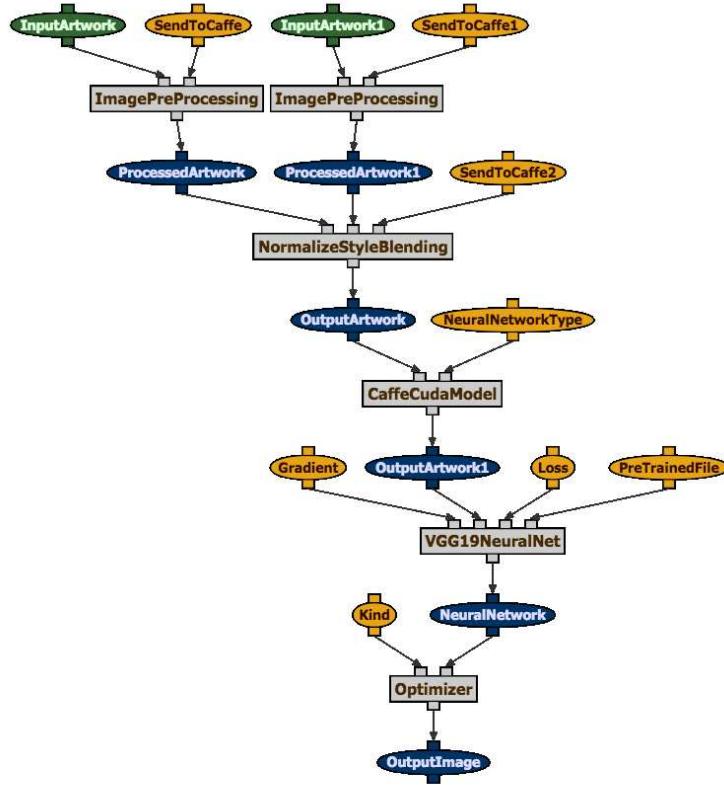


Fig. 4. Workflow for the Neural Algorithm of Artistic Style using Lua and Torch.

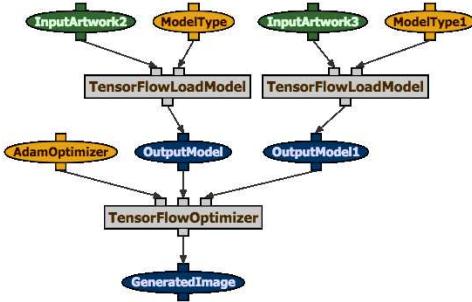


Fig. 5. Workflow for the Neural Algorithm of Artistic Style using Python and Google's TensorFlow.



- clinical omics,” *Genome Medicine*, vol. 7, no. 73, 2015.
- [19] B. Song, R. J. Sethi, and A. K. Roy-Chowdhury, “Robust Wide Area Tracking in Single and Multiple Views,” in *Visual Analysis of Humans*, T. B. Moeslund, L. Sigal, V. Krüger, and A. Hilton, Eds. Springer-Verlag, 2011, pp. 1–18.
 - [20] R. J. Sethi, “Towards defining groups and crowds in video using the atomic group actions dataset,” in *IEEE International Conference on Image Processing (ICIP)*, 2015.
 - [21] R. J. Sethi, Y. Gil, H. Jo, and A. Philpot, “Large-scale multimedia content analysis using scientific workflows,” in *ACM International Conference on Multimedia (ACM MM)*, 2013.

Fig. 6. Reproducing the results from [23] using the workflows in Figure 4.

- [22] M. Haider, Y. Gil, and Y. Liu, “A framework for efficient text analytics through automatic configuration and customization of scientific workflows,” in *In Proceedings of the Seventh IEEE International Conference on e-Science*, Stockholm, Sweden, 2011.
- [23] L. A. Gatys, A. S. Ecker, and M. Bethge, “A neural algorithm of artistic style,” Aug 2015.

Crossing Analytics Systems: A Case for Integrated Provenance in Data Lakes

Isuru Suriarachchi and Beth Plale

School of Informatics and Computing, Indiana University

Bloomington, IN, USA

Email: {isuriara,plale}@iu.edu

Abstract—The volumes of data in Big Data, their variety and unstructured nature, have had researchers looking beyond the data warehouse. The data warehouse, among other features, requires mapping data to a schema upon ingest, an approach seen as inflexible for the massive variety of Big Data. The Data Lake is emerging as an alternate solution for storing data of widely divergent types and scales. Designed for high flexibility, the Data Lake follows a schema-on-read philosophy and data transformations are assumed to be performed within the Data Lake. During its lifecycle in a Data Lake, a data product may undergo numerous transformations performed by any number of Big Data processing engines leading to questions of traceability. In this paper we argue that provenance contributes to easier data management and traceability within a Data Lake infrastructure. We discuss the challenges in provenance integration in a Data Lake and propose a reference architecture to overcome the challenges. We evaluate our architecture through a prototype implementation built using our distributed provenance collection tools.

I. INTRODUCTION

Industry, academia, and research alike are grappling with the opportunities that Big Data brings in mining data from numerous sources for insight, decision making, and predictive forecasts. These sources (*e.g.*, clickstream, sensor data, IoT devices, social media, server logs) are frequently both external to an organization and internal. Data from sources such as social media and sensors is generated continuously. Depending on the source, data can be structured, semi-structured, or unstructured. The traditional solution, the data warehouse, is proving inflexible and limited [1] as a data management framework in support of multiple analytics platforms and data of numerous sources and types.

The response to the limits of the data warehouse is the Data Lake [2], [1]. A feature of the Data Lake is its schema-on-read (as opposed to schema-on-write which happens at ingest time) where commitments to a particular schema are deferred to time of use. Schema-on-read suggests that data are ingested in a raw form, then converted to a particular schema as needed to carry out analysis. The Data Lake paradigm acknowledges that high throughput analytics platforms in use today are varied, so has to support multiple Big Data processing frameworks like Apache Hadoop¹, Apache Spark² and Apache Storm³. The vision of the Data Lake is one of data from numerous

sources being dropped into the lake quickly and easily, with tools around the lake as designated fishers of the lake, intent on catching insight by the rich ecosystem of data within the Data Lake.

This greater flexibility of the Data Lake leads to rich collections of data from various sources. It, however, leaves greater manageability burdens in the hands of the lake administrators. The Data Lake can easily become a “dump everything” place due to absence of any enforced schema, sometimes referred to in literature as a “data swamp” [3]. The Data Lake could easily ignore the fact that data items in a Data Lake can exist in different stages of their life cycle. One data item may be in raw stage after recent generation where another may be the fined result of analysis by one or more of the analysis tools. The complications of data life cycle increases the need for proper traceability mechanisms. In this paper the critical focus of our attention is on metadata and lineage information through a data life cycle which are key to good data accessibility and traceability [4].

Data provenance is the information about the activities, entities and people who involved in producing a data product. Data provenance is often represented in one of two standard provenance representations (*i.e.*, OPM [5] or PROV [6]). Data provenance can help with management of a Data Lake by making it clear where an object is in its lifecycle. This information can ease the burden of transformation needed by analysis tools to operate on a dataset. For instance, how does a researcher know what datasets are available in the lake for Apache Spark analysis, and which can be available through a small amount of transformation? This information can be derived by hand by the lake administrator and stored to a registry, but that approach runs counter to the ease with which new data can be added to the data lake. Another issue with the Data Lake is trust. Suppose a Data Lake is set up to organize data for the watershed basin of the Lower Mekong River in southeast asia. The contributors are going to be from numerous countries through which the Mekong River passes. How does the Data Lake ensure that the uses of the data in the lake are proper and adhere to the terms of contribution? How does a researcher, who uses the Data Lake for their research, prove that their research is done in a way that is consistent with the terms of contribution?

Provenance contributes to these questions of use and trust. If the Data Lake framework can ensure that every data product’s

¹<http://hadoop.apache.org/>

²<http://spark.apache.org/>

³<http://storm.apache.org/>

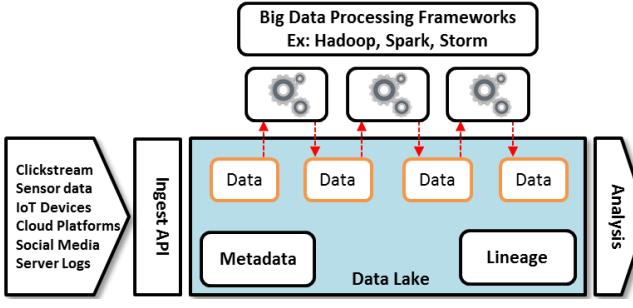


Fig. 1. Data Lake Architecture

lineage and attribution are in place starting from the origin, critical traceability can be had. However, that is a challenging task because a data product in a Data Lake may go through different analytics systems such as Hadoop, Spark and Storm which do not produce provenance information by default. Even if there are provenance collection techniques for those systems, they may use their own ways of storing provenance or use different standards. Therefore generating integrated provenance traces across systems is difficult.

In this paper we propose a reference architecture for provenance in a Data Lake based on a central provenance subsystem that stores and processes provenance events pumped into it from all connected systems. The reference architecture, which appeared in early work as a poster [7], is deepened here. A prototype implementation of the architecture using our distributed provenance collection tools shows that the proposed technique can be introduced into a Data Lake to capture integrated provenance without introducing much overhead.

The paper's three main contributions are: identification of the data management and traceability problems in a Data Lake that are solvable using provenance highlighting the challenges in capturing integrated provenance. Second, a reference architecture to overcome those challenges. Third, an evaluation of the viability of the proposed architecture using a prototype implementation with techniques that can be used to reduce the overhead of provenance capture.

II. DATA LAKE ARCHITECTURE

The general architecture of a Data Lake, shown in Figure 1, contains three main activities: (1) Data ingest, (2) Data processing or transformation and (3) Data analysis. A Data Lake may open up number of ingest APIs to bring data from different sources into the lake. In most cases, raw data is ingested into the Data Lake for later use by researchers for multiple purposes at different points of time. Activity in the Data Lake can be viewed as data transformation: where data in the Data Lake is input to some task, and output is stored back to the Data Lake. Modern large scale distributed Big Data processing frameworks like Hadoop, Spark and Storm are the source of such transformations, especially for Data Lakes implemented on HDFS. Mechanisms like scientific workflow systems such as Kepler [8] and legacy scripts may apply as well. As shown in Figure 1, a data product created as an output from one transformation can be an input into another transformation which itself may produce another one as a

result. Finally when all processing steps are done, resulting data products are used for different kinds of analysis reports and predictions.

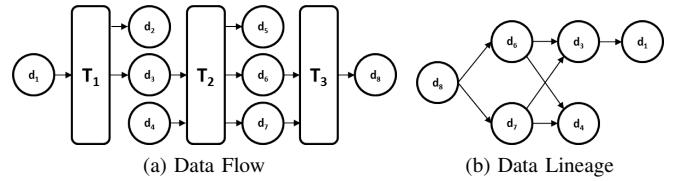


Fig. 2. Data Lineage in a Data Lake

III. PROVENANCE IN DATA LAKE

A. Role of Provenance

The Data Lake achieves increased flexibility at the cost of reduced manageability. In the research data environment, when differently structured data is ingested by different organizations through one of multiple APIs, tracking becomes an issue. Chained transformations that continuously derive new data from existing data in the lake further complicate the management. How can minimal management be added to the lake without invalidating the attractive benefit of ease of ingest? We posit that this minimal management is in the form of mechanisms to track origins of the data products, rights of use and suitability of transformations applied to them, and quality of data generated by the transformations. Carefully captured provenance can satisfy these needs allowing, for instance, answers to the following two questions: 1.) Suppose sensitive data are deposited into a Data Lake; social science survey data for instance. Can data provenance prevent improper leakage into derived data? 2.) Repeating a Big Data transformation in a Data Lake is expensive due to high resource and time consumption. Can live streaming provenance from experiments identify problems early in their execution?

B. Challenges in Provenance Capture

Data in a Data Lake may go through number of transformations performed using different frameworks selected according to the type of data and application. For example, in a HDFS based Data Lake, it is common to use Storm or Spark Streaming for streaming data and Hadoop MapReduce or Spark for batch data. Other legacy systems and scripts may be included as well. To achieve traceability across transformations, provenance captured from these systems must be integrated, a challenge since many do not support provenance by default.

Techniques exist to collect provenance from Big Data processing frameworks like Hadoop and Spark [9], [10], [11], [12]. But most are coupled to a particular framework. If the provenance collection within a Data Lake depends on such system specific methods, provenance from all subsystems should be stitched together to create a deeper provenance trace. There are stitching techniques [13], [14] which bring all provenance traces into a common model and then integrate them together. However the process of converting provenance

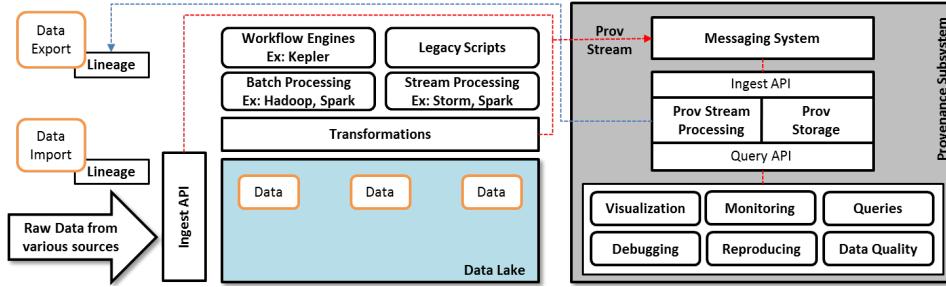


Fig. 3. Provenance for Data Lakes: Reference Architecture

traces from different standards into a common model may lose provenance information depending on the data model followed by each standard. As a Data Lake deals with Big Data, most transformations generate large provenance graphs. Converting such large provenance graphs into a common model and stitching them together can introduce considerable compute overheads as well.

C. Provenance Integration Across Systems

To address provenance integration, we propose a central provenance collection system to which all components within the Data Lake stream provenance events. Well accepted provenance standards like W3C PROV [6] and OPM [5] represent provenance as a directed acyclic graph ($G = (V, E)$). A node ($v \in V$) can be an activity, entity or agent while an edge ($e = \langle v_i, v_j \rangle$ where $e \in E$ and $v_i, v_j \in V$) represents a relationship between two nodes. In our provenance collection model, a provenance event always represents an edge in the provenance graph. For example, if process p generates the data product d , the provenance event adds a new edge ($e = \langle p, d \rangle$ where $p, d \in V$) into the provenance graph to represent the ‘generation’ relationship between activity p and entity d .

In addition to capturing usage and generation, additional details like configuration parameters and environment information (e.g., CPU speed, memory capacity, network bandwidth) can be stored as attributes connected to the transformation. Inside each transformation, there can be number of intermediate tasks which may themselves generate intermediate data products. A MapReduce job for instance has multiple map and reduce tasks. Capturing provenance from such internal tasks at a high enough level to be useful helps in debugging and reproducing transformations.

When the output data from one analysis tool is used as the input to another, integration of provenance collected from both transformations can be guaranteed only by a consistent lake-unique persistent ID policy [6]. This may require a global policy enforced for all contributing organizations to a Data Lake. This unique ID notion could be based on file URLs and randomly generated data identifiers which are appended to data records when producing outputs so that the following transformations can use the same identifiers. It could also be achieved using globally persistent IDs such as the Handle system or DOIs. As a simple example, consider Figure 2a. The data product d_1 is subject to transformation T_1 and generates d_2 and d_3 as results. T_2 uses d_3 together with a new data

product d_4 and generates d_5 , d_6 and d_7 . Finally T_3 uses d_6 and d_7 and generates d_8 as the final output. When all three transformations T_1 , T_2 and T_3 have sent provenance events, a complete provenance graph is created in the central provenance collection system. Figure 2b shows the high level data lineage graph which represents the data flow starting from d_8 .

D. Reference Architecture

The reference architecture, shown in Figure 3, uses a central provenance collection system. Provenance events captured from components in the Data Lake are streamed into the provenance subsystem where they are processed, stored and analysed. The Provenance Stream Processing and Storage component at the heart of this architecture accepts the stream of provenance notifications (Ingest API) and supports queries (Query API). A live stream processing subsystem supports live queries while storage subsystem persists provenance for long term usage. When long running experiments in the Data Lake produce large volumes of provenance data, stream processing techniques become extremely useful as storing full provenance is not feasible. The Messaging System guarantees reliable provenance event delivery into the central provenance processing layer. Usage subsystem shows how provenance collected around the Data Lake can be used for different purposes. Both live and post-execution queries over collected provenance with Monitoring and Visualization helps in scenarios like the two use cases that we discussed above. There are other advantages as well such as Debugging and Reproducing experiments in the Data Lake.

In order to capture information about the origins of data, provenance must be captured at the Ingest. Some data products may carry their previous provenance information which should be integrated as well. Researchers may export data products from the Data Lake in some situations. Such data products should be coupled with their provenance for better usage.

IV. PROTOTYPE IMPLEMENTATION

We set up a prototype Data Lake and implemented a use case on top of it to evaluate the feasibility of our reference architecture. We used our provenance collection tools to capture, store, query and visualize provenance in our Data Lake. The reference architecture introduces both stored provenance processing and real time provenance processing for Data Lakes. In this prototype, we implement stored provenance

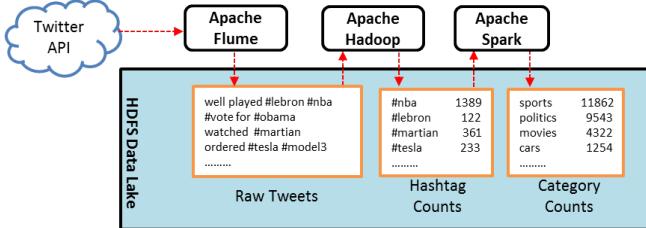


Fig. 4. Data Lake use case

processing; real time provenance processing is future work. The central provenance subsystem uses our Komadu [15] provenance collection framework.

A. Komadu

Komadu is a W3C PROV based provenance collection framework which accepts provenance from distributed components through RabbitMQ⁴ messaging and web services channels. It does not depend on any global knowledge about the system in a distributed setting. This makes it a good match for a Data Lake environment where different systems are used to perform different data transformations. Komadu API can be used to capture provenance events from individual components of the Data Lake. Each ingest operation adds a new relationship (R) between two nodes (a node can be an activity(A), entity(E) or agent(G)) of the provenance graph being generated. For example, when an activity A generates an entity E , the `addActivityEntityRelationship(A , E , R)` operation can be used to add a `wasGeneratedBy` relationship between A and E . Using the query operations, full provenance graphs including all connected edges can be generated for Entities, Activities and Agents by passing the relevant identifier. Backward only and forward only provenance graphs can be generated for Entities. In addition to that, Komadu API consists of operations to access the attributes of all types of nodes. Komadu Cytoscape⁵ plugin can be used to visualize and navigate through provenance graphs.

B. Data Lake Use Case

The Data Lake prototype was implemented using an HDFS cluster and the transformations were performed using Hadoop and Spark. Analysing data from social media to identify trends is commonly seen in Data Lakes. As shown in Figure 4, we have implemented a chain of transformations based on Twitter data to first count hash tags and then to get aggregated counts based on categories. Apache Flume⁶ was used to collect Twitter data and store in HDFS through the Twitter public streaming API. For each tweet, Flume captures the Twitter handle of the author, time, language and the full message and writes a record into an HDFS file. After collecting Twitter data over a period of five days, a Hadoop job was used to count hash tags in the full Twitter dataset. A new HDFS file with hash tag counts is generated as the result of the first Hadoop job which is used by a separate Spark job to

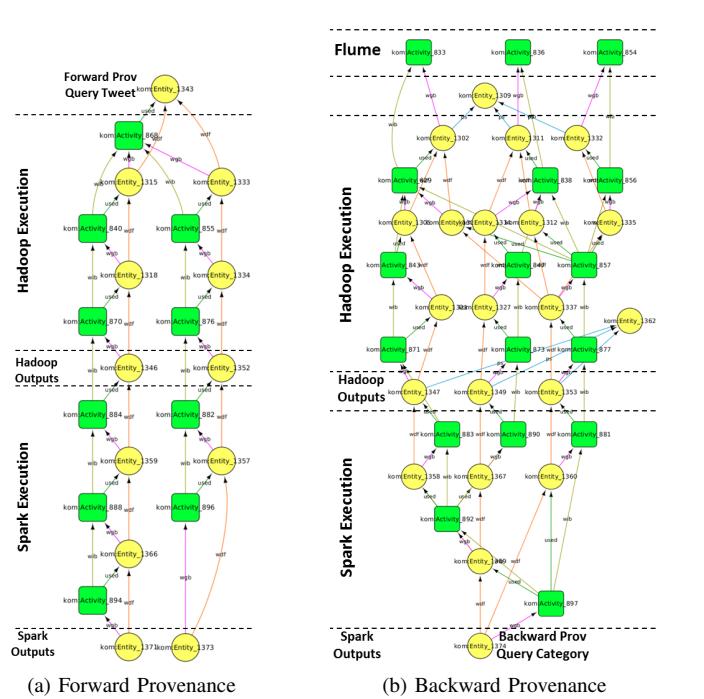


Fig. 5. Forward and Backward Provenance

get aggregated counts according to categories (sports, movies, politics etc). We just used a fixed set of categories for this prototype implementation to make it simple. In real Data Lakes, these transformations can be performed by different scientists at different times. They may use frameworks based on their preference and expertise. That is why we used two different frameworks for the transformations in our prototype to show how provenance can be integrated across systems.

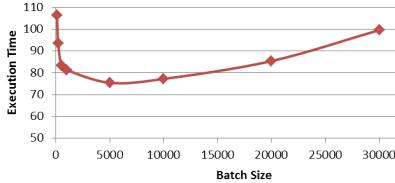
Komadu and its tool kit was used to build the provenance subsystem (shown in Figure 3) in our prototype. Komadu supports RabbitMQ messaging system and includes tools to fetch provenance notifications from RabbitMQ queues. A RabbitMQ instance was deployed in front of our Komadu instance so that all provenance notifications generated by Flume, Hadoop and Spark goes through a message queue in RabbitMQ. Ingested provenance events are asynchronously processed by Komadu and stored in relational tables. Stored provenance remains as a collection of edges until a graph generation request comes in. This delayed graph generation leads to efficient provenance ingest with minimum back pressure. This helps in a Data Lake environment where high volumes of provenance are generated.

To assign consistent identifiers for data items in our Data Lake, we followed the practice of appending identifiers to data records when output data is written to the Data Lake. Subsequent transformation uses the same identifiers for provenance collection. Provenance events were captured in our prototype by instrumenting the application code that we implemented for each transformation. Tweet capturing code in Flume was instrumented to capture provenance at the data ingest into the Data Lake. *Map* and *Reduce* functions in the Hadoop job and *MapToPair* and *ReduceByKey* functions in the Spark job were instrumented to capture provenance from transformations. We

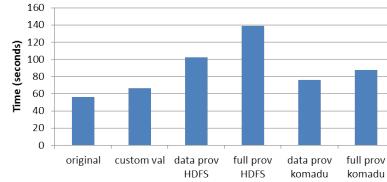
⁴<http://www.rabbitmq.com/>

⁵<http://www.cytoscape.org/>

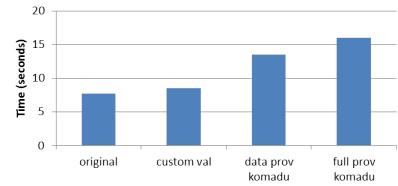
⁶<https://flume.apache.org/>



(a) Execution time with varying batch size



(b) Hadoop Execution times for different scenarios



(c) Spark Execution times for different scenarios

Fig. 6. Experiments based on prototype Data Lake

implemented a client library with a simple API (like Log4J API for logging) which can be used to easily instrument Java applications for provenance capture. It minimizes the provenance capturing overhead by using a dedicated thread pool to asynchronously send provenance events into the provenance subsystem. In addition to that, the client library uses an event batching mechanism to minimize the network overhead by reducing the number of messages sent into the provenance subsystem over the network.

C. Provenance Queries and Visualization

After executing the provenance enabled Hadoop and Spark jobs on collected Twitter data, Komadu query API was used to generate provenance graphs. Komadu generates PROV-XML provenance graphs and it comes with a Cytoscape plugin which can be used to visualize and explore them. Fine-grained provenance includes input and output datasets for each transformation, intermediate function executions and all intermediate data products generated during the execution. Provenance from Flume, Hadoop and Spark have been integrated together through the usage of unique data identifiers.

Figure 5 shows forward and backward provenance graphs generated for a very small subset of tweets. Forward provenance is useful to derive details about the usages of a particular data item. Figure 5a shows a forward provenance graph for a single tweet. It shows the hash tags generated by that particular tweet in Hadoop outputs and the categories to which those hash tags contributed in Spark outputs. A backward provenance graph starting from a category under Spark outputs is shown in Figure 5b. This graph can be used to find all tweets which contributed for that category. For example, if a scientist wanted to get an age distribution of the authors who tweeted about sports, it can be done by finding the set of Twitter handles of the authors through backward provenance.

D. Performance Evaluation

To build our prototype, we used five small VM instances with 2 CPU cores of 2.5GHz speed, 4 GB of RAM and 50 GB local storage on each instance. Four instances were used for the HDFS cluster including one master node and three slave nodes. Total of 3.23 GB Twitter data was collected over a period of five days by running Flume on the master node. Hadoop and Spark clusters were set up on top of our four node HDFS cluster. One separate instance was allocated to set up the provenance subsystem using RabbitMQ and Komadu tools.

In order to minimize the provenance capture overhead, we used a dedicated thread pool and a provenance event batching mechanism in our client library. When the batch size is set to a relatively large number (>500), execution time becomes almost independent of the thread pool size as the number of messages sent through the network reduces. Therefore, we set the client thread pool size to 5 in each of our experiments. Figure 6a shows how the provenance enabled Hadoop execution time for a particular job varies when the batch size is increased from 100 to 30000 (provenance events). As per this result, we set the batch size to 5000 in each of our experiments. We used JSON format to encode provenance events and the average event size is around 120 bytes. The average size of a batched message sent over the network is around 600 KB (5000 x 120 bytes).

Figure 6b shows the execution times of the Hadoop job for different scenarios. Column ‘original’ represents the Hadoop execution time without capturing any provenance. In order to relate Map and Reduce provenance, we had to use a customized value field (in key-value pair) which contains data identifiers like in Ramp [10]. As shown by ‘custom val’ column in the chart, usage of customized value introduces an overhead of 19.28% and that is included in all other cases. Execution overhead depends on the granularity of provenance as well. Columns ‘data prov komadu’ and ‘full prov komadu’ shows the execution times of Hadoop when our technique is used to capture provenance. Data provenance (data relationships only) case adds a 36.47% overhead while full (data and process relationships) provenance case adds a 56.93% overhead. Table I shows a breakdown of provenance sizes generated for each case in Hadoop for the input size of 3.23 GB. Size of provenance doubles for full provenance case compared to data provenance and that leads to greater capturing overheads. As it is a common practice [10] to write provenance into HDFS in Hadoop jobs, we modified the same Hadoop job to store provenance events in HDFS as well and compared the overhead with our method. As shown by ‘data prov HDFS’ and ‘full prov HDFS’ columns in Figure 6b, that adds larger overheads compared to our techniques. Better performance have been achieved by modifying or extending Hadoop [11]. But our techniques operate completely on application level without modifying existing frameworks.

Figure 6c shows the execution times for the Spark job for different scenarios. Like in Hadoop, we used a customized output value to include data identifiers in Spark as well. That adds an overhead of 7.5% compared to original execution

TABLE I
SIZE (IN GB) OF PROVENANCE GENERATED BY HADOOP

	Map	Combine	Reduce	Total
Data Provenance	3.232	1.281	0.529	5.042
Full Provenance	9.733	1.824	0.813	12.37

time as shown by ‘custom val’ column. Data provenance and full provenance cases using Komadu add overheads of 76.1% and 108.35% respectively. Overhead percentages added by provenance capture in Spark is larger compared to Hadoop as Spark works faster than Hadoop and our techniques introduce same level of overhead in both cases.

V. RELATED WORK

Apache Falcon⁷ manages the data lifecycle in Hadoop Big Data stack. Falcon supports creating lineage enabled data processing pipelines by connecting Hadoop based processing systems. Apache Nifi⁸ is another data flow tool which captures lineage while moving data among systems. Neither tool captures detailed provenance within transformation steps (like in Figure 5). Few recent studies target provenance in individual Big Data processing frameworks like Hadoop and Spark. Wang J. et al. [9], [16] present a way of capturing provenance in MapReduce workflows by integrating Hadoop into Kepler. Ramp [10] and HadoopProv [11] are two attempts to capture provenance by extending Hadoop. Provenance in Apache Spark [12] and provenance in streaming data [17] have also been studied. Capturing provenance in traditional scientific workflows [18], [19] is another area which has been studied in depth. None of these studies focus on integrating provenance from different frameworks in a shared environment. While any of these Big Data processing frameworks can be connected to a Data Lake, as we argued above, a Data Lake can not depend on such framework specific provenance collection mechanisms due to provenance integration challenges. Therefore, provenance stitching [13] techniques are hard to apply. Wang, J. et al. [20] identify the challenges in Big Data provenance which are mostly applicable in a Data Lake environment. Distributed Big Data provenance integration has been identified as a challenge in their work where we present a solution in the context of a Data Lake.

VI. CONCLUSION AND FUTURE WORK

The reference architecture for integrated provenance demonstrates early value of data provenance as a lightweight approach to traceability. Future work addresses the viability of the approach in obtaining necessary information without excessive instrumentation or manual intervention. Scalability of the technique is to be further assessed within a real Data Lake environment. Persistent ID solutions have tradeoffs; the suitability of one over another in the Data Lake setting is an open question. We have implemented only the stored provenance processing techniques in the presented prototype.

⁷<http://falcon.apache.org/>

⁸<http://nifi.apache.org/>

But our reference architecture highlights the power of real-time or live provenance processing in Data Lakes which is also left as a future work.

ACKNOWLEDGMENT

This work is funded in part by a grant from the National Science Foundation, ACI-0940824.

REFERENCES

- [1] B. Stein and A. Morrison. The enterprise data lake: Better integration and deeper analytics. [Online]. Available: <https://www.pwc.com/us/en/technology-forecast/2014/cloud-computing/assets/pdf/pwc-technology-forecast-data-lakes.pdf>
- [2] I. Terrizzano, P. M. Schwarz, M. Roth, and J. E. Colino, “Data wrangling: The challenging journey from the wild to the lake.” in *CIDR*, 2015.
- [3] M. Chessel, F. Scheepers, N. Nguyen, R. van Kessel, and R. van der Starre. (2014) Governing and managing big data for analytics and decision makers. [Online]. Available: <http://www.redbooks.ibm.com/redpapers/pdfs/redp5120.pdf>
- [4] How to design a successful data lake. [Online]. Available: <http://knowledgent.com/whitepaper/design-successful-data-lake/>
- [5] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, B. Plale, Y. Simmhan, E. Stephan, and J. V. den Bussche, “The open provenance model core specification (v1.1),” *Future Gener. Comput. Syst.*, vol. 27, no. 6, pp. 743–756, Jun. 2011.
- [6] L. Moreau and P. Groth, “Provenance: an introduction to prov,” *Synthesis Lectures on the Semantic Web: Theory and Technology*, vol. 3, no. 4, pp. 1–129, 2013.
- [7] I. Suriarachchi and B. Plale, “Provenance as essential infrastructure for data lakes,” in *IPAW*. Springer, 2016.
- [8] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock, “Kepler: an extensible system for design and execution of scientific workflows,” in *SSDBM*, June 2004, pp. 423–424.
- [9] J. Wang, D. Crawl, and I. Altintas, “Kepler+hadoop: A general architecture facilitating data-intensive applications in scientific workflow systems,” in *WORKS*. ACM, 2009, pp. 12:1–12:8.
- [10] H. Park, R. Ikeda, and J. Widom, “Ramp: A system for capturing and tracing provenance in mapreduce workflows,” in *VLDB*. Stanford InfoLab, August 2011.
- [11] S. Akoush, R. Sohan, and A. Hopper, “Hadoopprov: Towards provenance as a first class citizen in mapreduce,” in *TaPP*. USENIX Association, 2013, pp. 11:1–11:4.
- [12] M. Interlandi, K. Shah, S. D. Tetali, M. Gulzar, S. Yoo, M. Kim, T. D. Millstein, and T. Condie, “Titan: Data provenance support in spark,” *PVLDB*, vol. 9, no. 3, pp. 216–227, 2015.
- [13] P. Missier, B. Ludascher, S. Bowers, S. Dey, A. Sarkar, B. Shrestha, I. Altintas, M. Anand, and C. Goble, “Linking multiple workflow provenance traces for interoperable collaborative science,” in *WORKS*, Nov 2010, pp. 1–8.
- [14] W. Oliveira, D. de Oliveira, and V. Braganholo, “Experiencing prov-wf for provenance interoperability in swfmss,” in *IPAW*. Springer, 2014, pp. 294–296.
- [15] I. Suriarachchi, Q. Zhou, and B. Plale, “Komadu: A capture and visualization system for scientific data provenance,” *Journal of Open Research Software*, vol. 3, no. 1, 2015.
- [16] D. Crawl, J. Wang, and I. Altintas, “Provenance for mapreduce-based data-intensive workflows,” in *WORKS*. ACM, 2011, pp. 21–30.
- [17] W. Sansrimahachai, L. Moreau, and M. Weal, “An on-the-fly provenance tracking mechanism for stream processing systems,” in *ICIS*, June 2013, pp. 475–481.
- [18] I. Altintas, O. Barney, and E. Jaeger-Frank, “Provenance collection support in the kepler scientific workflow system,” in *Provenance and annotation of data*. Springer, 2006, pp. 118–132.
- [19] Y. Simmhan, B. Plale, and D. Gannon, “A framework for collecting provenance in data-centric scientific workflows,” in *ICWS*, Sept 2006, pp. 427–436.
- [20] J. Wang, D. Crawl, S. Purawat, M. Nguyen, and I. Altintas, “Big data provenance: Challenges, state of the art and opportunities,” in *Big Data*, Oct 2015, pp. 2509–2516.

Interactive Provenance Summaries for Reproducible Science

Xiang Li and Xiaoyang Xu

Computation Institute
University of Chicago
Chicago, Illinois, 60637

Email: xiaoyangx, lix1@uchicago.edu

Tanu Malik

School of Computing
DePaul University
Chicago, Illinois, 60604

Email: tanu@cdm.depaul.edu

Abstract—Recorded provenance facilitates reproducible science. Provenance metadata can help determine how data were possibly transformed, processed, and derived from original sources. While provenance is crucial for verification and validation, there remains the issue of the granularity—detail at which provenance data must be provided to a user, especially for conducting reproducible science. When data are reproduced successfully the need for detailed provenance is minimal and an essence of the recorded provenance suffices. However, when data are not reproduced correctly users want to quickly drill down into fine-grained provenance to understand causes for failure.

In this paper, we describe a drill-up/drill-down method for exploring provenance traces. The drill-up method summarizes the trace by grouping nodes and edges of the trace that have same derivation histories. The method preserves provenance data flow semantics. The drill-down method compares summary groups and ranks groups that may have information about the errors. Both the methods are implemented in an efficient manner using light-weight data structures so as to be suitable for reproducible science. We conduct a thorough experimental analysis to show how the operators perform in compressing and expanding real provenance graphs.

I. INTRODUCTION

Reproducibility requirements often entail experiments to be repeated in different computational environments [10], [4]. Containerizing has become a *de facto* method for reproducing computational experiments. In this method, code, data, and environment, i.e., elements associated with an experiment are copied into a single container. The resulting containers are significantly smaller in size than virtual machine images and require minimal installation and configuration in the new environment.

Provenance associated with a computational experiment is considered a crucial element of these containers. Included provenance describes how resulting data were derived from original sources and processed using binaries. A correct and authorized trace can thus be useful for validation and verification of future runs of the experiment, either exactly repeated or by changing parameters or data¹. While most tools for building containers, such as Smart Containers [6], Parrot [13],

¹While there is no standard definition for reproducibility [3], in this paper we imply repeating to imply an exact run of the experiment, and reproducing to imply a similar run obtained by changing any of the code, data, and environment

SciPackage [11], Light Virtualization [12] record provenance, interacting with the recorded provenance, especially for verification and validation is currently time-consuming.

Owing to the light-weight property of the containers, the modest-size recorded provenance, is typically stored in light-weight databases such as SQLite or LevelDB. These databases, however, provide limited interfaces for querying and particularly so for exploring provenance, which is stored as directed acyclic graphs. Consequently, interacting with provenance data for conducting reproducible science becomes time consuming.

Current methods [2], [8], [1] for exploring provenance graphs may be included within the container, but most are either non-interactive or require significant user input; for example, provenance browsers [1] can be included for interacting with provenance, but require explicit queries to be formulated. Similarly provenance views [2] are based on the user explicitly knowing parts of the graph that are of relevance. Such knowledge is typically known for workflow systems but absent for ad hoc computation experiments that must be reproduced. Statistical and graph clustering methods [8] provide one-time summaries but are merely restricted to determining the patterns of very large provenance graphs.

Practically, the user is concerned with reproducing the computational experiment. If the experiment reproduced correctly (based on human judgement), then user simply wants to verify if the new provenance trace is similar to the old. In this case, low-level details of the provenance trace are not important. Instead, a summary is sufficient. However, in case the experiment is not reproduced correctly, the low-level details of the provenance trace become vital, and the user must be able to drill-down into such details relatively quickly.

In this paper, we describe a drill-up/drill-down method for interacting with provenance traces in containers. The drill-up method summarizes provenance so as to absorb its essence quickly; the drill-down method enables a user to quickly find anomalies. The former is based on grouping nodes and edges of the execution trace that have similar derivation histories. It produces a summary that is complete and preserves provenance data flow semantics. The drill-down method, instead of producing another summary, provides users with guidance on which summary grouping is more informative than others. General graph summarization methods [14] use resolution

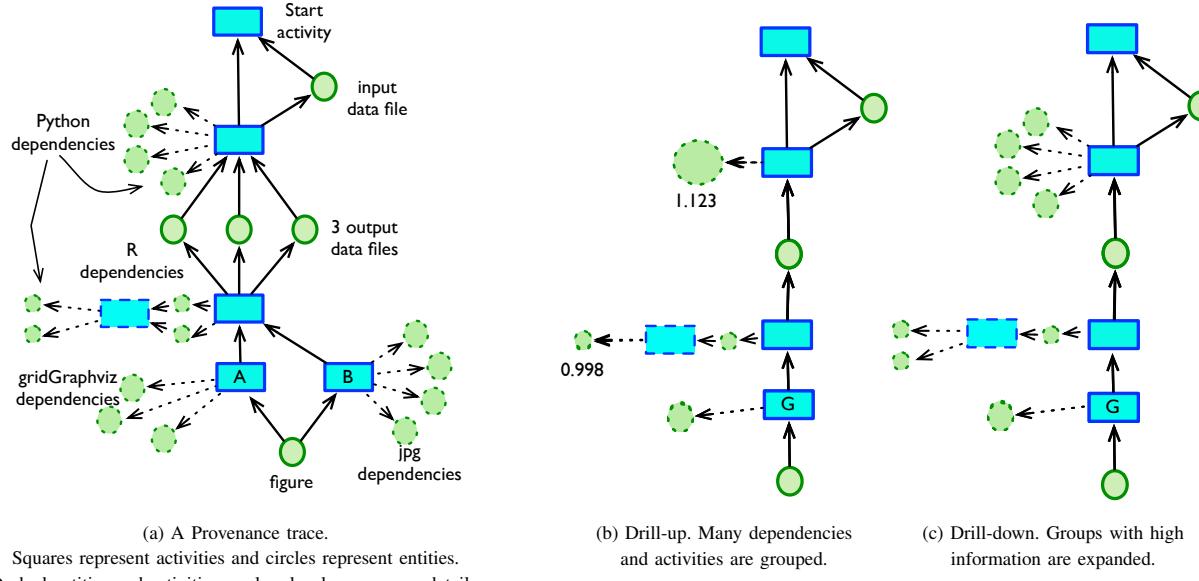


Fig. 1: The drill-up and drill-down operators on a provenance trace

parameter to drill-up/down but the entire graph needs to be processed, which is inefficient for quickly drilling down to a few specific nodes in a provenance graph.

Figure 1 shows the use of the drill-up and drill-down methods for provenance traces. In Figure 1(a) the computational experiment is a simple Python analysis program that reads input data, transforms it into data for three variables that are plotted using R and visualized as a jpg picture. A explicit start node is added to show start of the experiment. Most container techniques being operating system specific, record provenance at the program and process level. The dashed nodes denote software dependency information (crucial for reproducing and therefore recorded), and are low-level details which in the case the program reproduces correctly can be ignored.

The drill-up method automatically ignores such low-level details by aggregating nodes that have the same *derivation histories* into a group. So for example, execution trace nodes, such as *A* and *B* in the graph, which have same derivation history and are similar files in a directory are grouped. Given a node, its derivation history is determined by node and edge types of its ancestors. If a node has two ancestors then it acquires the derivation history of both.

If the figure, however, fails to reproduce, then the user would immediately like to drill down. The failure might arise due to several reasons—a missing dependency, incorrect input files, or activity taking longer than required. We describe a method for scoring the nodes of the summary graph. The score indicates which nodes of the summary graph may contain the error. Figure 1(b) shows a scoring on the summary graph and a drill-down based on that scoring (Figure 1(c)).

The paper makes the following contributions:

- **Drill-up and drill-down methods for provenance graphs:**

We summarize provenance graphs by grouping nodes based on their derivation history. We show that such a summary preserves data flow properties and is complete.

- **Efficient implementation within light-weight databases:** We show the summary method is light-weight and efficient, especially for conducting reproducible science in which the user wants to play with execution traces interactively, and for inclusion within light-weight databases used within reproducible packages.

- **Experimental evaluation:** We apply the methods to ad hoc workflows contained within reproducible packages and observing provenance at a fine granularity.

The rest of the paper is organized as follows. Section II describes a provenance model and drill-up/down requirements. The next two sections (Section III and Section IV) describe the drill-up and drill-down methods. Section V evaluates the efficiency and quality of the operators. Related work related to provenance and graph summarization is described in Section VI and we conclude in Section VII.

II. PROVENANCE RECORDING IN CONTAINERS

Most containers use application virtualization (AV) mechanisms to create a package and record its provenance. AV mechanisms track application code using system call interposition methods such as *ptrace* (Unix), *dyld* (Mac OS X), etc. These methods, during interposition copy code, data, and environment and also track provenance dependencies. This tracking is done for all input files, including dependencies, system processes, network processes, and temporary data that the application interacts with, thus resulting in fairly large amounts of provenance. Provenance recording, depending upon the AV method, typically include additional attributes such as process name, owner, group, parent, host, creation time, time of interaction, command line, environment variables and process binary's path. In the package, provenance is audited once by the author of the application and stored in the database as a reference for verification during re-execution time. For the recorded provenance, we assume that the following holds:

- 1) Recorded provenance defines a set of activity and entity types valid in for a domain.
- 2) Using recorded provenance it is straightforward to determine data dependencies that connect entities, e.g., a file written by a process p may depend on a file read by p .
- 3) The produced provenance can be represented in PROV.

Given the assumptions, we can define a model for recorded provenance generically as follows:

Definition 1 (Provenance Model). *Let \mathbb{L} be a domain of labels. A provenance model is a triple $\mathbb{P} = (\mathcal{A}, \mathcal{E}, \mathcal{L})$ where $\mathcal{A} \subseteq \mathbb{L}$ is a set of activity types and $\mathcal{E} \subseteq \mathbb{L}$ is a set of entity types. \mathcal{L} is a subset of $\mathbb{L} \times (\mathcal{A} \cup \mathcal{E}) \times (\mathcal{A} \cup \mathcal{E})$. Each triple in \mathcal{L} represents an edge type with an allowed start and end activity or entity type. We require that activity, entity, and edge labels be pairwise distinct.*

The recorded provenance can be defined in terms of a trace, formally defined as a graph in which the nodes are instances of the provenance model's activity and entity types and edges represent provenance dependencies.

Definition 2 (Provenance Trace). *Let $\mathbb{P} = (\mathcal{A}, \mathcal{E}, \mathcal{L})$ be a provenance model. A provenance trace for \mathbb{P} is a labeled directed graph $G = (V, E, T)$ with nodes V and edges $E \subseteq V \times V$. Each node must be of one of the activity and entity types specified in the provenance model and each edge must fulfill the type constraints specified by \mathcal{L} .*

Unlike the real provenance record in which each edge also records the time interval attribute, $T : E \rightarrow \mathbb{T} \times \mathbb{T}$ indicating when the two connected nodes interacted, in the provenance trace we ignore such interactions. In this paper we assume such time intervals to be negligible and interactions to be instantaneous. This is a safe assumption given that our current objective is summarization of the graph.

Given a provenance model \mathbb{P} and its corresponding execution trace G , we consider two operators, the drill-up operator, $\ominus(G)$ which creates a summary ϕ of G and $\oplus(\phi)$ which given a summary, will create a information entropy based ranked order of groups to drill-down upon. Given the provenance model and the trace, the operators must satisfy the following three properties:

- 1) **The summary must be complete:** The resulting summary graph must be complete in that every node and edge in G can be mapped to a node and edge of the summary graph.
- 2) **The summary must preserve provenance data flows:** The summary must preserve the provenance data flows of G , i.e every path in the summary graph is a subset of the paths in G .
- 3) **The summary must guide to erroneous nodes or outliers:** The summary must not deeply hide nodes and edges that are useful for reproducing experiments.

III. DRILL-UP OPERATOR

The drill-up operator is based on grouping nodes into groups that have the same type (*i.e.*, entity or activity) and have the

same derivation history, where derivation history implies the same node and edge labels from the start node. Since G is a directed acyclic graph, a grouping consists of nodes that do not have a relationship between them. To identify nodes with the same derivation history first nodes of the same type are grouped, defined as

Definition 3 (Node Grouping). *$\phi = \{G_1, G_2, \dots, G_k\}$ is a node-grouping such that*

- (1) $\forall G_i \in \phi, G_i \subseteq \mathcal{A}(\mathcal{G})$ or $G_i \subseteq \mathcal{E}(\mathcal{G})$, and $G_i \neq \emptyset$,
- (2) $\cup_{G_i \in \phi} G_i = V(G)$,
- (3) $\forall G_i, G_j \in \phi$ and $(i \neq j), G_i \cap G_j = \emptyset$.

For a given node grouping ϕ , the ancestor groups of a node v is the set $AncestorGroups_{\phi, E}(v) = \{Label(\phi(u)), Label \in \mathcal{L} \text{ and } (u, v) \in E\}$. Since, ancestor groups is a set, a node may have different ancestor groups. Nodes that do not have an ancestor are assigned the start node as an ancestor, with a start label edge. Now we define grouping nodes by derivation history.

Definition 4 (Derivation History Grouping). *A grouping $\phi = \{G_1, G_2, \dots, G_k\}$ has the same derivation history if it satisfies the following:*

- (i) *Node Grouping Definition 3,*
- (ii) $\forall u, v \in V(G)$, if $\phi(u) = \phi(v)$, then $\forall E_i \in \mathcal{L}, AncestorGroups_{\phi, E_i}(u) = AncestorGroups_{\phi, E_i}(v)$.

Finally, we define the drill-up operator $\ominus(G)$ in terms of the derivation history grouping.

Definition 5 (Drill-down). *The drill-down operator \ominus takes as input G that conforms to the provenance model \mathbb{P} and outputs ϕ that satisfies Definition 4.*

The derivation history grouping definition differs from [14] in two distinct ways. First, the grouping in [14] is defined on both ancestors and descendants, which implies that nodes with the same types and same ancestors are separated into different groups if their descendants are different. From a provenance perspective, such a separation is redundant. Second, to enable grouping by the same ancestors, we also consider edge labels as part of group definition. In [14] the neighbor group definition is only based on node labels. Thus the derivation history grouping is homogeneous in terms of node types and commonality of ancestors. We now show that $\ominus(G)$ as defined preserves the provenance data flow and is complete, which is not the case for the grouping defined in [14].

Theorem 1. $\ominus(G)$ is complete and preserves the data flow.

Proof. We first prove that $G' = \ominus(G)$ results in a complete graph. Let there be an edge $E = (u, v), E \in G$ that is not present in E' , where E' is the set of edges in G' . This implies that nodes u and v are not in ϕ . However, this violates condition (2) for node grouping. Thus all nodes and edges in G .

We now prove that G' preserves the data flow *i.e.*, every path in the summary graph is a subset of the paths in G . In

other words there does not exist a path that is in G' but not in G . Let us assume there exists a path $P = u_{G_1} \rightarrow v_{G_2} \rightarrow w_{G_3}$ in G' , where u_{G_1} is the node corresponding to group G_1 and Path P does not exist in G . This implies that there is a path from nodes in u_{G_1} to nodes in v_{G_2} and v_{G_2} to w_{G_3} but not from u_{G_1} to w_{G_3} . By this definition v_{G_2} must have at least two nodes v' and v'' grouped together such that there is a path u to v' and v'' to w . This implies u is an ancestor of v' but not of v'' . Such a grouping violated grouping by derivation histories (condition (ii) of Definition 4). \square

The details of a fast implementation of $\ominus(G)$ based on topologically sorting G and when child nodes are combined checking if topological sort is violated are provided in the companion technical report [16]. The full pseudo code of the algorithm of $\ominus(G)$ is also provided in the report.

IV. DRILL-DOWN OPERATOR

The drill-down operator helps a user to flexibly zoom-in into groups of the summary graph which have more information about errors that arise while reproducing an experiment. In this section, we present a qualitative treatment about how to determine the subset of groups that must be drilled-down. The specific description of the types of errors is present in our experimental section (Section V).

To determine the groups to drill-down, we match the immutable $G = (V, E, T)$ available to us as part of the reproducible package with a new $G' = (V', E', T)$ generated when the package is run again. For nodes that are not present in G' , i.e., $V - V'$, we need to determine specific ancestors and their corresponding groups in the summary graph. These groups are the potential groups in which to drill down. One way to identify such groups is to compute the edit distance between G and G' through a sequence of vertex and edge deletion operations and then check for isomorphism. However, it will still not identify the specific ancestors that are the cause of errors and their corresponding groups in the summary graph since there can be many such ancestors. In addition, checking for isomorphism is redundant since the graphs belong to the same computational experiment.

To determine specific ancestors (descendants), which need to be identified multiple times, instead of querying the graph repeatedly, we maintain a sketch of the provenance graph. This sketch is implemented as a matrix filter [9] in which the node u of an edge $(u \rightarrow v)$ is hashed into a row array of m bits, and the node v into a column array of m^2 bits. Using k independent hash functions $\{h_1, \dots, h_k\}$, each of which has a range of $\{1, \dots, m\}$, the i^{th} bit b_i of the row array is associated with the i^{th} set of m bits in the column array. A count of the number of ancestors(descendants) is maintained.

For nodes in $V - V'$, we determine the signature of the ancestors. All nodes and their corresponding groups that satisfy the signature are then identified. Obtaining the signature of ancestors is as $O(1)$ operation. Once a group is identified, we ignore checking for all other nodes in that group, thus identifying the groups an $O(G)$ operation.

While the above method will identify the potential groups to drill down, the number of groups to drill down may itself be several. To guide the user to relevant groups which might be more useful to drill down, we compute an inverse entropy measure of each group. Intuitively this measure informs how useful a group is for the purpose of drill-down in that what is the probability of finding a node that is directly leading to the cause of incorrect provenance graph. To determine useful groups, the nodes in a group are weighed with numerical measures. Given a weighted node group, an inverse entropy measure is defined as:

$$1/H(w_1, w_2, \dots, w_n) = -1/\left(\sum_{i=1}^n \left(\frac{w_i}{W}\right) \log_2\left(\frac{w_i}{W}\right)\right) \quad (1)$$

where W denotes the sum of the reals, if $W > 0$, and 0 otherwise. We take $0\log_2(0)$ to be 0 in this computation. (Also, from here onward, we denote \log_2 by \lg .) We define the entropy of a node-weighted, n -node group G_i with node weights w_1, w_2, \dots, w_n , to be $H(w_1, w_2, \dots, w_n)$. We will also use the shorthand notation $H(G_i)$ to denote $H(w_1, \dots, w_n)$.

The justification for using an entropy follows from the fact that we wish to drill down into the group that provides the most information about the distribution of node weights to the viewer. It would not be very informative, for example, to drill down into a group in which all node weights are identical, which is equivalent to finding a group that has the maximum entropy. However, if in a group 99% of the weight is concentrated in a few nodes then the user will possibly be interested in drilling down in this group to see what properties of the nodes give the group a larger weight. The intuition to find the inverse of the entropy agrees with maximizing entropy, since entropy is maximized when all the weights are identical.

We describe two ways to assign node weights, in decreasing order of computational efficiency.

1) *By node properties:* Each activity and entity node recorded in G consists of several properties, such as file sizes, paths, access, process cpu and memory consumption, time it took to run the process, etc. The objective in node properties is to weight node based on the normalized values of the properties. For instance, entity nodes can be weighted by file sizes, and activity nodes by process memory consumption, since these two property values a good indication.

2) *By number of descendants:* We weight nodes based on the number of other nodes in whose lineage a given node appears. Intuitively this is weighing nodes that have a chance of stronger influence on their descendants. To determine this value we simply use the count value in the matrix filter.

Finally, the drill-down operator \oplus is defined as

Definition 6. *The \oplus operator takes as input ϕ and G' and outputs φ_{ranked} , a list of groups to expand ordered by Equation 1*

V. EXPERIMENTAL ANALYSIS

A. Implementation

1) *LevelDB implementation:* LevelDB is a fast key-value storage library from Google that provides an ordered mapping

from string keys to string values. Its API only supports Put, Get, Delete, CreateSnapshot, WriteBatch, and RangeIteration functions. Bloom filters are part of LevelDB. To store a graph in a LevelDB, we use LevelGraph which is a wrapper for storing graphs in LevelDB. LevelGraph considers each edge as an RDF triple and uses six indices for every triple in order to access them as fast as possible. The intuition for storing six triple is based on HexaStore [15] in which an edge is indexed in six possible ways, one for each possible ordering of the edge. The replicated indices provide upto five times speedup. We have implemented our drill-up and drill down methods as C++ routines in LevelGraph. All experiments were run on a 2.8GHz Pentium 4 machine running Ubuntu 14.04, and equipped with a 250GB SATA disk.

2) *Provenance datasets in Containers:* In general, we can build containers of all application programs. In this work, we consider applications for which we have established containers because those applications were as described by scientific communities either hard to reproduce, or in general required containerization, an in which we can test reproducibility scenarios. These containers are built with our SciPackage [11] in which provenance is audited using *ptrace*. We term these applications as **Athena**, which is a major experiment at the CERN Large Hadron Collider, **VIC**, which is a large-scale hydrologic model that applies water and energy balances to simulate terrestrial hydrology at a regional spatial scale [7], and **Swift**, where in we consider a raytracing workflow implemented and executed through the Swift parallel scripting language [5]. For details on these applications, we refer the reader to the companion technical report [16].

Table I shows the number of nodes and edges, average degree, and types of PROV edge relationships, and the number of properties that are recorded in these datasets.

TABLE I: Provenance graph analysis of analyzed datasets

	Number of Nodes	Number of Edges	Average Degree	PROV Edge Type	Number of Properties
Athena	7392	10237	3	11	25
VIC	3288	4936	2	6	12
RayTracing	1045	1284	6	5	4

3) *Failures in reproducing:* To evaluate the drill-down operator we must be able to simulate failures in workflows. The most common reproducibility issue in these workflows is missing dependency and here containerization solves that problem completely. Thus we must not be drilling down into dependency information. Thus instead we change input program and parameters for Athena and VIC workflows based on program semantics which lead to failures. Nodes and groups of interest that must be drilled-down into are manually identified, and tested against the drill-down operator. In these workflows, node weights are assigned based on number of descendants. In Swift no errors are introduced, but drill down is based simply on node properties.

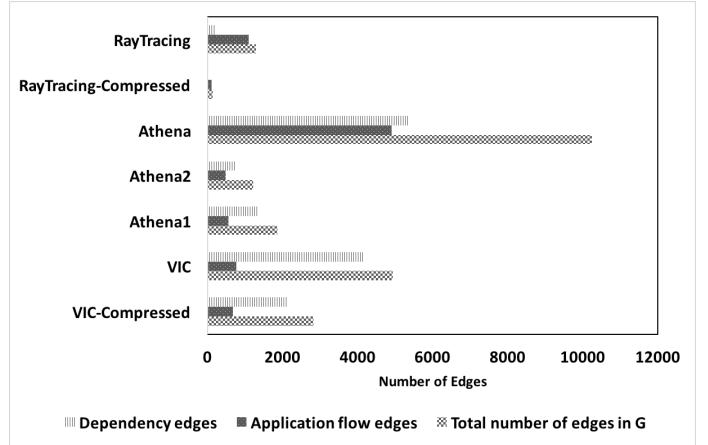


Fig. 2: Compression in Athena, VIC, and Swift

B. Effectiveness Evaluation

1) *Is there a summarization?:* We apply drill-up to the provenance traces generated from the workflows to see if there is effective summarization of the traces. The majority of edges in Athena and VIC are dependency edges, and so the measure for drill-up is to determine how good is the compression within dependency edges. To be able to compare the relative performance of drill-up on the different traces, we plot (i) the total number of edges, (ii) the number of edges corresponding to the application flow, (iii) the number of edges that are dependencies in the original graph and compare these values with that of the drilled-up graph (Figures 2).

We see a significant reduction in total number of edges compression ratio between 2.4 and 9.0 for all applications. Swift is a homogeneous workflow and therefore the compression is maximum. In Athena, we consider two runs Athena1, which is exactly repeating the workflow and Athena2 in which the input program is changed. While there is a factor of 8.5 reduction for Athena1, the repeatable workflow, the compression is reduced to 5.5 for the reproducible workflow. The variation is simply due to change in input parameters. For VIC there the compression is good with respect to dependencies, but not good with respect to main application flow. In reality there is compression within each sub-module of the hydrological parameter but barely any across them.

2) *Drill-down performance: Identifying ancestors in groups:* We examine if we are correctly able to identify ancestors, especially as the provenance graph grows. We measured the number of false positive answers to queries about whether a path between two vertices exists in a matrix filter. Since the sketch contains provenance metadata of an increasing number of nodes and edges it is expected to provide an increasing number of false positive responses. Figure 3 shows that the rate of such false positives is very low, ensuring that the sketches are robust as the provenance grows.

3) *Drill-down performance: Does the drill-down guide to the most important groups?:* For each workflow, the table (Table II) compares the number of groups that were identified as relevant to be drilled, and the number of groups which were

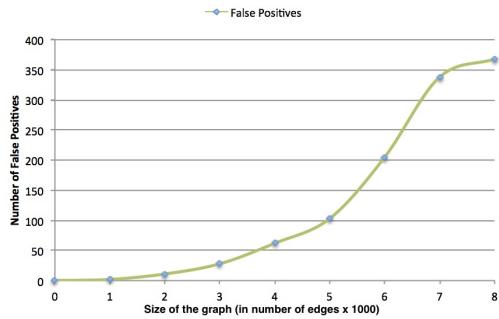


Fig. 3: Rate of False Positives

TABLE II: Group guidance

	Number of groups to be drilled	Number of groups not identified	Ranked Order
Athena	5	1	Yes
VIC	3	0	No
RayTracing	2	1	No

not identified for drilling. The table also shows the entropy-based ranked order and the order given to us by the workflow developers as the ones they would like to drill down.

VI. RELATED WORK

ZOOM*userview [2] provides provenance graph abstractions to focus on the most relevant information. A user view is similar to drill-up and determines what level of sub-workflow the user can see, and thus what data and tasks are visible in provenance queries. While, we have focused on automatic creation of relevant views instead of they being user-defined like in ZOOM*userview, we also address the problem of selective drill-down, which is unaddressed in ZOOM*userview.

Macko et. al [8] summarize large system graphs using statistical clustering methods and identify semantically meaningful tasks in an objects history. Our methods are non-statistical and based on pure aggregation of nodes and derivation histories. Thus they are easier to implement and use in light-weight databases embedded within reproducible packages than clustering based methods.

Tian et. al [14] define SNAP and k-SNAP operators on general graphs. These operators are based on user-selected node attributes and relationships and allow the user to control the resolution (i.e. the number of groups in a summary) based on the value of k. While SNAP works on both directed and undirected graphs, in this work, we have adapted SNAP for provenance graphs: grouping nodes with same ancestors but different descendants, and providing guidance to user about how to explore summary groups.

Typical relational databases come with query languages to support drill-up/drill down operations. In reproducible packages, provenance is however, stored in light-weight databases, such as SQLite and LevelDB. These databases support limited form of SQL and currently do not support the rollup operators. In this paper we have implemented the operators within LevelDB, a lightweight key-value database.

VII. CONCLUSION

In this paper we have presented an interactive method for exploring provenance that is included as part of the reproducible packages. The method provides two operators for interaction, a drill-up method that summarizes provenance graphs based on derivation histories, and a drill-down method that provides guidance into which summary group contains the nodes that are likely sources of error or anomalies. Through experimental evaluation we show that the methods are useful for summarizing large and complex provenance graphs obtained from reproducing both workflows and ad hoc data analysis experiments. The source code of our work is available at <http://www.github.com/tanumalik/ProvSummary>.

ACKNOWLEDGEMENTS

This work was supported in part by National Science Foundation grants ICER-1440327, SES-0951576, and ICER-1343816.

REFERENCES

- [1] M. K. Anand, S. Bowers, and B. Ludäscher. Provenance browser: Displaying and querying scientific workflow provenance graphs. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pages 1201–1204. IEEE, 2010.
- [2] S. Cohen-Boulakia, O. Biton, S. Cohen, and S. Davidson. Addressing the provenance challenge using zoom. *Concurrency and Computation: Practice and Experience*, 20(5):497–506, 2008.
- [3] Reproducibility of Data-Oriented Experiments in eScience. Dagstuhl Seminar 16041. <http://www.dagstuhl.de/de/programm/kalender/sempn/?sempn=16041>, 2016.
- [4] J. Freire, P. Bonnet, and D. Shasha. Computational reproducibility: state-of-the-art, challenges, and database research opportunities. In *SIGMOD*, 2012.
- [5] L. M. Gadelha Jr, B. Clifford, M. Mattoso, M. Wilde, and I. Foster. Provenance management in swift. *Future Generation Computer Systems*, 27(6):775–780, 2011.
- [6] D. Huo, J. Nabrzyski, and C. F. Vardeman II. Smart containers: An ontology design pattern towards preservation of computational experiments. <http://linkedscience.org/wp-content/uploads/2015/04/Linked-Science-ISWC-2015-Huo-et-al.pdf>, 2016.
- [7] X. Liang, E. F. Wood, and D. P. Lettenmaier. Surface soil moisture parameterization of the vic-2l model: Evaluation and modification. *Global and Planetary Change*, 13(1):195–206, 1996.
- [8] P. Macko, D. Margo, and M. Seltzer. Local clustering in provenance graphs. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 835–840. ACM, 2013.
- [9] T. Malik, L. Nistor, and A. Gehani. Tracking and sketching distributed data provenance. In *International Conference on eScience*, 2010.
- [10] T. Malik, Q. Pham, and I. T. Foster. *SOLE: Towards Descriptive and Interactive Publications*. CRC Press, 2014.
- [11] Q. Pham, T. Malik, and I. T. Foster. Using provenance for repeatability. In *USENIX NSDI Workshop on Theory and Practice of Provenance (TaPP)*, 2013.
- [12] Q. Pham, T. Malik, B. Glavic, and I. Foster. LDV: Light-weight Database Virtualization. In *ICDE*, pages 1179–1190, 2015.
- [13] D. Thain and M. Livny. Parrot: An application environment for data-intensive computing. *Scalable Computing: Practice and Experience*, 6(3):9–18, 2005.
- [14] Y. Tian, R. A. Hankins, and J. M. Patel. Efficient aggregation for graph summarization. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 567–580. ACM, 2008.
- [15] C. Weiss, P. Karras, and A. Bernstein. Hexastore: sextuple indexing for semantic web data management. *Proceedings of the VLDB Endowment*, 1(1):1008–1019, 2008.
- [16] X. Xu, X. Li, and T. Malik. Technical Report. Interactive Provenance Summaries for Reproducible Science. <http://dbgroup.cdm.depaul.edu/pubs/2016/ProvSummary>.

ExTASY: Scalable and Flexible Coupling of MD Simulations and Advanced Sampling Techniques

Vivekanandan Balasubramanian*, Iain Bethune†, Ardita Shkurti‡, Elena Breitmoser†
Eugen Hruska¶||, Cecilia Clementi§||, Charles Laughton‡ and Shantenu Jha*

*Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ 08854, USA

†EPCC, The University of Edinburgh, James Clerk Maxwell Building, Peter Guthrie Tait Road, Edinburgh, UK, EH9 3FD

‡ School of Pharmacy and Centre for Biomolecular Sciences, University of Nottingham,
University Park, Nottingham NG7 2RD, UK

§ Department of Chemistry, Rice University, Houston, TX 77005, USA

¶ Department of Physics, Rice University, Houston, TX 77005, USA

|| Center for Theoretical Biological Physics, Rice University, Houston, TX 77005, USA

Abstract—For many macromolecular systems the accurate sampling of the relevant regions on the potential energy surface cannot be obtained by a single, long Molecular Dynamics (MD) trajectory. New approaches are required to promote more efficient sampling. We present the design and implementation of the Extensible Toolkit for Advanced Sampling and analYsis (ExTASY) for building and executing advanced sampling workflows on HPC systems. ExTASY provides Python based “templated scripts” that interface to an interoperable and high-performance pilot-based run time system, which abstracts the complexity of managing multiple simulations. ExTASY supports the use of existing highly-optimised parallel MD code and their coupling to analysis tools based upon collective coordinates which do not require a priori knowledge of the system to bias. We describe two workflows which both couple large “ensembles” of relatively short MD simulations with analysis tools to automatically analyse the generated trajectories and identify molecular conformational structures that will be used on-the-fly as new starting points for further “simulation-analysis” iterations. One of the workflows leverages the Locally Scaled Diffusion Maps technique; the other makes use of Complementary Coordinates techniques to enhance sampling and generate start-points for the next generation of MD simulations. We show that the ExTASY tools have been deployed on a range of HPC systems including ARCHER (Cray CX30), Blue Waters (Cray XE6/XK7), and Stampede (Linux cluster), and that good strong scaling can be obtained up to 1000s of MD simulations, independent of the size of each simulation. We discuss how ExTASY can be easily extended or modified by end-users to build their own workflows, and ongoing work to improve the usability and robustness of ExTASY.

I. INTRODUCTION AND MOTIVATION

Approximately 30-40% of compute cycles on US XSEDE [1], [2] is devoted to research on biomolecular systems using Molecular Dynamics (MD) simulations. Much of the computational cost comes from the need for an adequate sampling of the conformational space accessible to these complex and flexible systems in order to answer a particular research question. For example, to calculate free energies one needs an adequate sample from the Boltzmann weighted ensemble of states for the system in order to estimate the thermodynamic quantity of interest. Another example is the study of kinetic processes such as self-assembly or drug-target association, where the integration of data from large numbers

of trajectories is required to build a statistically meaningful model of the dynamical process.

The high dimensionality of these macromolecular systems and the complexity of the associated potential energy surfaces (creating multiple metastable regions connected by high free energy barriers) pose significant challenges to adequately sample the relevant regions of the configurational space. In other words, beside the “curse of dimensionality” associated with the large number of degrees of freedom, MD trajectories can easily get “trapped” in a low free energy state and fail to explore other biologically relevant states. The waiting time to escape from a local free energy minimum increases exponentially with the height of the free energy barrier that needs to be crossed to reach another state. Metastable states separated by free energy barriers of several tens of $k_B T$ (where k_B is the Boltzmann constant and T is the physiological temperature) are not uncommon in biologically relevant systems, but can not at present be routinely sampled with standard MD simulations.

In practice, better sampling of the relevant regions of a macromolecule configurational space can be achieved through methodologies able to bias the sampling towards scarcely visited regions, reducing the waiting time inside a metastable state by artificially flattening the energy barriers between states - e.g. Metadynamics [3] or Accelerated Dynamics [4]. Although the results can usually be reweighed to reproduce the correct Boltzmann statistics, kinetic properties are not easily recovered from biased simulations (without combining with unbiased simulations, e.g. [5]). In addition, the design of an effective bias usually requires some *a priori* information on the system of interest, for instance a suitable choice of collective variables to describe long timescale processes.

An alternative approach to tackle the sampling problem is the development of ensemble or swarm simulation strategies, where data from large numbers of simulations, which may be weakly coupled or not coupled at all, are integrated (e.g. Replica Exchange [6] and Markov State Models (MSMs) [7]).

This last class of methods is of increasing interest for a variety of reasons. Firstly, the hardware roadmap is now based almost entirely on increasing core counts, rather than

clock speeds. These developments appear to favour weak scaling (larger and larger molecular systems to be simulated) over strong scaling (getting more data faster on a system of fixed size). However, by running ensembles of simulations over these cores and integrating the data using, e.g. MSM approaches, timescales far in excess of those sampled by any individual simulation are effectively accessed. In the last few years several studies have been published [8]–[12] where, using MSM methods, processes such as protein folding or ligand binding have been completely and quantitatively characterized (thermodynamically and kinetically) from simulations orders of magnitude shorter than the process of interest.

It is becoming increasingly clear that the application of ensemble simulation strategies on state-of-the-art computational facilities has an unparalleled potential to permit accurate simulations of the largest, most challenging, and generally most biologically relevant, biomolecular systems. The main challenge in the development of the ensemble approach for faster sampling of complex macromolecular systems is on the design of strategies to adaptively distribute the trajectories over the relevant regions of the systems configurational space, without using any *a priori* information on the system global properties. The definition of smart “adaptive sampling” approaches that can redirect computational resources towards unexplored yet relevant regions is currently of great interest.

In light of the challenge posed by trends in computer architecture, the need to improve sampling, and the range of existing MD codes and analysis tools, we have designed and implemented the Extensible Toolkit for Advanced Sampling and analYsis (ExTASY). ExTASY provides three key features within a single framework to enable the development and applications requiring advanced sampling in a flexible and highly scalable environment. Firstly, as an extensible toolkit, ExTASY allows a wide range of existing software to be integrated, leveraging the significant community investment in highly optimised and well-tested software packages and enabling users to continue to work with tools that they are familiar with. Support for specific MD codes and analysis tools is provided in order to demonstrate how ExTASY may be used, but users can easily add other tools as needed.

Secondly, ExTASY is flexible, providing a programmable interface to link individual software components together and construct sampling workflows. Workflows capture a sequence of execution of individual tools, and the data transfers and dependencies between them. First class support is provided for defining large ensembles of independent simulations. Thus complex calculations may be scripted and then executed without the need for user intervention.

Thirdly, ExTASY workflows may be executed either locally, or on remote High Performance Computing systems. Complexities such as the batch queueing system and data transfer are abstracted, making it easy for users to make use of the most appropriate compute resources they have access to. In addition, this abstraction allows workflows to be executed without exposing each component to queue waiting time, and respecting the dependencies defined in the workflow, for many

simulations to be scheduled and executed in parallel.

The rest of the paper is organized as follows: In the next section we discuss the design and implementation of ExTASY. After a brief discussion in Section III of two distinct applications that have been used to design and validate ExTASY, Section IV provides a careful analysis of the performance and scalability of ExTASY. Given the complex interplay between functionality and performance when designing an extensible and production tool, we perform a wide range of experiments aimed to investigate strong and weak scaling properties, *inter alia* over a set of heterogeneous HPC platforms. We conclude with a discussion of the scientific impact as well as the lessons for sustainable software development.

II. RELATED WORK

The need for better sampling has driven developments in algorithms, hardware, and software for molecular simulation.

One of the features of the popular Metadynamics method [3] or Accelerated Dynamics [4] is that a constant analysis of what has been sampled so far is used to bias future sampling into unexplored regions. A range of alternative approaches are now emerging that do likewise, but where alternating segments of data-gathering, and analysis to inform the direction of future sampling, are more coarsely grained. This iterative approach has the advantage over the Metadynamics method that the identity of “interesting” directions for enhanced sampling does not need to be defined *a priori*, but can emerge and respond flexibly to the developing ensemble. Another advantage is that the MD-based sampling process and analysis method do not have to be implemented within the same executable, or in two tightly-coupled executables, permitting greater flexibility. Many such methods make use of collective variables (CVs) to define directions in which to promote sampling. A variety of novel, and established, algorithms for the unsupervised and adaptive construction of CVs exist. In addition to the work of Preto and Clementi [13], interleaving cycles of MD simulation with data analysis through Locally Scaled Diffusion Maps [14], related methods include the non-targeted PaCS-MD method of Harada and Kitao [15], variants thereof [16], and the PCA-based method of Peng and Zhang [17].

Better sampling can also come from faster sampling, which has been enabled through hardware developments such as ANTON [18] and MD-GRAPE [19]. These special purpose computers enable much faster calculations of the different contributions to the forces along the trajectories, thus speeding up the clock time required to perform a time integration step in the MD simulation and allowing execution of significantly longer MD trajectories. The cost of, and access to such special purpose computers ensure that in spite of their potential, they will not be as accessible for the wider scientific community as general purpose approaches. Further ANTON requires a customized ecosystem, from bespoke MD engines and ANTON specific data analysis middleware (e.g., HiMach). Thus ANTON style special-purpose approaches to biomolecular simulation science cannot take advantage of the rich community driven advances and eco-system.

Methods such as Replica Exchange and Metadynamics require a tight coupling between the simulation and analysis processes, and are thus typically implemented within the core MD code (e.g. replica exchange methods are implemented in AMBER [20], CHARMM [21], GROMACS [22], LAMMPS [23], and NAMD [24], for example), or are provided by a separate package that communicates in a fine-grained manner with the running MD program through specially-installed “hooks”; for example, the PLUMED package [25] that provides Metadynamics capabilities (amongst others) to AMBER, GROMACS, LAMMPS, NAMD and also Quantum ESPRESSO [26].

General-purpose workflow systems may be applied in the area of MD, but this requires much effort in customising the workflow system to the particular application, often requiring input from developers themselves. There also exist workflow systems that are built from scratch for specific MD applications such as [27]. To our knowledge, there is no established and generally available package to support the types of coarser-grained, adaptive MD applications described above. In many cases, such applications are created using customized scripts that are tightly coupled to a particular workload and/or compute resource. This approach is not extensible, scalable or fault-tolerant and increases the burden on both users and developers.

III. EXTASY: REQUIREMENTS, DESIGN AND IMPLEMENTATION

In this section we first present the requirements that have been considered in the design and implementation of ExTASY, which we then go on to discuss.

A. Requirements

Consistent with the design of many new software systems and tools, we analyze the functionality, performance and usability requirements of ExTASY.

1) Functionality: Specific to sampling, there is a need to couple two very distinct computational stages; each stage is short-lived when compared to the typical duration of a monolithic simulation. Furthermore, the two stages differ in their resource requirements significantly: one stage is characterized by multiple compute intensive MD simulations, the other by a single analysis program that operates on data aggregated from multiple simulations. The “Ex” in ExTASY is a reference to the extensible nature of the framework, thus any coupling must be between the abstraction of stages and not specific codes.

Scientists may have access to multiple systems, and wish to submit jobs on each, or may be forced to migrate from using one system to another due to CPU time allocation, or system end-of-life. It is imperative that any software system support **interoperability**, i.e. use of heterogeneous systems with minimal changes. MD simulations may be executed over multiple nodes, depending on the system size, thus ExTASY should also support the **ability to execute tasks** over multiple nodes.

2) Performance: In order to obtain good sampling, a large number of simulations must be executed. In many cases, the aggregate number of cores required by the set of simulations (“ensemble”) is much higher than the total number of cores that are available at a given instance or that can be allocated on a system. The framework must decouple the aggregate (or peak) resource requirement of the workload from the number of cores available or utilized. On the other hand, if access to a large number of cores is available, the framework should be able to use them effectively. In this regard, the **strong and weak scalability** of the framework is to be investigated.

3) Usability: Depending on the application, the user might need to change to using a larger/smaller set of input data, modify simulation parameters or replace any of the simulation or analysis tools. The framework should offer **easy application setup**, minimizing the user’s time and effort. The user should only be concerned with decisions on “what” the workflow is and “where” it is being executed. The details as to “how” the deployment and execution occurs should be hidden by the underlying software. Thus the framework should use tools that **abstract complexities** of deployment and execution from the user. Workflow users and developers should be able to concentrate their efforts on the application logic and expect the underlying software to provide a level of transparent and automatic data movement and job submission.

B. Design

From these requirements, we identify the following as the primary design objectives of ExTASY.

- 1) Support range of HPC systems, abstract task execution, data movement from the user.
- 2) Flexible resource management and coupling capabilities between different stages as well as within a stage.
- 3) Provide the users with an easy method to specify or change workload parameters without delving into the application itself.

These design objectives put together lead to the following simple software architecture:

1) Middleware for resource and execution management: The ExTASY framework is aimed to provide an easy method to run advanced sampling algorithms on HPC systems. In this process, the ExTASY framework should abstract users from the complexity of application composition, workload execution and resource management. There is need for a middleware that provides such resource and execution management that provides many of the required functionalities and performance. The details of how tasks are mapped or executed on to resources is abstracted from the ExTASY user. The user is only required to provide details of the workload and identify the resource(s) that are accessible. This design choice thus acknowledges the separation of concerns: workload description from its execution on HPC systems.

2) Configuration files: Composing the advanced sampling algorithms discussed in the previous sections using the components provided by a particular middleware stack, requires specific knowledge of the middleware itself. The ExTASY

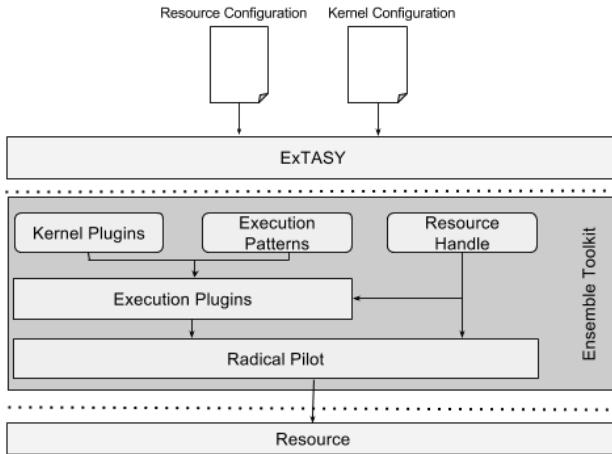


Fig. 1. Design of the ExTASY framework using Ensemble Toolkit as middleware. The ExTASY framework provides ready-to-use scripts created using components provided by Ensemble Toolkit. Parameters related to the resource and workload are exposed via configuration files, which alone are the files that users interact with. Within Ensemble Toolkit, the workload is converted into executable units by the execution plugins and submitted to the resource using RADICAL-Pilot.

framework bypasses this complexity by adding one more level of abstraction. It provides ready-to-use scripts in accordance with advanced sampling methods discussed in this paper that are configurable via configuration files. In these configuration files, the user is exposed to only application-level, meaningful parameters that can be modified as necessary (see [28]).

C. Implementation

1) *Ensemble Toolkit*: As mentioned previously, ExTASY requires a middleware for resource and execution management. We chose to use Ensemble Toolkit [29] as the middleware component as it provides several relevant features such as the ability to support MPI tasks, dynamic resource management – one type of which is to be able to execute more tasks than the resources available, support for heterogeneous HPC systems and strong and weak scalability guarantees. Ensemble Toolkit has been tested up to 1,000s of tasks with short and long term plans to support 10,000s and 100,000s of tasks [29]. Ensemble Toolkit is in turn based upon the pilot abstraction (and the RADICAL-Pilot [30] implementation of the pilot abstraction) to provide much of the flexible and scalable resource management capabilities. For example, in addition to the use of HPC systems shown in this paper, RADICAL-Pilot supports execution on `localhost`, generic Linux clusters, and cloud resources such as Amazon EC2.

Ensemble Toolkit exposes three components to the user that can be used to express many applications: Kernel Plugins, Execution Patterns, and Resource Handles. Scripts that are part of ExTASY framework use these components to describe the application logic.

2) *Configuration files*: The application logic is expressed via components of Ensemble Toolkit. The resource and the workload specifications are exposed via configuration files. The ExTASY framework has two types of configuration files:

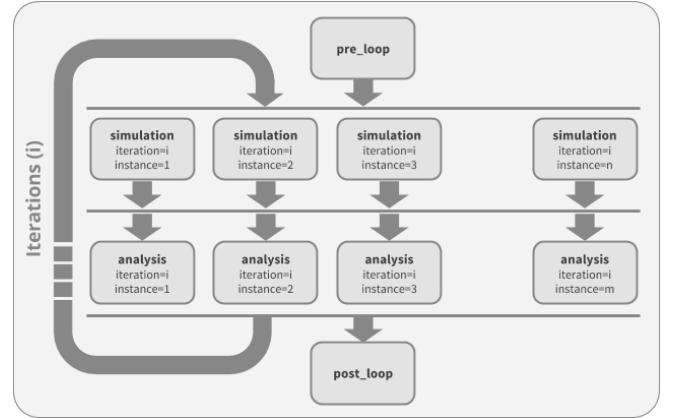


Fig. 2. The SAL pattern common to both sampling algorithms. The crux of the pattern is an iteration over 2 stages: simulation and analysis, where the number of simulation and analysis instances can be different. The pattern may also contain pre- and post- processing stages.

(i) resource configuration, which consist of details of the resource where the application will be executed such as the resource name, the runtime, and the username and account details used to access the resource, and (ii) kernel configuration, which defines the workload parameters such as the location of input files for the MD simulation and analysis tools, parameters for the tools, and workflow parameters such as the number of simulations.

IV. APPLICATIONS

We illustrate the capabilities of the ExTASY approach via two exemplar applications. The two different advanced sampling algorithms implemented with ExTASY are the Diffusion Map-directed-MD (DM-d-MD) and CoCo-MD techniques. Both these algorithms have a common execution pattern: an ensemble of simulation tasks followed by an analysis stage, performed for multiple iterations following the pattern shown in Figure 2.

In the case of the DM-d-MD algorithm, the simulation stage uses Gromacs and LSDMap for the analysis stage. Whereas in the CoCo algorithm, the simulation stage consists of Gromacs runs / trajectory conversions and the analysis uses CoCo. The individual simulation or analysis tools might differ depending on the algorithm chosen but the overall pattern is observed to be the same.

As ExTASY wraps for existing software components and implements established algorithms, we make no effort here to assess correctness of the output. Indeed for performance testing we have used a single iteration, which is insufficient to produce reliable, well converged sampling.

A. Diffusion Map-directed-MD

The Diffusion Map-directed-MD (DM-d-MD) technique [13] improves the efficiency of computational resources by choosing which replicas of the protein are used to run MD. When replicas are too close to each other, the MD trajectories will be similar. The information gain from simulating MD with close replicas is small. Part of the replicas which are too

close to each other are deleted. To hold the total number of replicas constant, replicas which are too far apart from each other are duplicated. In DM-d-MD, a non-linear dimensionality reduction technique, the locally scaled diffusion map (LSDMap) [14] is used to calculate the distance between different replicas. The deletion or duplication of replicas would destroy the correct sampling of the protein. By changing the weights of individual replicas in the reweighting step, the correct sampling of the protein is obtained.

The DM-d-MD technique requires only the protein starting structure. No additional information about the protein is necessary. The user can fine tune the sampling mainly by varying the total number of replicas and the way how the local scale in LSDMap is calculated. At the begin on the method, the replicas are generated from the protein starting structure. After the MD step, the LSDMap is calculated. LSDMap requires only the final structure for each replica from the MD step. Based on the LSDMap results new replicas for the next iteration of DM-d-MD are chosen from the current replicas. The reweighting ensures that the

It was shown that DM-d-MD technique is, at least, one order of magnitude faster compared to plain MD [13]. This comparison was done for alanine dipeptide and a 12-aminoacid model system, Ala12.

B. The CoCo-MD workflow

The CoCo (**C**omplementary **C**oordinates) technique [31] was designed originally as a method to enhance the diversity of ensembles of molecular structures of the type produced by NMR structure determination. The method involves the use of PCA [32]–[34] in Cartesian space to map the distribution of the ensemble in a low (typically 2-4 dimensional) space, and then the identification of un-sampled regions. CoCo generates new conformations for the molecule that would correspond to these un-sampled regions. The number of new structures generated is under the users control the algorithm divides the space into bins at a chosen resolution, marks bins as sampled or not, first returns a structure corresponding to the centre of the un-sampled bin furthest from any sampled one, marks this bin as now sampled, and iterates as many times as desired.

In the CoCo-MD workflow, an ensemble of structures from MD simulations are analysed using the CoCo method; new conformations become the start points for a new round of MD simulations. The latest MD data is added to the previous set, and CoCo repeated. The method is agglomerative - all MD data generated so far is used for each analysis; but also adaptive - a fresh PCA is performed each time. Applied to simulations of the alanine pentapeptide, the CoCo-MD workflow is able to reduce mean first passage times from the extended state to other local minimum states by factors of ten or greater compared to conventional simulations [35].

V. PERFORMANCE EVALUATION

A. Experiment setup

1) *Physical system:* The 39-residue mixed α/β protein NTL9(1-39) (pdb code 2HBA, 14,100 atoms including water)

is chosen as the physical system for our experiments. NTL9 has an experimentally measured folding time of around 1.5 ms [36], and its folding process has been extensively studied by experiment and all-atom MD simulations, both by means of the Folding@Home distributed computing platform coupled with MSM analysis [9], and by Anton supercomputer [37].

The relatively small size of NTL9, and the existence of previous MD simulation results over long timescales, make this protein an ideal candidate for testing and benchmarking our approach. Albeit small, NTL9 is much larger than a simple peptide, and exhibits a folding process with two competing routes [37], thus presenting a non-trivial test for adaptive sampling.

2) *HPC systems used:* One of the requirements of ExTASY as that it should be interoperable, so we have used several different HPC systems for our experiments, and characterised the performance of ExTASY on each.

Stampede is a Dell Linux cluster located at the Texas Advanced Computing Center, and is part of the Extreme Science and Engineering Discovery Environment (XSEDE). It consists of 6400 compute nodes, each with 2 Intel Xeon ‘Sandy Bridge’ processors, for a total of 16 CPU cores per node, as well as an Intel Xeon Phi co-processor (not used in our experiments). Stampede uses the SLURM batch scheduler for job submission.

ARCHER is a Cray XC30 supercomputer hosted by EPCC, and operated on behalf of the Engineering and Physical Sciences Research Council (EPSRC) and the Natural Environment Research Council (NERC). It has 4920 compute nodes, each with 2 Intel Xeon ‘Ivy Bridge’ processors, giving 24 cores per node. ARCHER uses the Portable Batch System (PBS) for job submission.

Blue Waters is a Cray XE6/XK7 operated by the National Centre for Supercomputing Applications on behalf of the National Science Foundation and the University of Illinois. The XE6 partition used in this work consists of 22640 compute nodes with 2 AMD ‘Interlagos’ processors, giving 32 cores per node. Blue Waters uses the TORQUE/Moab workload manager for job submission.

B. Evaluation of individual components

Since the performance of the entire workflow depends on the performance of each of the component parts, we investigate the scaling of both the simulation code (Gromacs) and the analysis tools in isolation on each of the three target platforms, using the NTL9 system used for the full sampling workflows.

1) *Simulation tools:* The parallel efficiency of Gromacs with respect to a single core on each machine is shown in Figure 3. While efficiencies of 69% (ARCHER, 24 cores), 78% (Stampede, 16 cores) and 46% (Blue Waters, 32 cores) suggest that while the scaling for such a relatively small simulation is not ideal, using a single node per simulation is a good use of the available hardware. Beyond a single node, the efficiency drops off so although multiple node simulation tasks are supported by Ensemble Toolkit they are not useful for this benchmark case.

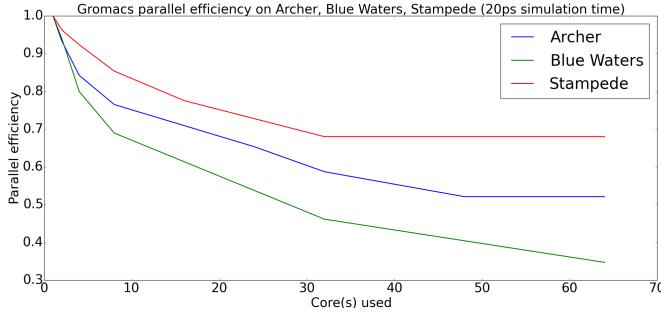


Fig. 3. Gromacs parallel efficiency on ARCHER, Blue Waters and Stampede. A single 20ps gromacs simulation of the NTL9 system is performed using various core counts on the three machines and the execution time is measured.

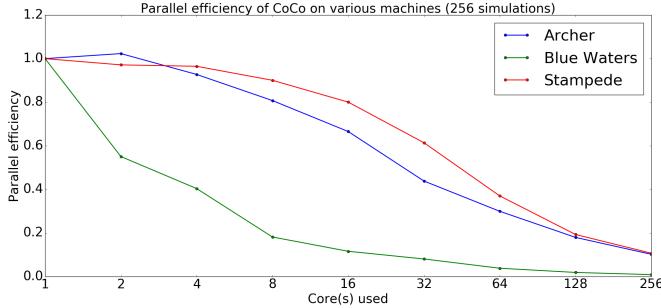


Fig. 4. Parallel efficiency of CoCo on ARCHER, Blue Waters and Stampede. A total of 256 simulations are analyzed using various core counts on the three machines and execution time is measured.

2) *Analysis tools:* Due to the nature of the two workflows, there are many parallel simulation tasks, but only a single analysis task. Therefore, the analysis task may be configured to run on as many cores as are available to the simulations. Both CoCo and LSDMap are parallelised using MPI, and consist of parts which are independent, e.g., reading of trajectory files in CoCo, and involve communication e.g. diagonalisation of the covariance matrix in CoCo and the diffusion matrix in LSDMap, so the parallel scaling is expected to be sub-linear. The performance of CoCo is also strongly dependent on I/O since it reads the entire trajectory file rather than just the final configurations like LSDMap.

Figures 4 show the strong scaling of CoCo for a fixed input of 256 simulations. We see that CoCo is able to scale to at least 256 cores on ARCHER, Blue Waters and Stampede, thus for our following experiments we configure the workflow to run CoCo with as many cores as there are input trajectories. LSDMap (Figure 5) however, does not scale efficiently much beyond a single node on each machine, even with over 2000 input structures. Nevertheless, we run LSDMap on as many cores as are available, even though it cannot use them fully. Due to the structure of the workflow, those cores would otherwise sit idle during the analysis phase.

C. Evaluation of ExTASY

1) *Characterization of overheads:* In addition to the necessity of characterizing performance overhead introduced by a new system approach and implementation, in order to instill confidence in potential users of ExTASY it is important to

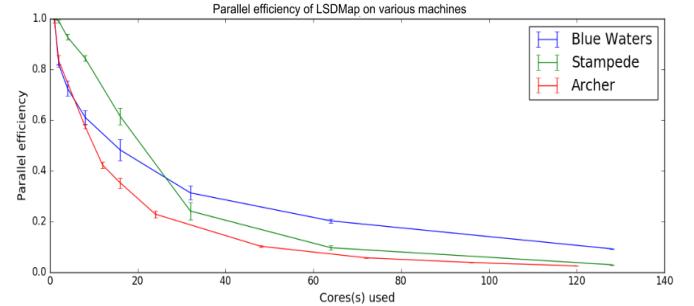


Fig. 5. Parallel efficiency of LSDMap on ARCHER, Blue Waters and Stampede. A total of 2449 structures are analyzed using various core counts on the three machines and execution time is measured.

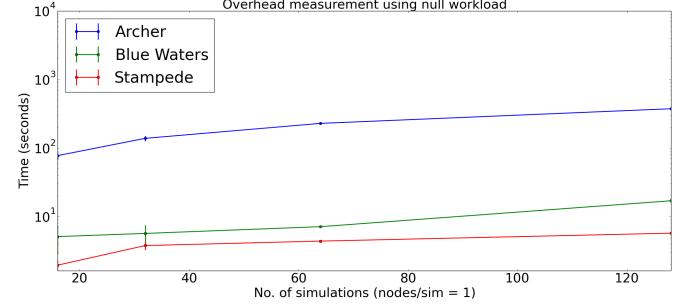


Fig. 6. Measurement of overhead from ExTASY on ARCHER, Blue Waters and Stampede. Using only the simulation stage of the SAL pattern, overhead from ExTASY is measured on the three machines at various core counts.

measure the overheads imposed by ExTASY. The objective is to discern the contributions from the different aspects of the ExTASY framework as opposed to MD and analysis components. The time taken to create, launch and terminate all of the simulations or the instantiate ExTASY framework itself on the resource are examples of the former overhead. All of the experiments use Ensemble Toolkit version 0.3.14.

We ran a single iteration of the workflow with null workloads, i.e., where the task did no work (`/bin/sleep 0`), but otherwise was configured as a “normal” simulation task, launched using MPI and taking a whole node on each of the machines. The number of tasks ranged from 16 to 128, and they were all run concurrently. Figure 6 shows that the overheads on Stampede and Blue Waters are relatively small, growing from <5s to around 15s for 128 tasks. On ARCHER the overheads are much larger (70–350s), which after further investigation is due to the `aprun` job launcher, which performs poorly when multiple concurrent tasks are launched.

2) *Strong scaling test:* To test the strong scaling of the ExTASY workflows, we fix the number of executing instances (independent MD simulations) to 128 and vary the total number of CPU cores used to execute the workflow. The largest experiments use enough CPU cores that all of the MD simulations execute concurrently. For example, on ARCHER, 128 instances, each executing on a single node (24 cores) gives a maximum of 3072 cores.

Since the MD simulations are independent and may be executed concurrently by ExTASY, as the number of cores is

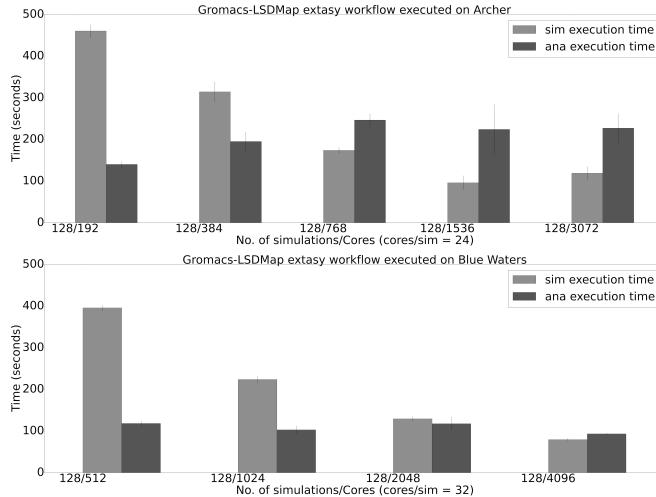


Fig. 7. Strong scaling of DM-d-MD workflow on ARCHER (**top**) and Blue Waters (**bottom**). The number of simulations is held constant at 128, number of cores per simulation at 24 on ARCHER and 32 on Bluewaters. The total number of cores used is varied with a constant workload, hence measuring strong scaling performance of the framework.

increased, the simulation time should decrease proportionally. The time spent in the analysis part is expected to be constant since the total amount of MD data is constant, and despite the parallelisation of the analysis tools shown in Figure 5, the time to completion plateaus at fairly low core counts.

Figure 7 (**bottom**) shows the results of this experiment for the DM-d-MD workflow executed on Blue Waters. The simulation time decreases from 395.7s on 512 cores to 79.52s on 4096 cores, a speedup of 4.97x with 8x as many cores – yielding a scaling efficiency of 62%. The analysis time is essentially constant at around 100s, as expected. The loss of scaling efficiency for the simulation part comes from two sources. Firstly, there is the fixed overhead discussed in Section V-C1 associated with the execution of 128 concurrent tasks, which is approximately 15s. Secondly, the actual computation which occurs within each task take longer when more simulations are run concurrently, due to the fact that they all write to the same shared filesystem. For example, when 16 instances are run concurrently on 512 cores, the MD simulations take an average of 45.6s each. When all 128 instances are run concurrently, each takes 49.0s, or 3.4s slower. If these effects are removed, the effective scaling efficiency on 4096 cores rises to 77%.

Similar results are obtained on ARCHER (Figure 7, **top**), although the scaling of the simulation part tails off on 3072 cores, and the LSDMap analysis takes somewhat longer, with higher variability than Blue Waters. Both of these are due to the fact that both the MD and analysis involve significant I/O, and it is known that opening many small files concurrently is slow as the metadata servers of the parallel Lustre filesystem become a bottleneck [38].

CoCo-MD on Stampede (Figure 8, **bottom**) has similar strong scaling for the simulation part as DM-d-MD. The simulation time decreases from 363s on 256 cores to 83.7s on 2048 cores – a speedup of 4.3x for a 8-fold increase in the

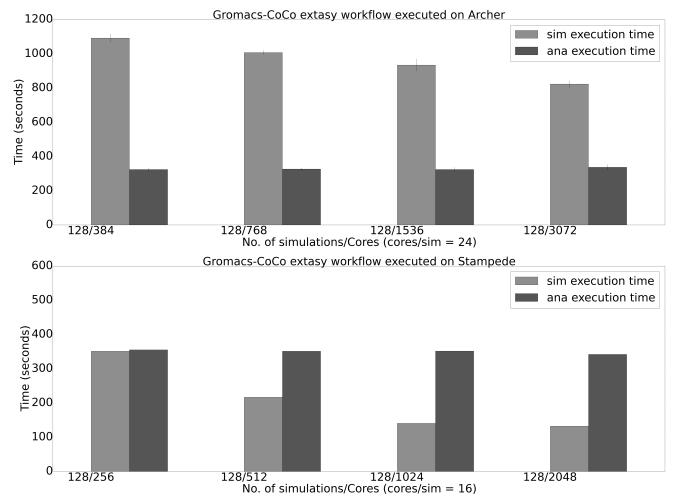


Fig. 8. Strong scaling of CoCo-MD workflow on ARCHER (**top**) and Stampede (**bottom**). The number of simulations is held constant at 128, number of cores per simulation at 24 on ARCHER and 16 on Stampede. The total number of cores used is varied with a constant workload, hence measuring strong scale performance of the framework.

number of cores (54% efficiency). However, the analysis time (CoCo) does not scale due to the fact that the parallelisation in CoCo is limited to the number of input trajectories, which is 128 in this case, even if more cores are available.

The CoCo-MD workflow on ARCHER (Figure 8, **top**) does not show as good scaling as DM-d-MD, or CoCo-MD on the other platforms. The reason for this lies in the fact that after the actual MD calculation, a ‘trajectory conversion’ step is required to prepare the data for CoCo. This step only takes a fraction of a second to execute, but there is a very large overhead caused by aprun, which allocates resources to and launches each individual task. This does not occur on Blue Waters, which uses the ORTE [39] implementation of RADICAL-Pilot that is not yet default on ARCHER.

3) *Weak scaling test*: To investigate the weak scaling properties of ExtASY, we fix the ratio of instances to CPU cores, and vary the number of instances with the constraint that all simulations can execute concurrently. For example, on ARCHER 16 instances are executed on one node each (24 cores) giving a total of 384 cores, and the number of instances is increased to 128, i.e., the number of cores is 3072.

Since all simulations run concurrently, and the length of each simulation does not change, we expect the simulation time to be a constant. However, the analysis part will increase since the performance of the analysis tools is a function both of the input data size (depending on the number of instances), as well as the number of cores available, and even though the number of cores is proportional to the data size, the amount of work grows faster than linearly with the data size.

For the DM-d-MD workflow on Blue Waters (Figure 9, **top**) we observe a small increase of 21.8s in the simulation part as we scale from 512 to 4096 cores. Similar to the strong scaling results, this is combination of the overhead due to the increased number of tasks and a slowdown of the tasks themselves. The analysis is found to increase sub-linearly. As

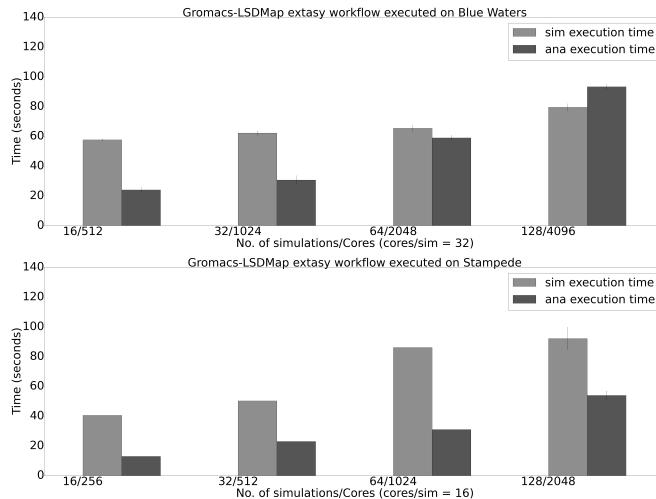


Fig. 9. Weak scaling of DM-d-MD workflow on Blue Waters (**top**) and Stampede (**bottom**). The number of cores per simulation is held constant at 32 on Blue Waters and 16 on Stampede. The total number of simulations is varied from 16-128 and the cores used are increased proportionally. By keeping the ratio of the workload to the number of resources constant, we observe the weak scaling performance of the framework.

discussed in section V-B2 the LSDMap computation consists of parts which are both linear and quadratic in the size of the input. Combined with an increasing number of cores available for the tool, sub-linear scaling is the result. Similar behaviour is observed on Stampede (Figure 9, **bottom**), although with a different weighting of the simulation and analysis parts, reflecting the fact that the performance of each kernel depends on how well optimised it is on the execution platform.

The weak-scaling of CoCo-MD on ARCHER (Figure 10, **top**) shows very clearly the aprun bottleneck discussed in Section V-C2, and the effect increases with the number of concurrent tasks. As expected, the analysis part scales better than linearly since CoCo consists of parts which weak scale ideally (independent operations per trajectory file) and parts such as the construction and diagonalisation of the covariance matrix which grow as the data size squared or more.

On Stampede, the weak scaling of the simulation part of the CoCo-MD workflow (Figure 10, **bottom**) is much better than ARCHER. The simulation time grows only by around 50s compared to over 700s on ARCHER over the range of cores that we tested. CoCo scales almost identically to ARCHER.

4) Effect of larger ensembles: To distinguish the effects caused by strong scaling (increasing parallelism with a fixed amount of work) and weak scaling (increasing parallelism proportionally to the amount of work), we also measured the effect of increasing the amount of work with a fixed number of compute cores available. Figure 11 shows the results for the DM-d-MD workflow running on Blue Waters as we vary the number of MD instances from 128 to 1024, keeping the total number of cores available at 4096. Since each task runs on a single node (32 cores per instance), only 128 simulation tasks can run concurrently. Ideally, we would expect that the simulation time should increase linearly with the number of instances. In practice, we see that the time taken grows by only

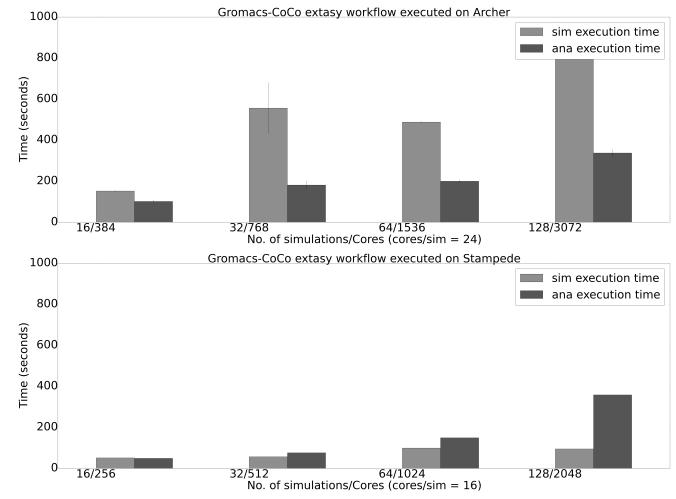


Fig. 10. Weak scaling of CoCo-MD workflow on ARCHER (**top**) and Stampede (**bottom**). The number of cores per simulation is held constant at 24 on ARCHER and 16 on Stampede. The total number of instances is varied from 16-128 and the cores used are increased proportionally. By keeping the ratio of the workload to the number of resources constant, we observe the weak scaling performance of the framework.

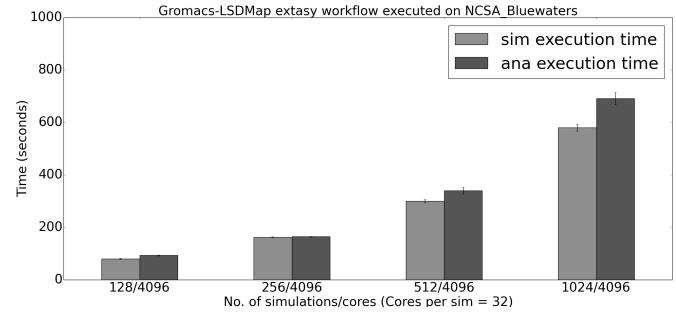


Fig. 11. DM-d-MD workflow on Stampede. Workload is increased from 128 instances to 1024, keeping the number of cores constant at 4096. Within the overheads, the increase in execution time is in proportion with increase in the workload.

7.4x between 128 and 1024 instances i.e., a factor of 8. This is due to the fact that some of the overheads related to managing the tasks that occur before or after execution in the 128 task case are one-time overheads, i.e., those overheads are hidden as they are done concurrently (in the RADICAL-Pilot Agent) with the execution of the remaining tasks when the number of instances is greater than 128. The scaling of the analysis part is consistent with that discussed in Section V-B2, that there is close to linear scaling since the larger the ensemble size, the more parallelism is available in LSDMap

5) Dynamic simulations: An important characteristic of the LSDMap and CoCo based workflows is that the number of instances typically changes after each simulation-analysis iteration. Thanks to the pilot-abstraction, the ExTASY framework supports flexible mapping between the number of concurrent instances and the total number of cores, while being agnostic of the number of cores per instance. This functionality is used by the DM-d-MD workflow, where, depending on the progress through the conformation space of the system being explored, LSDMap may decide to spawn more (or less) trajectories

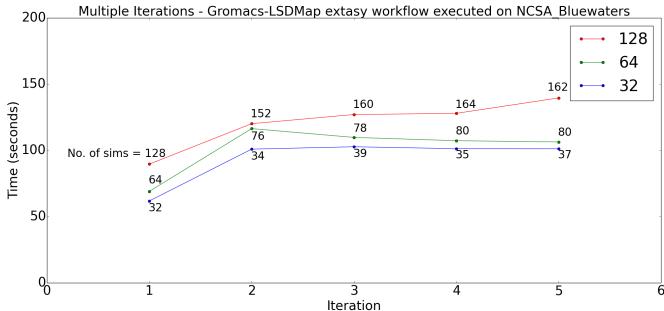


Fig. 12. Support for dynamic workload in ExTASY: The DM-d-MD algorithm dictates the number of instances at every iteration. The number of instances in each iteration (for a total of 5 iterations) when starting with 32, 64, 128 instances is presented.

for the next iteration of sampling. Figure 12 illustrates this capability. We ran the DM-d-MD workflow on Blue Waters for three configurations with 32, 64 and 128 initial instances. We can see that after an initial growth phase the number of instances seems to stabilise for the remaining iterations, although the difference from the starting configuration and the number of iterations taken to stabilise is not algorithmically or systematically predictable. The flexible resource utilization capabilities that ExTASY is built upon prove critical.

D. Summary of Experiments

We have shown illustrative performance data for two different applications – CoCo-MD and DM-d-MD – based on different analysis methods, on three distinct HPC platforms - ARCHER, Blue Waters and Stampede. The overall scaling to O(1000) simulations is clearly demonstrated, and we have analysed the scaling behaviour of the ExTASY framework itself (overheads) and the individual simulation and analysis programs which constitute the workflow. Data from similar experiments using AMBER as the simulation code can be found at <https://bitbucket.org/extasy-project/extasy-experiments>.

VI. DISCUSSION AND CONCLUSION

State-of the-art computational biophysics approaches aim to balance three different requirements: force-fields accuracy, advanced sampling capabilities, and rigorous and fast data analysis. These points are strongly interconnected. In particular, it is becoming clear that advanced sampling and data analysis need to be tightly coupled to work efficiently and accurately, as the configurational space that has already been sampled needs to be analyzed on-the-fly to inform on how to proceed with further sampling. Furthermore, many advanced sampling algorithms for biomolecular simulations require flexible and scalable support for multiple simulations. As no final solution yet exists on the best strategy for adaptive sampling (and different physical systems may require a combination of strategies), there is a need to allow combination of different MD engines with different analysis tools, and can be easily extended or modified by end-users to build their own workflows, both for development of new strategies, and for applications to the study of complex biomolecular systems.

ExTASY is designed and implemented to provide a significant step in this direction. ExTASY allows users to simulate many parallel MD trajectories (by means of standard MD engines), to extract long timescale information from the trajectories (by means of different dimensionality reduction methods), and to use the information extracted from the data for adaptively improving sampling. ExTASY has been utilized by different MD engines and analysis algorithms, with only pre-defined and localized changes

In Section III, we formally identified the functional, performance and usability requirements to support the coupling of MD simulations with advanced sampling algorithms. We then presented the design and implementation of ExTASY, an *extensible, portable* and *scalable* Python framework for building advanced sampling workflow applications to achieve these requirements. After establishing accurate estimates of the overhead of using ExTASY, Section V consisted of experiments designed to validate the design objectives; we performed experiments that characterized ExTASY along traditional scaling metrics, but also investigated ExTASY beyond single weak and strong scaling performance. With the exception of some machine specific reasons, ExTASY displayed linear scaling for both strong and weak scaling tests on various machines up to 1000s of simulation instances on up to 1000s of nodes for both DM-d-MD and CoCo-MD workflows.

In order to keep the footprint of new software small, ExTASY builds upon well-defined and understood abstractions and their efficient and interoperable implementation (Ensemble Toolkit, RADICAL-Pilot). This provides double duty: the core functionality of ExTASY is provided by simple higher level extensions of complex system software, while benefitting from the performance and optimization of the underlying system software layers. This also allows ExTASY to employ good systems engineering practice: well-defined and good base performance, while being amenable to platform-specific optimizations (e.g. using ORTE on Blue Waters [39]).

The design of ExTASY to reuse existing capabilities for extensibility to different MD codes and sampling algorithms while providing well defined functionality and performance are essential features to ensure the sustainability of ExTASY. Compared to existing software for advanced sampling, ExTASY provides a much more flexible approach that decouples the definition of the application and choice of individual tools from the details of execution on particular compute platforms, is architected to enable efficient and scalable performance and has a simple but general user interface. The ExTASY toolkit is freely available from <http://www.extasy-project.org>.

The ExTASY toolkit has been used to deliver two hands-on computational science tutorials to the biomolecular simulations community with a focus on advanced sampling. Participants were given the opportunity to utilize HPC systems in real time for advanced sampling problems of their own. Details of both events can be found at <http://extasy-project.org/events.html>. A link to the lessons and experience from the first workshop can be found at: <https://goo.gl/nMSd27>.

ACKNOWLEDGMENTS

This work was funded by the NSF SSI Awards (CHE-1265788 and CHE-1265929) and EPSRC (EP/K039490/1). This work used the ARCHER UK National Supercomputing Service (<http://www.archer.ac.uk>). We acknowledge access to XSEDE computational facilities via TG-MCB090174 and Blue Waters via NSF-1516469. We gratefully acknowledge the input from various people who have helped the development of the EXTASY workflows: everyone else involved the EXTASY project, the attendees at EXTASY tutorials and beta testing sessions, and particularly David Dotson and Gareth Shannon who provided in-depth comments and suggestions.

REFERENCES

- [1] “NSF XSEDE Annual Report (2012),” page 382, Fig 32 <https://www.xsede.org/documents/10157/169907/2012+Q2+Annual+Report.pdf>.
- [2] Using XDMoD to facilitate XSEDE operations, planning and analysis XSEDE ’13 Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery, doi 10.1145/2484762.2484763.
- [3] A. Barducci *et al.*, “Metadynamics,” *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 1, no. 5, pp. 826–843, 2011. [Online]. Available: <http://dx.doi.org/10.1002/wcms.31>
- [4] L. C. Pierce *et al.*, “Routine access to millisecond time scale events with accelerated molecular dynamics,” *Journal of Chemical Theory and Computation*, vol. 8, no. 9, pp. 2997–3002, 2012, pMID: 22984356. [Online]. Available: <http://dx.doi.org/10.1021/ct300284c>
- [5] H. Wu *et al.*, “Multi-ensemble Markov models of molecular thermodynamics and kinetics,” *Proc. Natl. Acad. Sci. USA*, vol. in press, 2016.
- [6] Y. Sugita and Y. Okamoto, “Replica-exchange molecular dynamics method for protein folding,” *Chemical Physics Letters*, vol. 314, no. 12, pp. 141 – 151, 1999. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0009261499011239>
- [7] J. D. Chodera and F. Noé, “Markov state models of biomolecular conformational dynamics,” *Current Opinion in Structural Biology*, vol. 25, pp. 135 – 144, 2014. [Online]. Available: <http://dx.doi.org/10.1016/j.sbi.2014.04.002>
- [8] F. Noé *et al.*, “Constructing the equilibrium ensemble of folding pathways from short off-equilibrium simulations,” *Proceedings of the National Academy of Sciences*, vol. 106, no. 45, pp. 19 011–19 016, 2009. [Online]. Available: <http://dx.doi.org/10.1073/pnas.0905466106>
- [9] V. A. Voelz, G. R. Bowman, K. Beauchamp, and V. S. Pande, “Molecular simulation of ab initio protein folding for a millisecond folder NTL9(1–39),” *J. Amer. Chem. Soc.*, vol. 132, no. 5, pp. 1526–1528, 2010.
- [10] I. Buch, T. Giorgino, and G. De Fabritiis, “Complete reconstruction of an enzyme-inhibitor binding process by molecular dynamics simulations,” *Proc. Natl. Acad. Sci. USA*, vol. 108, no. 25, pp. 10 184–10 189, 2011. [Online]. Available: <http://www.pnas.org/content/108/25/10184.abstract>
- [11] S. Gu *et al.*, “Quantitatively characterizing the ligand binding mechanisms of choline binding protein using markov state model analysis,” *PLoS Comput. Biol.*, vol. 10, no. 8, pp. 1–11, 08 2014. [Online]. Available: <http://dx.doi.org/10.1371%2Fjournal.pcbi.1003767>
- [12] N. Plattner and F. Noé, “Protein conformational plasticity and complex ligand-binding kinetics explored by atomistic simulations and markov models,” *Nat. Commun.*, vol. 6, 2015.
- [13] J. Preto and C. Clementi, “Fast recovery of free energy landscapes via diffusion-map-directed molecular dynamics,” *Phys. Chem. Chem. Phys.*, vol. 16, no. 36, p. 19181, 2014.
- [14] M. A. Rohrdanz *et al.*, “Determination of reaction coordinates via locally scaled diffusion map,” *J. Chem. Phys.*, vol. 134, March 2011.
- [15] X. D. Guo *et al.*, “Computational studies on self-assembled paclitaxel structures: Templates for hierarchical block copolymer assemblies and sustained drug release,” *Biomaterials*, vol. 30, no. 33, pp. 6556 – 6563, 2009. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0142961209008503>
- [16] R. Harada *et al.*, “Simple, yet powerful methodologies for conformational sampling of proteins,” *Phys. Chem. Chem. Phys.*, vol. 17, pp. 6155–6173, 2015. [Online]. Available: <http://dx.doi.org/10.1039/C4CP05262E>
- [17] J. Peng and Z. Zhang, “Simulating large-scale conformational changes of proteins by accelerating collective motions obtained from principal component analysis,” *Journal of Chemical Theory and Computation*, vol. 10, no. 8, pp. 3449–3458, 2014, pMID: 26588312. [Online]. Available: <http://dx.doi.org/10.1021/ct5000988>
- [18] D. E. Shaw *et al.*, “Anton, a special-purpose machine for molecular dynamics simulation,” *Commun. ACM*, vol. 51, no. 7, pp. 91–97, 2008.
- [19] I. Ohmura *et al.*, “MDGRAPE-4: a special-purpose computer system for molecular dynamics simulations,” *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 372, no. 2021, 2014. [Online]. Available: <http://rsta.royalsocietypublishing.org/content/372/2021/20130387>
- [20] R. Salomon-Ferrer *et al.*, “An overview of the Amber biomolecular simulation package,” *WIREs Comput. Mol. Sci.*, vol. 3, no. 2, pp. 198–210, 2013.
- [21] B. R. Brooks *et al.*, “CHARMM: A program for macromolecular energy, minimization, and dynamics calculations,” *J. Comput. Chem.*, vol. 4, no. 2, pp. 187–217, 1983.
- [22] M. J. Abraham *et al.*, “GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers,” *SoftwareX*, vol. 12, pp. 19 – 25, 2015.
- [23] S. Plimpton, “Fast parallel algorithms for short-range molecular dynamics,” *J. Comp. Phys.*, vol. 117, no. 1, pp. 1–19, 1995.
- [24] J. C. Phillips *et al.*, “Scalable molecular dynamics with NAMD,” *J. Comput. Chem.*, vol. 26, no. 16, pp. 1781–1802, 2005.
- [25] G. A. Tribello *et al.*, “PLUMED 2: New feathers for an old bird,” *Comp. Phys. Comm.*, vol. 185, no. 2, pp. 604–613, 2014.
- [26] P. Giannozzi *et al.*, “Quantum ESPRESSO: a modular and open-source software project for quantum simulations of materials,” *J. Phys. Condens. Matter*, vol. 21, no. 39, p. 395502, 2009.
- [27] S. Pronk *et al.*, “Copernicus: A new paradigm for parallel adaptive molecular dynamics,” in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC ’11. New York, NY, USA: ACM, 2011, pp. 60:1–60:10. [Online]. Available: <http://doi.acm.org/10.1145/2063384.2063465>
- [28] I. Bethune, V. Balasubramanian, and S. Jha, “ExTASY: Extensible Tools for Advanced Sampling and AnalYsis,” 2016, manuscript in preparation.
- [29] V. Balasubramanian, A. Treikalas, O. Weidner, and S. Jha, “Ensemble Toolkit: Scalable and Flexible Execution of Ensembles of Tasks,” 2016, (accepted ICPP 2016) <http://arxiv.org/abs/1602.00678>.
- [30] A. Merzky *et al.*, “Executing Dynamic and Heterogeneous Workloads on Super Computers,” 2016, <http://arxiv.org/abs/1512.08194>.
- [31] C. A. Laughton, M. Orozco, and W. Vranken, “COCO: A simple tool to enrich the representation of conformational variability in NMR structures,” *Proteins*, vol. 75, no. 1, pp. 206–216, April 2009.
- [32] I. T. Jolliffe, *Principal component analysis*. Springer Verlag, 2002.
- [33] E. C. Sherer, S. A. Harris, R. Soliva, M. Orozco, and C. A. Laughton, “Molecular dynamics studies of DNA A-tract structure and flexibility.” *J Am Chem Soc*, vol. 121, pp. 5981–5991, 1999.
- [34] S. T. Wlodek *et al.*, “Molecular dynamics of acetylcholinesterase dimer complexed with tacrine.” *J Am Chem Soc*, vol. 119, pp. 9513–9522, 1997.
- [35] A. Shkurti, E. Rosta, V. Balasubramanian, S. Jha, and C. Laughton, “CoCo-MD: A Simple and Effective Method for the Enhanced Sampling of Conformational Space,” 2016, manuscript in preparation.
- [36] J.-C. Horng, V. Moroz, and D. P. Raleigh, “Rapid cooperative two-state folding of a miniature $\alpha\beta$ protein and design of a thermostable variant,” *J. Mol. Biol.*, vol. 326, no. 4, pp. 1261–1270, 2003.
- [37] K. Lindorff-Larsen, S. Piana, R. O. Dror, and D. E. Shaw, “How fast-folding proteins fold,” *Science*, vol. 334, no. 6055, pp. 517–520, 2011.
- [38] D. Henty, A. Jackson, C. Moulinec, and V. Szerem, “Performance of Parallel IO on ARCHER,” 2015. [Online]. Available: http://archer.ac.uk/documentation/white-papers/parallelIO/ARCHER_wp_parallelIO.pdf
- [39] M. Santcroos *et al.*, “Executing dynamic heterogeneous workloads on Blue Waters with RADICAL-Pilot.” *Proceedings of the Cray User Group (CUG) 2016*, 2016. [Online]. Available: <http://www2.epcc.ed.ac.uk/~ibethune/files/RP-ORTE-CUG2016.pdf>

Leveraging Burst Buffer Coordination to Prevent I/O Interference

Anthony Kougkas^{*†}, Matthieu Dorier[†], Rob Latham[†], Rob Ross[†], and Xian-He Sun^{*}

^{*}Illinois Institute of Technology, Department of Computer Science, Chicago, IL

akougkas@hawk.iit.edu, sun@iit.edu

[†]Argonne National Laboratory, Mathematics and Computer Science Division, Lemont, IL

{mdorier, robl, tross}@anl.gov

Abstract—Concurrent accesses to the shared storage resources in current HPC machines lead to severe performance degradation caused by I/O contention. In this study, we identify some key challenges to efficiently handling interleaved data accesses, and we propose a system-wide solution to optimize global performance. We implemented and tested several I/O scheduling policies, including prioritizing specific applications by leveraging burst buffers to defer the conflicting accesses from another application and/or directing the requests to different storage servers inside the parallel file system infrastructure. The results show that we mitigate the negative effects of interference and optimize the performance up to 2x depending on the selected I/O policy.

Keywords—I/O Interference; Parallel File Systems; I/O Policies; I/O Staging; Burst Buffers

I. INTRODUCTION

Large-scale applications already individually suffer from unmatched computation and storage performance, leading to a loss of efficiency in I/O-intensive phases. But another problem appears when several applications compete for access to a common parallel file system, leading to further degradation of I/O performance as a result of contention. Most modern supercomputers have moved from the paradigm of one large application using the entire machine to one where many smaller applications run concurrently. In [1], we see that half of the jobs executed on Argonne’s Intrepid machine were using less than 2048 cores (i.e. only 1.25% of the entire available cores); and since many of the same applications were ported to its successor, Mira, we suspect the pattern on this new system to be no different. Consequently, multiple applications commonly run concurrently and share the underlying storage system. This practice, however, can severely degrade the I/O bandwidth that each application experiences. This phenomenon, called *cross-application I/O interference*, stems from diverse sources: network contention at the level of each storage server, poor scheduling decisions within the storage service (i.e., parallel file system) leading to different servers servicing requests from distinct applications in a different order, or additional disk-head movements when interleaved requests from distinct applications reach the same storage device.

The use of *burst buffers* in HPC systems [2], [3] has emerged with the initial goal to relieve the bandwidth burden on parallel file systems by providing an extra layer of low-latency storage between compute and storage resources. The Cori system at the National Energy Research Scientific Computing Center (NERSC) [4], uses CRAY’s Datawarp technology [5]. The Los Alamos National Laboratory Trinity supercomputer [6] will also use burst buffers with a 3.7 PB capacity and 3.3 TB/s bandwidth. Intel has also discussed the use of burst buffer nodes under the new Fast Forward storage framework [7] for future HPC systems. One common characteristic in all these

use cases is to increase the total I/O bandwidth available to the applications and optimize the input/output operations per second (i.e., IOPS). In addition to serving as a pure storage option, the notion of a burst buffer can make storage solutions smarter and more active.

In this paper, we address the problem of cross-application I/O interference by coordinating burst buffer access to prevent such I/O degradation. We propose several ways to mitigate the effects of interference by *preventing applications from accessing the same file system resources at the same time*. We build on our previous work leveraging cross-application coordination [1] and propose three new strategies to mitigate interference. Two of these strategies are based on the use of burst buffers to delay actual accesses to the storage system when multiple applications are interfering. The third strategy dynamically partitions the parallel file system’s resources in order to dedicate distinct subsets of storage servers to each application when contention is detected. In summary, the contributions of this paper are: (i) we propose the coordination of burst buffer access to manage concurrent accesses to the underlying storage system (Section III); (ii) we design and implement several strategies to mitigate the effects of I/O interference (Section IV); (iii) and we evaluate these strategies with several microbenchmarks and show their results (Section V). Section VI presents related work, and Section VII summarizes our conclusions and briefly discusses future work.

II. BACKGROUND AND MOTIVATION

A. I/O Interference

Supercomputers generally are designed for maximum computing power to solve a small number of large, tightly-coupled and compute-intensive problems. While computing and network resources can be shared effectively by state-of-the-art job schedulers, the same cannot be said about the storage resources (i.e., shared parallel file systems). In fact, [8] and [9] suggest that I/O congestion, within and across independent jobs, is one of the main problems for future HPC machines. A significant source of performance degradation seen on the Jaguar supercomputer at Oak Ridge National Laboratory was identified as concurrent applications sharing the parallel file system [10].

The I/O performance degradation is caused by contention for resources. These resources include network hardware used for I/O requests, file system servers that are responsible for metadata operations and other I/O requests, the servers that are responsible for committing the I/O requests to the underlying storage devices, and the storage media itself, as well as virtual resources such as locks that are used to manage distributed accesses [11], [12]. In this study we focus on the file system level. The main cause of interference in this level is how

the parallel file system services I/O requests from multiple applications (i.e., the internal scheduler).

Prior work addressed this contention through storage server coordination, where the basic idea is to serve one application at a time in order to reduce the completion time and, in the meantime, maintain the server utilization and fairness [13]. However, applications still experience a bandwidth reduction since the storage resource is still shared. Other work suggested solutions in the file system scheduler [14], [15], [16]. However, those solutions need to be integrated into the parallel file system's server code. In this paper, we propose that by preventing applications from concurrently accessing the same storage resources, we can exploit the full potential of existing parallel file systems and allow the system to provide higher global I/O throughput across multiple applications.

B. Burst Buffers

Scientific applications often demonstrate bursty I/O behavior [17], [18]. Typically in HPC workloads intense, short phases of I/O activities, such as checkpointing and restart, periodically occur between longer computation phases [19], [20]. New storage system designs that incorporate non-volatile burst buffers between the main memory and the disks are of particular relevance in mitigating such burstiness of I/O [21].

Burst buffers as an intermediate storage tier located between RAM and spinning disks are designed to help scientific applications in many ways: improved application reliability through faster checkpoint-restart, accelerated I/O performance for small transfers and analysis, fast temporary space for out-of-core computations and in-transit visualization and analysis [4]. The most commonly used form of a burst buffer in current HPC systems is dedicated *burst buffer nodes* [2], [22]. These nodes can exist in the I/O forwarding layer or as a distinct subset of the computing nodes (i.e., not part of the compute resources but responsible for acting as burst buffer nodes) or even as entirely different nodes close to the processing resources (i.e., extra resources if available). A burst buffer can also be located inside the main memory of a computing node or as a fast non-volatile storage device placed in the computing node (i.e., NVRAM devices, SSD devices, PCM devices etc). Besides the above, the main functionality of burst buffers is to quickly absorb I/O requests from the computing elements and asynchronously issue them to the parallel file system (PFS), allowing the processing cores to return faster to computation. In this paper, we propose to use such burst buffers and, by coordinating them, tackle the I/O interference caused by multiple applications running concurrently in the system.

III. OUR APPROACH

Burst buffers are placed between the application and the parallel file system layer. Thus they are naturally suitable to act as *I/O traffic controllers* and prevent applications from accessing the underlying file system resources at the same time. Burst buffers can achieve this objective by making a certain application stage the I/O (i.e., buffer the requests) while another one is accessing the shared parallel file system. We argue that when concurrent accesses from multiple applications are detected, we can avoid the undesirable I/O interference by dynamically changing the data distribution to the PFS, specifically by using burst buffers to direct I/O traffic to the underlying resources in a nonconflicting manner.

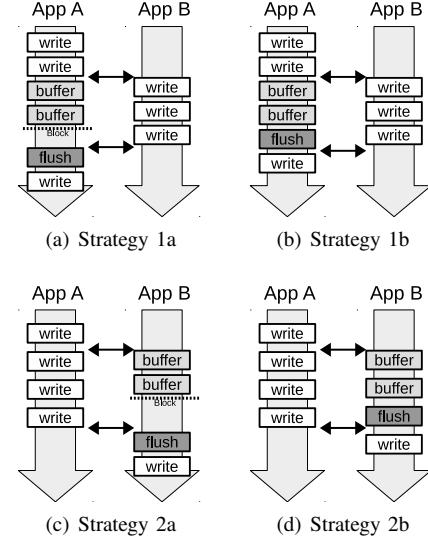


Fig. 1: Buffer-based coordination strategies. Black arrows correspond to cross-application communications that allow them to know when to switch their buffering system on or off.

By coordinating burst buffers, we can provide the much needed global, system-wide view of running applications with concurrent accesses to the underlying parallel file system and prevent I/O interference in this level by employing the following policies.

A. I/O Staging Policies

Previously proposed strategies, which consist of simply blocking one application for the benefit of the other (*first come - first served* order or *interruption* of the running application), have the disadvantage of completely blocking one application while it could actually perform some computation. As an example, if the I/O phase consists of compressing and writing chunks of data (as implemented in HDF5), the application could compress multiple chunks and stage them when there is contention, instead of blocking on a write operation. Another example is in the case of collective I/O [23] (in particular two-phase I/O) where instead of blocking on the first write, multiple rounds of communication could be completed before writing is performed.

We introduce two strategies based on I/O staging to prevent or mitigate I/O interference. Figure 1 demonstrates our approach for the cases where an application has some other options for making progress instead of waiting for another application to complete its I/O operations. Using the burst buffers, an application waiting for access to the file system can actually *stage* its I/O operations locally and execute them later, that is, block only when closing the file or forcing a flush.

The goal for these strategies is to prevent applications from accessing the underlying storage system at the same time while allowing them to do something else and not wait for the resource being blocked. Considering two applications *A* and *B*, where *A* starts its I/O phase before *B*, we hold the following assumptions: (a) applications notify each other in a timely manner about their respective I/O intentions (e.g., entering an I/O phase); (b) applications, when instructed to start staging their I/O, have some other work to perform (e.g.,

some computation); and (c) while an application is staging, its I/O consists of write-only phases. We propose two strategies with two variations each as follows.

Strategy 1a: Application *A* starts staging its operations as *B* enters an I/O-intensive phase. It blocks if necessary and flushes only after *B* has completed its I/O phase. It continues writing after flushing if its operations are not completed. This strategy is shown in Figure 1(a).

Strategy 1b: Application *A* starts staging its operations as *B* enters an I/O-intensive phase. It then flushes its buffer when it has no more available work even if *B* has not completed its operations yet. This strategy is shown in Figure 1(b).

Strategy 2a: As *B* starts its I/O phase, it learns that *A* is already accessing the file system and therefore starts staging its I/O requests. It blocks if necessary and flushes only after *A* has completed its I/O phase. It continues writing after flushing if its I/O operations are not completed. This strategy is shown in Figure 1(c).

Strategy 2b: When *B* starts its I/O phase, it discovers that application *A* is already accessing the file system; therefore, it starts staging its I/O requests. It flushes the buffers as soon as it has no more available work, even if *A* has not completed its I/O phase. It continues writing if its I/O operations are not completed. This strategy is shown in Figure 1(d).

We propose these strategies having in mind different classes of applications where the I/O phase might be time sensitive (i.e., they cannot wait for some other application to finish) or where more bursty behavior is present in one of them. The two different approaches in each strategy, *stage and block* or *stage and flush*, offer greater flexibility to the system in order to execute concurrent applications and get the maximum I/O throughput from the PFS.

B. Dynamic Partitioning of PFS

The dynamic partitioning strategy involves partitioning in space rather than in time. Figure 2 illustrates our approach. We shift from both applications accessing all available servers to partitioning the PFS into distinct subsets and directing each application’s requests to different sets of storage servers. The intuition behind the benefits of this strategy is that the performance of a parallel file system usually does not scale linearly with the number of storage servers accessed by an application. Preventing applications from accessing the same set of servers will prevent interference at the level of storage servers and their disks, leaving only the network as a potential source of contention, however a reduced set of storage servers will offer a lower bandwidth than will the full set. We distinguish two variations of this strategy.

Strategy 3a: We define a static, predefined partitioning where both applications access different storage servers from the beginning until the end of their execution. Even though fewer storage servers can offer less bandwidth to the application, the prevention of I/O interference in the file system level (i.e., exclusive access to the disks) might be enough to outperform the use of the entire PFS installation. Splitting the servers into disjoint sets can be performed according to several criteria; it depends on the available knowledge of the application’s I/O needs, scale, priority, and so forth. In this study, we explored proportional sharing of the available storage servers according to the dataset size (i.e., total amount of

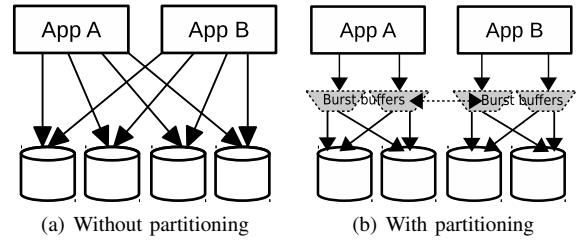


Fig. 2: Partition-based strategy. Instead of accessing all the storage servers, applications communicate and agree to interact with nonconflicting subsets of servers.

data) and the application size (i.e., number of MPI processes). For instance, for 8 available storage servers; consider that application A writes 48 MB per process and B writes 16 MB. For the same number of processes, application A would write on 6 servers and B on the remaining 2.

Strategy 3b: The second variant is dynamic partitioning of the file system when interference is detected. Burst buffers can make intelligent decisions; and, when appropriate (i.e., when multiple applications try to access the file system at the same time), they redirect I/O into separate and distinct subsets of storage servers for each application. When no contention exists, applications are exposed to the entire set of available servers in order to achieve maximum bandwidth. This strategy is appropriate only for write workloads, since read workloads are tied to the set of servers where the required data is stored. In our previous example, consider that application A starts writing to all 8 servers; when B starts an I/O phase, A directs all its I/O requests to 6 storage servers and B writes to the remaining 2. After B finishes, A goes back to writing on all 8 servers.

IV. DESIGN AND IMPLEMENTATION

In this section we present our design and implementation for our proposed solution and how we enable those coordination strategies through our library.

A. Design Overview

To evaluate the various strategies presented in Section III, we implemented a userspace buffering system, called BBIO (Basic Buffered I/O)¹ working under the POSIX and LibC interfaces. BBIO is a library comprising two parts. Its static part *libbbio.a* can be linked to any code and provides the user-level interface to initialize and control buffers. The dynamic part *libbbio_posix.so* can be preloaded to replace existing POSIX and LibC functions (such as `write` and `fwrite`). Hence, the application will buffer its I/O only if the dynamic library is preloaded, making the use of BBIO as simple as setting an environment variable.

The FILE structure provided by the standard C library already provides a local memory buffer. Although the user can control the size of this buffer, the user has no control over *when* the program will decide to flush it. In contrast, BBIO allows the user to control both the size of the buffer allocated to a file where this buffer is located and the moment the buffer can be flushed. It can also leverage local storage devices such as SSD instead of relying on local memory.

¹Our implementation is available at <https://bitbucket.org/mdorier/bbio>.

B. Interface and API

Our BBIO library presents the following interface to application developers.

- **BBIO_Init(const char* path, size_t size)**: initializes BBIO, giving it the path to a directory where it can write buffered data (path to an SSD, for instance). Leaving this path NULL instructs BBIO to use RAM as storage. The size provided is the maximum size allowed for a buffer associated with any single file descriptor. Whenever the buffer is filled or if a write is issued with a size that cannot fit the buffer, the buffer automatically resizes if there is available space or is flushed.
- **BBIO_Enable(int fd)**: enables buffering for a particular file descriptor. By default, buffering is enabled for all files outside of system directories.
- **BBIO_Disable(int fd)**: disables buffering for a particular file descriptor. If buffering was previously enabled and some data have been put in the buffer, the buffer is flushed.
- **BBIO_Flush(int fd)**: forces a flush on the buffer associated with the file descriptor.
- **BBIO_Finalize()**: finalizes BBIO. It will flush all the buffers currently managed.
- **BBIO_On_flush(int fd, BBIO_Callback cb)**: installs a callback function that will be called before any flush (whether this flush is triggered manually by BBIO_Flush, BBIO_Disable, or BBIO_Finalize or automatically when the buffer is full). The callback will not be called when trying to flush an empty buffer. The BBIO_Callback object must have the signature `void ()(int fd)`.

Using this interface lets us implement various interference-avoiding strategies based on cross-application communication. For example, using BBIO_On_flush can detect some other application's I/O traffic, thus preventing the application from flushing its buffer while another application is accessing the PFS, and wait for the file system to be available again.

C. Implementation Details

When BBIO captures a POSIX or a LibC function call, it first checks whether if the referenced file descriptor has a buffer associated with it. Such a buffer is either an mmap-ed file in a local disk or an anonymous memory segment. The buffer associated with a file is *not* a local copy of this file. Instead, the local file is a log of the operations to be performed on a file. For example, a write operation will add an operation code identifying it, followed by the size of the write and then a copy of the data. If several contiguous writes are issued, BBIO combines them by updating the size of the first one and appending the data of subsequent ones. Because of this log-structured implementation, BBIO is not yet able to work with files accessed both in read and write modes. BBIO will still associate a buffer to files opened in both read and write modes but will disable it if read operations are issued.

While most parallel file systems provide some ways to control the distribution policy across servers (stripe size and number of servers to stripe across), they usually do not provide a coordinated way to specify which servers should be used among multiple applications. To simulate such a possibility, we deployed separate PVFS instances on different sets of storage servers each, and let each application access its own PVFS instance.

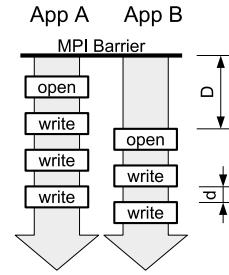


Fig. 3: Overview of our microbenchmark.

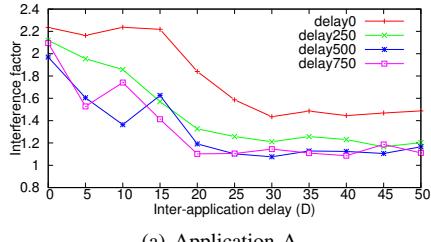
V. EXPERIMENTAL AND EVALUATION RESULTS

A. Methodology

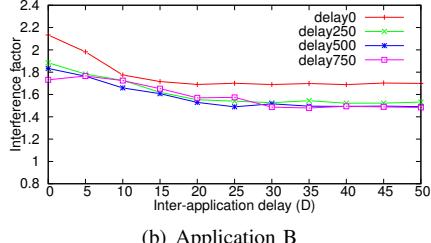
Platform description: All experiments were carried out on a 65-node SUN Fire Linux-based cluster at the Illinois Institute of Technology. The computing nodes are Sun Fire X2200 servers, each with dual 2.3 GHz Opteron quad-core processors and 8 GB of main memory. A 250 GB hard drive and an additional PCI-E X4 100 GB SSD are attached to each node as the storage devices. All 65 nodes are connected with Gigabit Ethernet. The network topology of the cluster consists of three groups of nodes connected to three distinct network switches with adequate capacity (i.e., 22 nodes on a router of 25 Gbits/sec) and a master node. We ran a series of network benchmarks to investigate the network's capabilities, and we found that the Gigabit Ethernet is sufficient to support our experiments without being a bottleneck. A subset of these compute nodes on network switch 1 was used to deploy a PVFS file system [24], and all client processes were dispatched on switches 2 and 3 simulating a supercomputer infrastructure with a separate PFS (i.e., all requests to servers go through the inter-rack network and not through the faster intra-rack connections).

Software used: The operating system is Ubuntu Server Edition 12.04, the parallel file system installed is OrangeFS v2.9.2, and the data distribution policy chosen is the “simple distribution” [25]. We compiled our code using gcc compiler version 4.8. The MPI implementation is MPICH 3.1.4. We developed an IOR-like microbenchmark that starts by splitting its set of processes into two separate sets running on different sets of compute nodes, representing two distinct applications. Each process writes a series of N requests of size S contiguously in a file, using POSIX fwrite calls. Between each request, the processes wait (sleep) a given delay d representing computation that could occur between requests. The second group of processes waits a specified amount of time D before starting its own series of I/O operations while the first set begins performing I/O. Both applications have a delay d between each write request. Figure 3 summarizes the behavior of our microbenchmark. The coordination between applications in our benchmark is done by using MPI communication. For example, in strategy 1, to know when B starts its I/O phase, A posts a nonblocking barrier before beginning its series of I/O operations. It then checks (`MPI_Test`) for completion of this barrier before each write. When B starts its I/O phase, it posts the matching nonblocking barrier, allowing A to know that B has started its I/O.

Measurements: For each group of processes acting as a distinct application, we wrapped all I/O operations between



(a) Application A



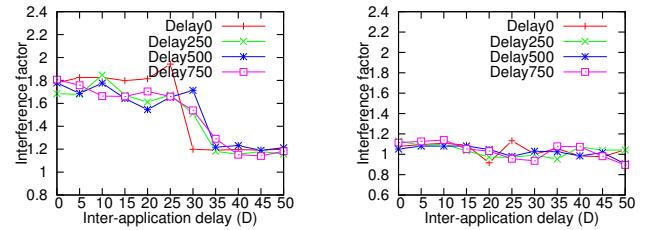
(b) Application B

Fig. 4: Default case with no strategy applied. Interference factor observed by each application as a function of the delay D (sec) and for different values of the inter-request delay d (ms).

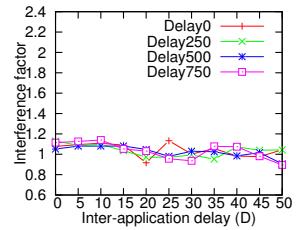
timing calls (i.e., MPI_Wtime), and we calculated the total time spent in I/O. We repeated all experiments five times, and we report the average values. We also leveraged the tools introduced in our previous work [1]: Δ -graphs are plots of a given performance metric as a function of the interapplication delay D . We consider only positive values of D (i.e., application B starts its I/O phase *after* application A). We also present the results in terms of an *interference factor*, which is a slowdown with respect to the application running without contention (i.e., an interference factor of 2 means that the duration of the write phase has been multiplied by 2). The interference factor thus is defined as $I_f = \frac{I/O\ time_{with\ interference}}{I/O\ time_{alone}}$. We always measure the duration of an I/O phase as the time between the moment the file is opened and the moment it is closed; thus the interrequest delay $(N - 1) \times d$ is counted in this duration, but the interapplication delay D is not.

B. Experimental Results

1) *Default case with interference (no strategies)*: We first quantify the interference encountered by the applications for different values of d and as a function of D . In this set of experiments, two groups of 256 processes write a series of 32 blocks of 1 MB in individual files, leading to 8 GB of output per application. We do not implement any coordination strategy; instead, we let the two applications interfere. Figure 4 shows the observed interference factor for applications A (a) and B (b). We first observe that when a delay is introduced between I/O requests, the interference factor experienced by each application is lower than when there is no delay at all. A second observation is that although the two applications are identical, the interference factor in both is often larger than 2 (up to 2.3 here) even when the two I/O phases do not start at the same time. This shows that the slowdown produced by interference is larger than what we would expect from a proportional sharing of resources, thus motivating our strategies. We also observe that when the applications start at the same time, the slowdown is significant and cannot be



(a) Application A



(b) Application B

Fig. 5: Strategy 1a. Interference factor observed by each application as a function of the delay D (sec).

ignored; on the other hand, when the inter-application delay is larger than 25-30 seconds, the interference factor reduces because the first application is almost finished when the second starts accessing the PFS.

2) *I/O staging-based strategies*: We evaluated our two I/O staging-based strategies with their two respective variations presented in Section III-A using the same configuration as above.

Results with Strategy 1a: Figure 5 presents the results where A buffers when B starts writing. Application A flushes and continues only after B is finished. This strategy is advantageous to B, whose observed interference factor is around 1 (no slowdown) since it enjoys exclusive access to the PFS. From A's perspective, staging its requests instead of interfering with B leads to a lower interference factor, between 1.2 and 1.8, instead of 1.2 to 2.2 in the default case with interference. For B there is no slowdown (since it is prioritized), and for A the interference factor is lower because burst buffers allow it to continue executing until it is absolutely necessary to block and wait for B to finish.

Results with Strategy 1b: Figure 6 depicts the results of the other variation, where A is buffering when it detects that B is writing; but instead of blocking waiting for B to finish, it flushes and continues. The intuition for this variation of the strategy is that we could tolerate some interference to happen in both applications. It lies somewhere between the previous variation of the same strategy and the pure blocking of A until B finishes. For A the behavior is similar to that described earlier, with the difference that interference is less apparent as D increases, because A is allowed to flush the buffer and need not block waiting for B to finish. Specifically for interapplication delay less than 30 seconds, the interference factor for A is kept around 1.5 (lower than the default but slightly lower than strategy 1a) and drops to 1.2 after that. Application B experiences an interference factor of around 1.2, slightly higher than before since the flushing of A happens earlier and forces B to lose the exclusive access to the PFS.

Results with Strategy 2a: This strategy is similar to Strategy 1a but with priority reversed. Here, A writes having exclusive access to the PFS, and B upon entering its I/O phase starts staging the requests until it blocks waiting for A to finish. Application B flushes the buffers and continues writing only after A has finished its I/O operations. Figure 7 shows the results. As expected, A demonstrates an interference factor close to 1, while B starts with around 1.5-1.6 for interapplication delay 0 seconds (i.e., when both applications

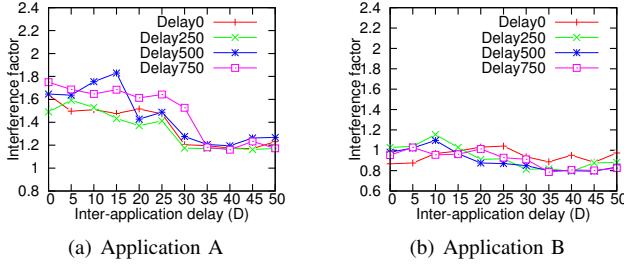


Fig. 6: Strategy 1b. Interference factor observed by each application as a function of the delay D (sec), when A buffers its operations and flushes even if B has not finished.

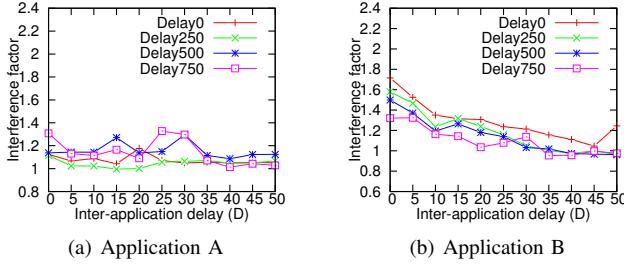


Fig. 7: Strategy 2a. Interference factor observed by each application as a function of the delay D (sec).

start at the same time) and drops close to 1 for delays larger than 30 seconds, approximately the time needed to complete the I/O phase.

Results with Strategy 2b: As with Strategy 1b, in this case B stages its operations when it enters its I/O phase because A is writing, however B flushes the buffers even if application A is not finished yet. Figure 8 demonstrates the results for both applications. Application A performs stably with interference factor around 1.2 for all values of D . Application B, on the other hand, when it starts at the same time as A, observes an interference factor of 1.6, which gradually decreases as delay D increases. One observation is that this decrease comes with higher interapplication D than Strategy 2a has, because the flushing of the buffers, before A has finished, increases the interference for both applications.

3) Partition-based Strategies: To evaluate the partitioning strategy, we conducted three tests with four configurations each as follows. The first test considers two applications A and B, identical in scale (i.e., same processes) and workload (i.e., data per process), and is referred as 50-50% in the figures. The proportional split of 8 available storage servers is in half for this test case. The second test considers two applications with the same scale, but A writes three times more data per process than does B, and is referred as 75-25% D in the figures. In the third test A has three times more processes than does B, but writes the same data per process and is referred as 75-25% S in the figures. For those two test cases the proportional split is 6 servers for A and the remaining 2 for B. As we presented in Section III-B, we have two cases for the partition-based strategy; static and dynamic partitioning of the PFS, referred to as Strategy 3a and 3b, respectively.

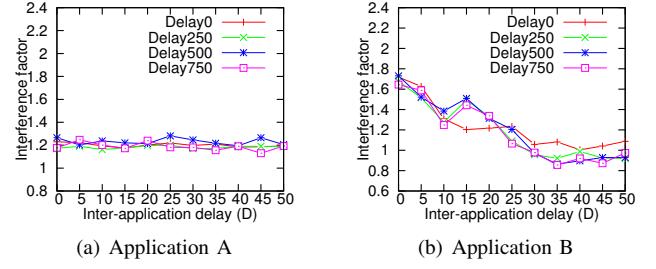


Fig. 8: Strategy 2b. Interference factor observed by each application as a function of the delay D (sec), when B buffers its operations and flushes even if A has not finished.

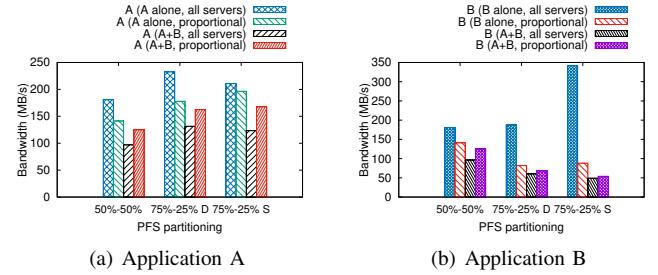


Fig. 9: Strategy 3a. Throughput observed by the applications when writing alone and in contention to all available servers or to proportional number of servers.

Results with Strategy 3a (static partitioning): The results are reported in Figure 9. We notice that when dividing by 2 the number of servers that an application accesses, the throughput observed by the application is decreased to about 75% of the throughput observed with all the servers, instead of 50%. When two applications concurrently access all 8 servers however, the slowdown is more than 2 \times . By partitioning the file system and letting each application access its own set of servers, we achieve an expected result: the application's throughput lies somewhere between the throughput observed with contention and the throughput observed with proportional servers for an individual application. This shows that by partitioning the file system so that concurrent applications do not access the same set of servers, we are able to mitigate the I/O interference and increase the I/O performance up to 40% relative to the default case where applications interfere while sharing the same storage resources. Note that each application, from the beginning till the end of its execution, has exclusive access to a distinct subset of servers.

Results with Strategy 3b (dynamic partitioning): In this experiment, we modified our benchmark to simulate the dynamic partitioning of the parallel file system's storage servers. There are three different installations of PVFS: one full deployment on all 8 storage servers, and two deployments with the proportional split. For instance, for the identical applications scenario where we split the PFS in half, two new deployments of PVFS on 4 storage servers each were used. Application A opens two files, one on the full and one on the proportional installation of the PVFS. Application B opens a file on the other proportional deployment. When A starts executing, it uses the

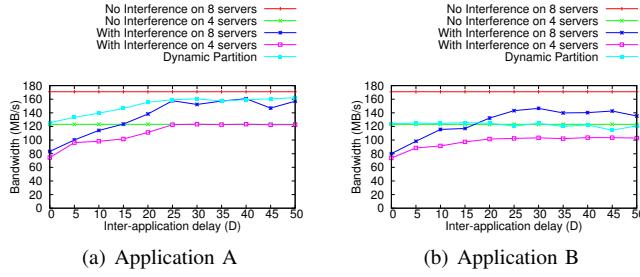


Fig. 10: Strategy 3b. Throughput observed by the applications when writing alone and in contention to all available servers or to proportional number of servers.

full installation on 8 servers. Upon the arrival of B, A redirects all its conflicted I/O requests to the proportional deployment. Thus no interference occurs since the applications are accessing a different set of servers. Once B finishes, A goes back to using all 8 servers.

Figure 10 demonstrates this dynamic partitioning strategy. The results are expressed in terms of bandwidth (MB/s). We can see that when both applications start at the same time in the case of interfering (i.e., blue and purple lines), the performance is around 70-80 MB/s, and it increases as the interapplication delay D increases (i.e., interference is less). After 30 seconds of delay D , there is no more interference since A has finished its operations and the bandwidth reaches maximum values. The dynamic partitioning strategy offers application A a bandwidth of 123 MB/s at delay 0, an increase of 75% with respect to the default case with interference. Bandwidth continues to increase with D since A is taking advantage of a full 8-server PFS deployment until it is forced, when B enters its I/O phase, to use the smaller 4-server PFS resource. On the other hand, B experiences stable performance since it always operates on a separate set of 4 servers, representing applications with smaller I/O bursts. We conducted the same dynamic partitioning strategy for the case 75-25% on data size and job size, and the results are similar. Because of space limitation, however, we do not present the results here. We note that, this partitioning strategy should be selected based on information on the applications' scale, I/O patterns, and the scalability of the file system.

4) Real Scientific Applications Workloads: To evaluate our strategies with real workloads we used three scientific applications, CM1 [26], Anonymous Application 1 and 2 by Los Alamos National Lab (referred to as LANL_App1 and LANL_App2) [27]. These applications are real-world codes that run on current supercomputers (for instance CM1 has been used on NCSI's Kraken and NCSA's BlueWaters). All of them use the POSIX I/O interface and follow the one-file per process logic for the write operations. Their workload comprises mostly by the check-pointing behavior they exhibit (i.e., periodically write their progress to the disk).

For the experiments in this paper, we created a workload generator that takes the I/O trace as an input and *replays* all the operations to the parallel file system. The applications are run on 256 cores on the same testbed. Since the I/O behavior is repetitive, we isolated the I/O access pattern from the traces for only one checkpoint phase and fed this to our workload generator to study the I/O interference. This means that for

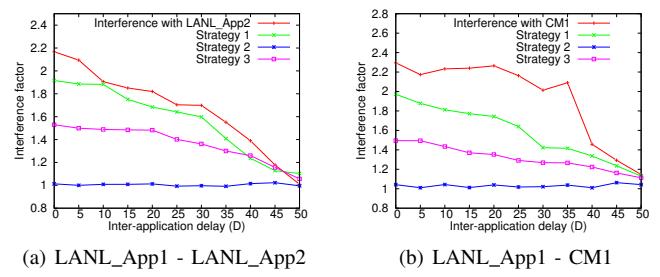


Fig. 11: Interference factor observed by LANL_App1 when interfering with LANL_App2 and CM1 as a function of the delay D (sec).

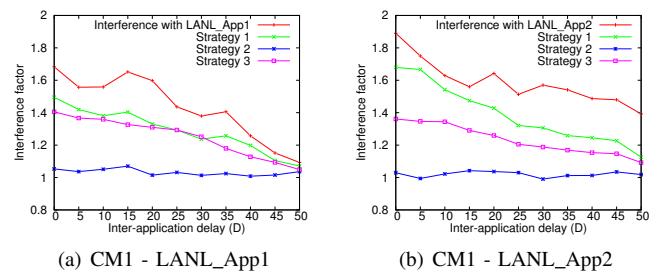


Fig. 12: Interference factor observed by CM1 when interfering with LANL_App1 and LANL_App2 as a function of the delay D (sec).

CM1 each process writes 52 MB, and for LANL_App1 and LANL_App2 each process writes 50 MB. The rest of the tests are similar to the ones presented previously.

Figure 11 shows the slowdown experienced by LANL_App1 (expressed in interference factor). In figure 11(a), when LANL_App1 executes at the same time with LANL_App2 the execution time is 2.2 times higher compared to when the applications execute with no interference. Strategy 1 prioritizes LANL_App2 but also allows LANL_App1 to lower the interference factor to 1.9. On the other hand, strategy 2 offers LANL_App1 exclusive access to the PFS thus there is no slowdown at all. Finally, strategy 3 forces the two concurrent apps to share the PFS in half and the results are similar as before. Note that we used dynamic partitioning for this test. In figure 11(b), LANL_App1 interferes with CM1 and the results are similar.

In Figure 12 (a) and (b), we see the results for CM1. All three strategies help alleviate the negative impact of the I/O interference in the performance of the application. We notice a slightly lower interference factor when CM1 runs at the same time with LANL_App1 and LANL_App2 mostly due to the different access patterns. However, the trend is the same and by using our proposed strategies one can mitigate the performance slowdown.

C. Scaling of Our Strategies

In this subsection, we first present how concurrent applications affect each others performance. By increasing the number of concurrent instances, we see that the execution time increases dramatically. In the following tests, we run

multiple instances of the same applications with the exact same parameters to investigate the relationship between the number of concurrent instances and the increase in the overall execution time (an interference factor of 3 means three times higher execution time). We used our benchmark and the three scientific applications to perform the following test. We varied the number of concurrent instances and measured the overall execution time. Due to the limitations of our testbed, we used 80 processes per instance (320 processes in total when running four instances). For example, we first ran one instance of our benchmark and measured the execution time. We then ran it again but this time interfering with one exact same instance, and then with two same instances and so on. In figure 13, we see the results. The slowdown is significant even with only two concurrent instances with interference factor around 2.4 and the situation gets worse with three and four where the interference factor reached 3.9. This shows that there is great potential to optimize performance by mitigating the I/O interference between concurrent applications.

We achieved this performance optimization by using our proposed solution. We can utilize our strategies one by one as is, or we can use a combination of all three strategies to achieve even better results. We distinguish some different scenarios. For strategies 1 and 2 burst buffers are heavily used to buffer I/O requests, and thus there might be a situation where all of the burst buffers would want to flush the data at the same time. This would create a new contention to the PFS and the benefits of using the burst buffers would be possibly eliminated. To alleviate this issue, we propose three heuristics to coordinate the flushing of the buffers. The first one is *token-based* where each app flushes the data from its burst buffers to the PFS if it holds the token. Upon completion, it simply passes the token to the next application. The second heuristic we propose is based on a *time-window*. In order to avoid "starving" an application by waiting for the token, we allocate a time-window in a round robin fashion to each application that can use it to flush data to the PFS. The third heuristic is a *priority-based* flushing of the buffers. This allows the system administration to assign the priority according to the needs. Thus, BBIO offers, through coordination of the burst buffers, a solution to the interference problem. Our third strategy is able to entirely avoid contention by efficiently sharing the available storage servers but it cannot scale to a large number of concurrent applications. A combination of all three strategies is likely to offer an overall efficient solution.

Consider the following scenario. App A is starting and then App B joins the system. We can utilize strategy 3 and solve the interference by sharing the servers. After a while, App C is also joining. Instead of further dividing the available storage servers we use either one of the other two strategies. Thus, App C uses the burst buffers to divert the I/O traffic from the PFS. When App A finishes, storage servers become available to App C again which flushes the buffers and continues normally.

In Figure 14 we report the execution time for all applications that run concurrently. The delay between applications was set to 0 and we employed our strategies in a first-come first-server fashion. Strategy 1 and 2 are somewhat mirroring the effect of prioritizing a certain application. For strategy 3, App A initially shares the 8 available servers with App B and then App B share its 4 servers with App C in half. Since the applications are exactly the same in terms of size and workload and due the limitation of space we do not present results from

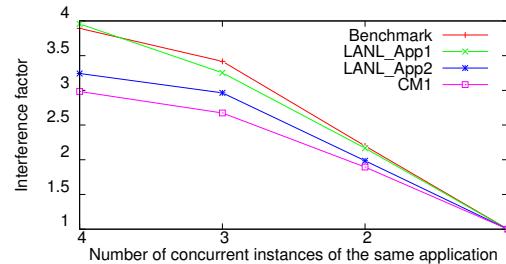


Fig. 13: Slowdown expressed in interference factor due to concurrent instances of the same application.

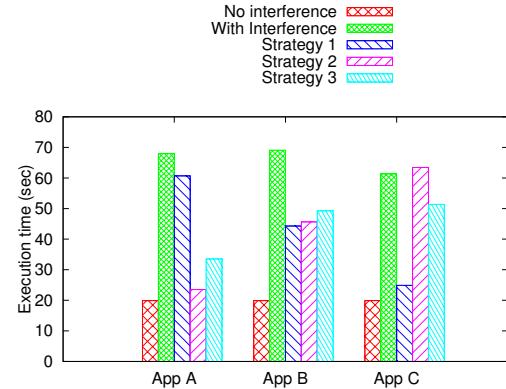


Fig. 14: Execution time (in seconds) for App A, App B and App C when running concurrently with each other. All strategies are activated in a FCFS manner.

all the proposed heuristics on when to flush the buffers.

D. Benefits and Limitations of Buffering

During our experiments we noticed an effect taking place in certain test cases. Figure 15 shows the performance of Application A for the following test: A is writing with a 750 ms delay between each I/O request (simulating a computation/communication pattern), and B is writing with no delays in between each I/O request. As a base case we include the test where both applications are writing with 0 delay between each I/O operation. When A is running alone with delay 0 it achieves 170 MB/s bandwidth and for delay 750 ms, 162 MB/s. When interfering with B, those values become 75 MB/s and 90 MB/s, respectively. By turning on the buffering system, we would intuitively expect a boost in performance. For delay 0, A achieves 96 MB/s bandwidth, which is 20% better compared with the interference case with buffering off. For delay 750 ms however, application A experiences a bandwidth of 87 MB/s, which is slightly lower than that with buffering off. Additionally, the interference factor between delay 0 is higher than the case with delay 750 ms.

This behavior is further examined and analyzed as follows. We investigated the internal behavior of every process inside each application. We focus on one application. The test comprises our microbenchmark where each process is writing 1 MB of data 32 times with a delay between each I/O operation. In Figure 16 we can see the duration of each I/O request for some randomly selected processes during our tests. When

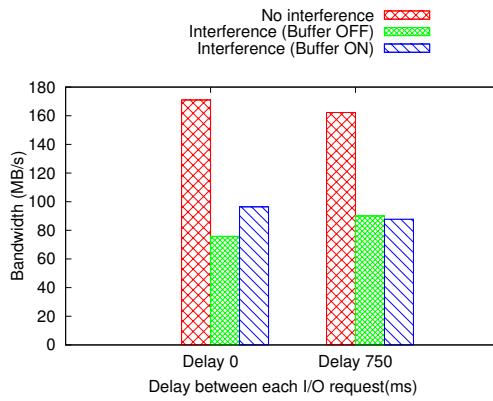


Fig. 15: Throughput observed within one application when buffering is turned on and off. Different workloads representing an I/O burst phase (delay 0) and a computation-I/O intensive alternating (delay 750).

all processes are writing at the same time with no delay between each I/O request, we see that the duration of those I/O operations is longer because of the interference. As the delay between each I/O request increases (i.e. 250 ms, 500 ms, 750 ms), we see a decrease on the duration of each request, a *self-stabilizing* effect where the computation/communication phase of one process might allow the I/O phase of another one to complete faster. Thus, *turning on the buffer and flushing it at some later point make all processes of a single application interfere internally with one another*. Hence, this solution is worse than allowing the interference with a second application in the first place. Therefore, counterintuitively, staging the I/O of one application to allow another one to exclusively access the shared storage resource and then flushing all the buffers from the delayed application at the same time might actually hurt the overall performance and highly depends on the access patterns the applications have.

VI. RELATED WORK

Many research studies have tried to address the I/O interference issue by scheduling I/O requests at the PFS level. In the network request scheduler presented by Qian et al. [28], requests embed deadlines and the targeted object’s identifier. Song et al. [13] achieve the same result with applications’ ids instead of objects’ ids. AGIOS, proposed by Boiteau et al. [15], also guides the file system’s scheduler’s decision through additional information that future I/O requests predicted thanks to traces. Zhang et al. [12] leverage a “reuse distance” to state whether it is worthwhile for a data server to wait for an application’s new I/O request or to service other applications requests. Lebre et al. [14] provide multi-applications scheduling with the goal of better aggregating and reordering requests, while trying to maintain fairness across applications. Gainaru et al. [16] propose scheduling techniques to optimize the system’s I/O efficiency under congestion. In [29], an efficient distributed message queue was used to increase the latency and minimize the interference in the network.

Closer to our work, Batsakis et al. [30] propose a system in which clients price their nonblocking requests depending on the ability to delay the requests, and an auction mechanism chooses which requests should be serviced first. In a way, the ability to

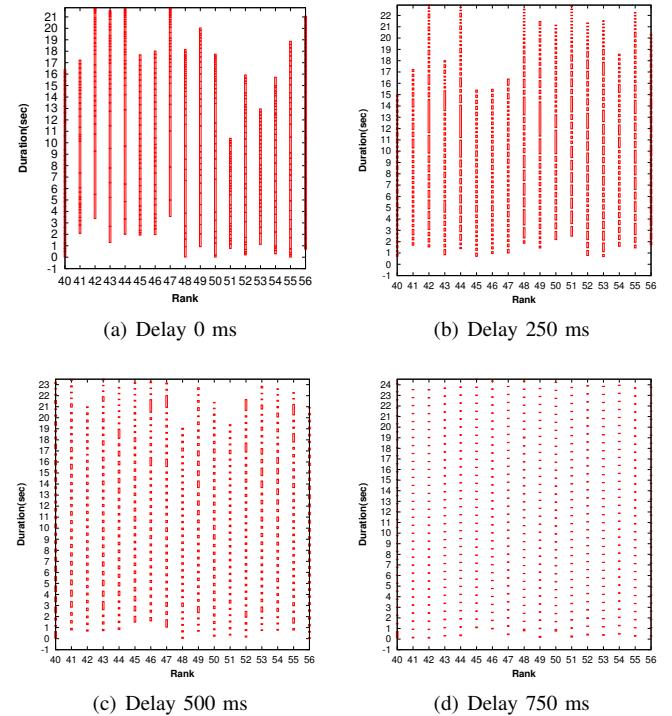


Fig. 16: Timelines per process with delays between each I/O request.

delay a request is also present in our proposed Strategies 1 and 2, where requests can be staged in a burst buffer to allow the computation to continue. But our proposed strategies work with any type of request, whereas theirs is tied to nonblocking requests only. Tanimura et al. [31] propose to reserve throughput from the storage system. This approach can be compared with our third strategy based on file system partitioning. However, this reservation is made at job submission in their approach, whereas ours leverages communications between applications in order to understand when and how the file system’s resources should be partitioned.

In a recently published work [32], an I/O orchestration framework named TRIO is proposed that also coordinates the timing when burst buffers are moving the checkpointing data to the PFS. By controlling the flushing orders among concurrent burst buffers TRIO tries to alleviate the contention on storage servers. However, it does not consider the application’s I/O access patterns, and an alternating computation-I/O behavior is not taken advantage of. TRIO focuses more on when the burst buffers will spill the data to the PFS, whereas we leverage the existence of burst buffers to act as a *traffic controller* and prevent I/O interference while allowing applications to do other *usefull* tasks. We also provide a comprehensive set of strategies to mitigate the negative effects of I/O interference, and we propose other ways to use burst buffer coordination (dynamic partitioning of PFS).

VII. CONCLUSION

Cross-application I/O interference is becoming an important issue as we move toward exascale. This issue has initiated unconventional approaches in which independent applications

can learn each other's behavior and coordinate their accesses to the shared, parallel file system. In this paper, we have proposed three strategies that applications can employ to better coordinate their accesses. Two rely on burst buffers and on the fact that instead of blocking, applications can stage their I/O requests and reissue them later, when the file system is more available. The third strategy ensures that the applications access distinct sets of storage servers. We have shown the potential of our three strategies with a microbenchmark; the results show performance improvements up to 2x.

As future work, we plan to move from the cross-application coordination scheme, where applications communicate their I/O behavior to each other, to a system wide coordination scheme where a global, centralized entity is responsible for managing all concurrent applications' accesses to the shared underlying storage resources using our strategies. In this direction, we plan to equip this entity with I/O prediction capabilities, using the OmnisceIO [33] approach, and thus select the best coordination strategy.

ACKNOWLEDGMENT

This material was based upon work supported by the U.S. Department of Energy, Office of Science, under Contract No. DE-AC02-06CH11357.

REFERENCES

- [1] M. Dorier, G. Antoniu, R. Ross, D. Kimpe, and S. Ibrahim, "CALCioM: Mitigating I/O interference in HPC systems through cross-application coordination," in *Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS '14)*, 2014.
- [2] G. Grider, "Exascale FSIO," <https://institute.lanl.gov/hec-fsio/conferences/2010/presentations/day1/Grider-HECFsIO-2010-ExascaleEconomics.pdf>, LANL.
- [3] N. Liu, J. Cope, P. Carns, C. Carothers, R. Ross, G. Grider, A. Crume, and C. Maltzahn, "On the role of burst buffers in leadership-class storage systems," in *Mass Storage Systems and Technologies (MSST), 2012 IEEE 28th Symposium on*. IEEE, 2012, pp. 1–11.
- [4] "NERSC's Cori burst buffers," <https://www.nersc.gov/users/computational-systems/cori/burst-buffer/>.
- [5] "CRAY's datawarp technology," <http://www.cray.com/sites/default/files/resources/CrayXC40-DataWarp.pdf>.
- [6] "LANL's Trinity specs," <http://www.lanl.gov/projects/trinity/specifications.php>.
- [7] Intel, "Extreme-Scale Computing R&D Fast Forward Storage and I/O Final Report," Intel, The HDF Group, CRAY, EMC, Tech. Rep., June 2014.
- [8] Y. Hashimoto and K. Aida, "Evaluation of Performance Degradation in HPC Applications with VM Consolidation," in *Networking and Computing (ICNC), 2012 Third International Conference on*. IEEE, 2012, pp. 273–277.
- [9] J. Lofstead and R. Ross, "Insights for Exascale IO APIs from Building a Petascale IO API," in *High Performance Computing, Networking, Storage and Analysis (SC), 2013 International Conference for*. IEEE, 2013, pp. 1–12.
- [10] B. Xie, J. Chase, D. Dill, O. Drokin, S. Klasky, S. Oral, and N. Podhorszki, "Characterizing Output Bottlenecks in a Supercomputer," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society Press, 2012, p. 8.
- [11] J. Lofstead, F. Zheng, Q. Liu, S. Klasky, R. Oldfield, T. Kordenbrock, K. Schwan, and M. Wolf, "Managing variability in the io performance of petascale storage systems," in *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society, 2010, pp. 1–12.
- [12] X. Zhang, K. Davis, and S. Jiang, "IOrchestrator: improving the performance of multi-node I/O Systems via inter-server coordination," in *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, 2010.
- [13] H. Song, Y. Yin, X.-H. Sun, R. Thakur, and S. Lang, "Server-Side I/O Coordination for Parallel File Systems," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2011, p. 17.
- [14] A. Lebre, G. Huard, Y. Denneulin, and P. Sowa, "I/O Scheduling Service for Multi-Application Clusters," in *IEEE International Conference on Cluster Computing*, Sept 2006.
- [15] F. Zanon Boito, R. Kassick, P. Navaux, and Y. Denneulin, "AGIOS: Application-Guided I/O Scheduling for Parallel File Systems," in *Proceedings of the International Conference on Parallel and Distributed Systems (ICPADS '13)*, Dec 2013.
- [16] A. Gainaru, G. Aupy, A. Benoit, F. Cappello, Y. Robert, and M. Snir, "Scheduling the I/O of HPC Applications Under Congestion," in *2015 IEEE International Parallel and Distributed Processing Symposium (IPDPS 2015), Hyderabad, India, May 25-29, 2015*, 2015.
- [17] N. Mi, A. Riska, Q. Zhang, E. Smirni, and E. Riedel, "Efficient management of idleness in storage systems," *ACM Transactions on Storage (TOS)*, vol. 5, no. 2, p. 4, 2009.
- [18] Y. Kim, R. Gunasekaran, G. M. Shipman, D. Dill, Z. Zhang, B. W. Settlemyer *et al.*, "Workload characterization of a leadership class storage cluster," in *Petascale Data Storage Workshop (PDSW), 2010 5th*. IEEE, 2010, pp. 1–5.
- [19] P. Carns, K. Harms, W. Allcock, C. Bacon, S. Lang, R. Latham, and R. Ross, "Understanding and improving computational science storage access through continuous characterization," *ACM Transactions on Storage (TOS)*, vol. 7, no. 3, p. 8, 2011.
- [20] "Leadership Computing Requirements for Computational Science," <https://www.olcf.ornl.gov/wp-content/%20uploads/2010/03/ORNL%20TM-2007%2044.pdf>.
- [21] "Large Memory Appliance/Burst Buffers Use Case," https://asc.llnl.gov/CORAL-benchmarks/Large_memory_use_cases_llnl.pdf.
- [22] D. Brown, P. Messina, D. Keyes, J. Morrison, R. Lucas, J. Shalf, P. Beckman, R. Brightwell, A. Geist, J. Vetter *et al.*, "Scientific grand challenges: Crosscutting technologies for computing at the exascale," *Office of Science, US Department of Energy, February*, pp. 2–4, 2010.
- [23] R. Thakur, W. Gropp, and E. Lusk, "Data sieving and collective i/o in romio," in *Frontiers of Massively Parallel Computation, 1999. Frontiers '99. The Seventh Symposium on the*, 1999.
- [24] R. B. Ross, R. Thakur *et al.*, "PVFS: A Parallel File System for Linux Clusters," in *Proceedings of the 4th annual Linux Showcase and Conference*, 2000.
- [25] "PVFS2 data distribution schemes," <http://www.orangefs.org/trac/orangefs/wiki/Distributions>.
- [26] G. Bryan, "CM1 code," <http://www2.mmm.ucar.edu/people/bryan/cm1/>.
- [27] "LANL anonymous applications trace files," <http://institutes.lanl.gov/plfs/maps/>, LANL.
- [28] Y. Qian, E. Barton, T. Wang, N. Puntambekar, and A. Dilger, "A Novel Network Request Scheduler for a Large Scale Storage System," *Computer Science - Research and Development*, vol. 23, 2009.
- [29] I. Sadooghi, K. Wang, D. Patel, D. Zhao, T. Li, S. Srivastava, and I. Raicu, "Fabriq: Leveraging distributed hash tables towards distributed publish-subscribe message queues," in *2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC)*. IEEE, 2015, pp. 11–20.
- [30] A. Batsakis, R. Burns, A. Kanevsky, J. Lentini, and T. Talpey, "CA-NFS: a Congestion-Aware Network File System," in *Proceedings of the 7th conference on File and storage technologies*, ser. FAST '09. Berkeley, CA, USA: USENIX Association, 2009.
- [31] Y. Tanimura, R. Filgueira, I. Kojima, and M. Atkinson, "Poster: Reservation-Based I/O Performance Guarantee for MPI-IO Applications Using Shared Storage Systems," in *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion*, 2012.
- [32] T. Wang, S. Oral, M. Pritchard, B. Wang, and W. Yu, "Trio: Burst buffer based i/o orchestration," in *Cluster Computing (CLUSTER), 2015 IEEE International Conference on*. IEEE, 2015, pp. 194–203.
- [33] M. Dorier, S. Ibrahim, G. Antoniu, and R. Ross, "OmniscieIO: A Grammar-Based Approach to Spatial and Temporal I/O Patterns Prediction," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '14)*, 2014.

A fast algorithm for neutrally-buoyant Lagrangian particles in numerical ocean modeling

Renske Gelderloos*, Alexander S. Szalay†, Thomas W.N. Haine*, Gerard Lemson†

*Department of Earth and Planetary Sciences

Johns Hopkins University, Baltimore, Maryland 21218, USA

Email: {rgelder2,thomas.haine}@jhu.edu

†Department of Physics and Astronomy

Johns Hopkins University, Baltimore, Maryland 21218, USA

Email: {szalay,glemon1}@jhu.edu

Abstract—As numerical ocean simulations become more realistic, analysis of their output is increasingly time consuming. As part of a larger effort to make ocean model output more accessible for analysis, we have developed a fast particle-tracking algorithm for exploring the kinematics of the simulated flow. The algorithm is independent of operating system and fully vectorized and parallelized. Furthermore, it enables sliding of particles along solid boundaries according to the 3D along-boundary flow field. The new algorithm can easily simulate several million particle trajectories, thus opening the way for Lagrangian analysis of large-scale and multi-time scale oceanic phenomena.

I. INTRODUCTION

Numerical simulations of the ocean circulation are quickly becoming more and more realistic. As the horizontal grid spacing achievable with global coverage is currently in the order of a kilometer, the numerical simulations now accurately resemble observations from in situ stations and satellites and surpass their coverage. This evolution in ocean modeling has opened up a wealth of information on physical, chemical, and biological oceanography, yet it has become increasingly hard to extract this information because of the sheer data volume. This problem is aggravated by the fact that, while ocean model codes typically take hundreds of years of developer-time to produce, and are run on massively-parallel supercomputers, the resources available for output analysis are generally limited to small groups of scientists with modest commodity workstations.

A method of extracting information on the kinematics of the simulated flow field that is growing in popularity is finding the Lagrangian pathways of virtual neutrally-buoyant (water) particles based on the ocean model output of the velocity field and the contributions to the ocean water density. Through conditional subsampling and statistical analyses of these particle paths and properties, one can determine oceanographically important information such as the origin and fate of certain water masses, along-path transformation of their properties, mixing hot spots, and the role of bathymetry in steering currents.

Two types of particle-tracking models are currently in use for physical oceanographic applications. The first type, to which Ariane [1] and TRACMASS [2] belong, are referred to as ‘analytical’ models. They calculate streamlines within

a grid cell based on the velocities at the grid cell edges. These streamlines define the trajectories of virtual particles. The velocity field is necessarily assumed stationary over the period of time sampling, which is a major disadvantage. An advantage of this method is that it is very fast, because it exploits analytic trajectory solutions, and thus allows for the simulation of millions of trajectories.

The other particle-tracking model type interpolates numerically the 4D velocity fields to find the evolution of particle locations. The most widely-used model in deep-water physical oceanography is the Connectivity Modeling System (CMS) [3]. Originally aimed at marine-biological applications, this tool offers a suite of options for non-neutrally buoyant and active particles as well as neutrally-buoyant passive particles. The model can be run in parallel to increase computational efficiency. There are two main disadvantages in the use of this model for our purposes. First, the model only runs on Unix. This requires a certain level of expertise in Unix/Fortran environments, which is inconvenient for the point-and-click applications we have in mind. Furthermore, we intend to integrate our Lagrangian particle code with advanced database technologies in an overarching project to accelerate access to and analysis of massive ocean model solutions. For this application a Windows environment is the natural choice. Therefore, a tool that is independent of operating system is preferred. Second, although the CMS ensures that particles do not collide with solid boundaries (the sea floor), its implementation is physically unrealistic. In our code particles slide freely along the wall according to the 3D along-bathymetry flow component.

For these reasons, we are developing a new particle-tracking tool. It has been used successfully in several oceanographic water-mass tracking applications [4], [5], [6]. Originally, the algorithm was implemented in a serial mode (see Section II), which scales poorly and cannot be applied to large-scale and multi-time scale problems that require millions of particle trajectories lasting several decades using velocity field increments with hour-scale time intervals. In this contribution, we describe a new algorithm that is vectorized and parallelized and has been optimized for Input/Output (I/O) efficiency. The parallelization strategy and performance improvement are

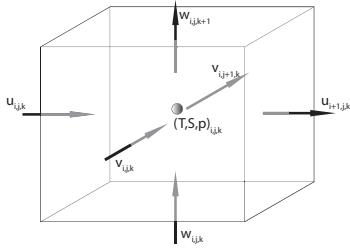


Fig. 1. The ocean model grid cell employs a finite volume discretized on an Arakawa C grid. Here, (u, v, w) are the velocity components and T, S, p are the temperature, salinity, and pressure fields for the cell with indices (i, j, k) .

discussed in Section III. Our strategy for further improvements is laid out in Section IV, and the main findings are summarized in Section V.

II. CURRENT ALGORITHM IMPLEMENTATION

The algorithm is programmed in Matlab [7], making use of its built-in Ordinary Differential Equation (ODE) solver capabilities. The algorithm loads successive sets of two sequential 3D velocity fields at a time, stored from the parent ocean circulation model, and uses the explicit Runge-Kutta (2,3)-pair ODE solver [8] for moderately stiff problems to determine a series of particle locations in the time interval between the two velocity fields (the particle trajectory). The velocity information is local to the particle position, cutout from the full vector fields. When the particle hits either a solid wall, or the edge of a grid cell, an ‘event’ is detected and the solver switches to a mode for along-boundary sliding, or loads a new local velocity-field neighborhood. The current algorithm runs on a computer cluster that was specifically designed for data-intensive science. Nevertheless, its serial implementation inhibits good scalability.

The algorithm consists of four stages:

- 1) The ocean model domain, grid, and bathymetry information are loaded and particles are seeded at their initial positions.
- 2) The space-time trajectory of each particle is computed according to the ocean model velocity fields for a predetermined length of time.
- 3) The temperature and salinity (hereafter T/S) of the particles are extracted through interpolations from the T/S ocean model output fields.
- 4) Particle positions and T/S information are saved for further analysis.

The spatial interpolation in stages 2 and 3 requires special care as velocity, T, and S fields are provided on an Arakawa C grid [9]. This means the velocity and T/S fields are staggered in space (Figure 1): Zonal (west-east) velocity components are defined on the western face of the 3-dimensional grid cell, meridional (south-north) velocity components on the southern face, vertical velocity components on the bottom face, and scalar properties such as T, S, and pressure at the cell center.

The current implementation of stage 2 is as follows. The full time period of interest is broken into discrete steps corresponding to the times of the velocity field dumps. The particle trajectories are computed between these consecutive times. For each interval, the full 3D simulation velocity cube is read into memory for both the start and the end of this interval (or they are shuffled and just the next velocity cube is read). Then a serial loop over the particles is started, which computes each particle’s trajectory using the MATLAB ODE solver. The integration starts with the current position of the particle, and then builds an array of the indices of neighboring cells ($3 \times 3 \times 2$ array). This creates a layer of one cell thick in the direction of the velocity component (see Figure 1), and 2×2 cells wide in the transverse. The code also tests for edge conditions, to make sure the particle’s location is still within this local computational domain. This index array is used to create a local neighborhood for the interpolations, which applies until the particle leaves the current cell. The MATLAB `interp1()` routine performs the linear interpolations.

The ODE solver is interrupted when the particle strikes either (I) the ocean floor, or (II) the edge of the current cell. At the cell-crossing interrupts, the index array of the local neighborhood is recomputed and the small velocity cutouts are refilled with data from the full fields. At the ocean-floor-impingement interrupts the code switches to a 2D mode, where the particles slide on a plane within each cell representing the sea bed. In this different code path the ODE solver is interrupted when the particle either (II) strikes the edge of the current cell (handled as before), or (III) it leaves the boundary. To handle the ocean-floor-departure interrupts, the fluid velocity component perpendicular to the sliding plane is computed. When it exceeds a (small positive) threshold, the particle detaches from the ocean floor, and the integration switches back to the 3D mode.

The current algorithm implements stage 2 sequentially for each particle, which was necessary to handle the small neighborhood cutouts. For earlier versions of MATLAB, the cost of `interp1` is a sensitive function of the cutout size. The sequential implementation also manages the interrupts, which occur at different, unpredictable times for each particle and the switching between the 3D and 2D code paths.

Once all particle trajectories have been determined for the entire requested time interval, the code moves to stage 3. The T/S fields are loaded successively as in stage 2 and the algorithm loops over the individual particles to interpolate their T/S properties onto their space-time trajectories.

Our test machine contains 2 XEON X5650 processors, with 12 cores, 48GB of RAM and an NVIDIA K40 GPU with 12GB memory. There are several disk volumes configured for high performance: We have four OCZ Vertex 4 SSD disks with 128GB each, and 24 1TB Samsung HDDs in the system, the latter configured as RAID0 arrays for maximum I/O performance. For most of our experiments we have placed the ocean model output fields on a 12 disk RAID0 array, which provides sequential read speeds in excess of 1GB/s. The typical CPU load during stage 2 of the current algorithm

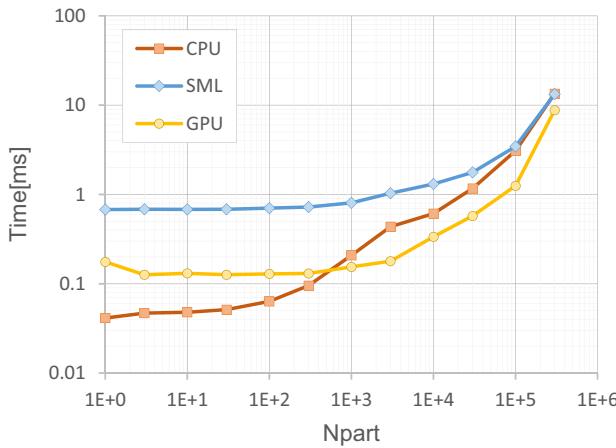


Fig. 2. Evaluation of the different interpolation strategies in MATLAB. (CPU) is using the whole domain with `griddedInterpolant()`, (SML) is using the local interpolation cutout with `interp1()`, while (GPU) shows the results of using `interp1()` using the whole domain but running on the GPU. Times are shown in msec for the number of particles on the abscissa.

is only about 8-10%. This implies that a large acceleration potentially exists for an improved algorithm.

III. PARALLEL PARTICLE TRACKING

In order to run the particle tracking in parallel, the code needed a thorough overhaul and reorganization. Given the complexity of the ODE solver interrupts, we rewrote the code incrementally to validate each new version against the previous one. At the same time, we preserved the efficient features, including: (i) switching between the 3D and 2D equations of motion, (ii) using the robust, flexible MATLAB ODE solvers for the integration, and (iii) using the efficient MATLAB interpolation routines.

A. Evaluating interpolation strategies

The first step towards parallelizing the particle integration was to see if we could abandon the local interpolation cutouts. In recent versions of MATLAB the `interp1()` routine calls the kernel function `griddedInterpolant`. However, `interp1()` is re-initialized with the data grid on every call. When the data grid is the same for many calls, this re-initialization is unnecessary and costly, and performing it once-only outside the particle loop is better. Also, the `interp1()` function has also been fully rewritten in CUDA by Mathworks. The GPU version requires that the data cube should reside in GPU memory as a `gpuArray`. There is an additional restriction: all the grid vectors must be monotonically increasing, unlike in the CPU version, which only requires monotonicity.

We ran a series of experiments to understand the performance of the MATLAB `interp1()` function. We wrote a test tool to load two consecutive model velocity fields into a 4D (540x360x216x2) array. We chose a random cell in the middle of the domain and placed N_{part} particles in it, where N_{part} varied between 1 and 10^6 . We then timed three different interpolation schemes: (CPU) using the fastest

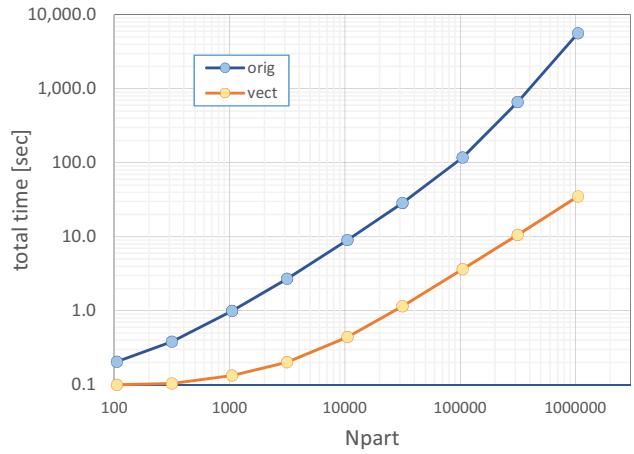


Fig. 3. The speed of the cubic interpolation determining the ocean depth at a given point from the ocean model geometry definitions. The figure shows both the serial and vectorized versions, as a function of the number of particles evaluated simultaneously. The ratio is 10 (160) for 10^3 (10^6) particles.

possible version of the interpolation over the whole cube, (SML) interpolating in a small local cutout neighborhood as in the original algorithm (in this case a 3x3x3x2), and (GPU) performing the interpolation on the GPU using the whole cube. The entire experiment was repeated 10^3 times.

The results are shown on Figure 2. It is clear that for a small number of particles (CPU) is fastest, as there are no setup costs, despite the large data cube involved. It is always faster than even the local cutout interpolation (SML). The interpolation on the GPU is slower for a small number of particles, but the performance stays flat for up to $N_{\text{part}} = 10^3$ (GPU). This indicates that the GPU version of `interp1()` is re-initializing, although faster than the CPU, and this constant overhead dominates for $N_{\text{part}} \leq 10^3$. Above a thousand particles the GPU is the preferred interpolation technique (with the data grid moved to the GPU once only). The rise in the GPU curve for $N_{\text{part}} > 10^3$ is probably due to the cost of copying the particle positions into GPU memory. In any case, for fewer than 10^3 particles we should use the CPU-based, whole cube interpolation; while for more than 10^3 particles we should switch to the GPU.

Interpolation is also used for the type (II) ocean-floor-impingement interrupts. The bathymetric depth is computed from the geometry definition, using a cubic interpolator, through the MATLAB `csapi()` spline function. This provides a function handle that can then be evaluated by passing it to the `fnval()` function. Such interpolations are calculated at every time step of the integration, for each particle. These are currently evaluated for particles one by one, so vectorization can yield potentially a large speedup. Figure 3 shows that for 10^3 (10^6) particles we achieve a factor of 10 (160) acceleration over the serial method. The cubic interpolation is necessary in order to create a smooth boundary map. Using GPUs is not currently an option, as the cubic interpolation routines only run on the CPU.

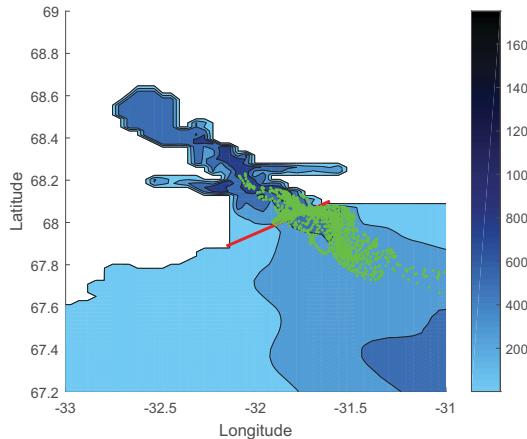


Fig. 4. The 2D (longitude, latitude) trajectories of 40 particles over 40 days in our small scale test problem. The red dots show the initial positions of the particles, placed at three different depths. The colors in the contour plot show the depth (m) of the water.

B. Incremental code conversion

The second step was to build a benchmark problem for algorithm improvement. We used a small scale case with 40 particles which enabled us to easily run tests and display the particle trajectories. The 2D trajectories are displayed on Figure 4, and in 3D on Figure 5. Plots like these were used to assess the accuracy of the trajectories computed various ways. We also built a tool to compare the particle trajectories from two codes by displaying the absolute value of the difference in each coordinate, as shown on Figure 6.

We isolated the interpolation code into a separate function, so that different options could be easily switched back and forth. In the original code, the local cutout domain was just one cell thick in one direction. The MATLAB linear `interp1()` routine returns NaN if the query coordinates fall outside the interpolation domain. The ODE solver generates an interrupt when the particle intersects the cell boundary, and to do so, it probes positions outside the cell. In order to avoid the NaN error, the original code clipped all coordinates to the edges of the local domain, causing a small error in the trajectory. Initially, we implemented the whole cube interpolation but left this edge clipping in place. The benchmark code accelerated from 130s to 90s. We estimate that in this case the I/O takes about 80s, so this corresponds to an approximately five-fold speedup for the ODE solver. The trajectories were indistinguishable.

Next, we eliminated the clipping but retained the same type (II) cell-crossing interrupt condition. The trajectories change, in some cases by as much as a few km. The change in the trajectories only occurs for particles that strike and then slide along the ocean floor. Even a small change in the first part of the trajectory results in the particle hitting the wall at a different space-time location, and this change amplifies when the particle returns to the ocean interior. We believe that interpolation over the whole domain without clipping is correct. The clipping was necessary to efficiently handle the

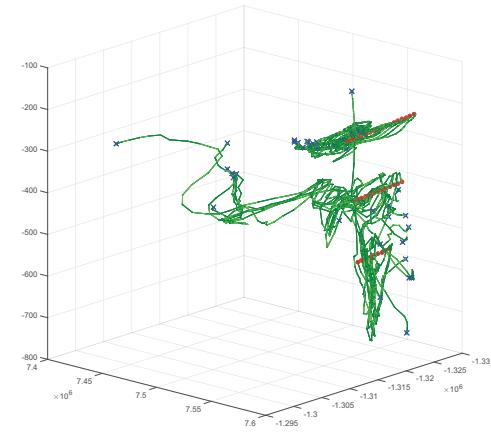


Fig. 5. The 3D trajectories of the 40 particles in the test problem. The red dots show the initial positions of the particles, placed at three different depths. The blue crosses show the final positions after 10 days. The coordinates are shown in meters. The sudden changes in directions along the trajectories are due to collisions with the boundary.

earlier versions of `interp1()`.

In the third step we removed the part (II) cell-crossing interrupts. This results in a small change in the trajectories, of the order of one km. The MATLAB ODE solver documentation recommends specifying multiple sub-steps between the starting and final times of interest. We have implemented a variable subdivision scheme, which can break the time interval into 1, 2, 4 or 8 sub-steps. Our experiments show that this has a small effect on the runtime, but that trajectories converge as the number of sub-steps increase.

Next we removed from the code all the parts related to the small cutout domains, including the incremental updates of the cell indices following each particle. This resulted in a much

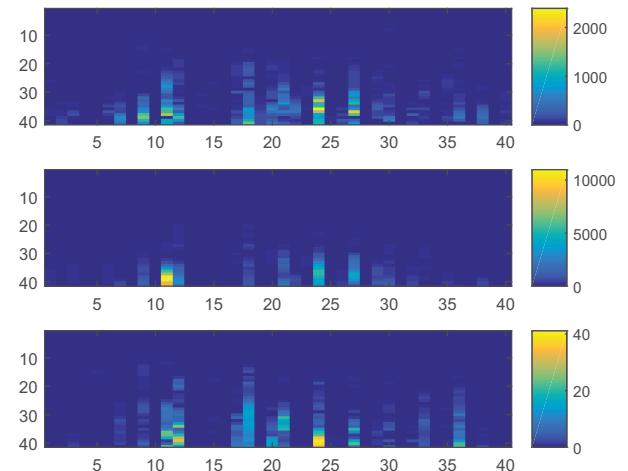


Fig. 6. The effect of eliminating the clipping of the projected coordinates at the cell boundaries in the 3D trajectories of the 40 particles. The first row shows $|\Delta x|$, the second $|\Delta y|$, and the third $|\Delta z|$, all measured in meters. The abscissa is the particle identifier and the ordinate is time. All particles start at the top. In case of a perfect match, the plot should be all blue.

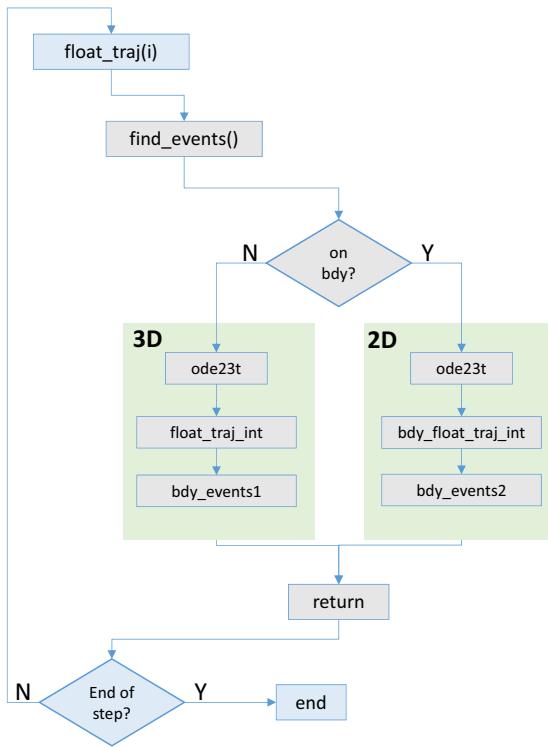


Fig. 7. Code logic in the particle integration loop for particle i . The function `float_traj(i)` is called recursively until the end of the model interval is reached. The inner function `find_events()` performs either a 3D integration (free motion) or a 2D integration as the particle slides along the sea floor, until an event occurs. The two interrupt handlers `bdy_events1()` and `bdy_events2()` catch the events when the particle either strikes or leaves the wall.

simpler code structure.

C. Vectorization: the ‘Bucket-List’ Algorithm

The third step is to vectorize the integration of particle trajectories. The type (II) and type (III) interrupts are the tricky part. A diagram of the inner loop of the serial code is shown on Figure 7. This is executed over each particle in a recursive fashion until the time of the next model output dump is reached. The function `find_events()` calls the two different ODE solvers, one for the 3D the other for the 2D motions. The ODE integration is interrupted by the type (II) and (III) events described in Section II. During the vectorization phase we must represent this logic.

We want to deploy an algorithm that handles the complex switching between the different code branches in parallel: the ‘Bucket-List’. Imagine that we start the trajectory integration (t_0) with all particles in open water, commencing 3D motion. The execution of the algorithm as a function of time is illustrated in Figure 8.

We place all these particles into a single bucket (A_1) which will be handled in parallel, using the ODE solver with a vector of $3N_{\text{part}}$ components, where N_{part} is the number of particles. The 3D integration (shown in green on the figure) continues until either the end of the interval is reached (right arrow)

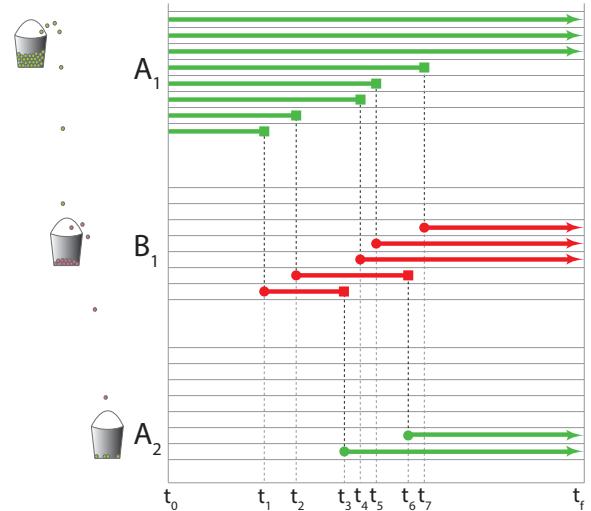


Fig. 8. The ‘Bucket-list’ parallelization strategy for the particle-tracking algorithm with event detection (striking the ocean floor). The green lines show the interior particles, moving in 3D, the red lines show the boundary particles, sliding in 2D. The small squares represent a boundary event (interrupt), a particle either strikes or leaves the wall. The dotted lines represent a particle being dropped from one bucket to the next. A (B) buckets perform 3D (2D) integration to handle the forking in Figure 7.

or an interrupt occurs because a particle strikes the wall, at time t_1 . This event is shown with a small square on Figure 8. Then the particle is removed from the bucket A_1 and dropped into bucket B_1 . Bucket B_1 collects particles that perform 2D sliding motion along the wall for the first time. Bucket A_1 continues incrementally until other particles hit the wall (at times t_2, t_4, t_5, t_7), and drop into B_1 . Finally, when no more particles hit the boundary, the integration continues to the end of the current interval (t_f).

Now consider the particles in bucket B_1 . We keep a list of its particles, sorted by their arrival times; t_1, t_2, t_4, t_5, t_7 . We use a sweep algorithm to pick up the particles one by one as the 2D integration proceeds. The ODE solver starts at t_1 , when the first dropped particle arrives. We start integrating that single particle once the second event occurs, at t_2 , setting the end of the integration to t_2 . Then the integration proceeds with two particles until the next event in the list, at t_4 .

During this process a particle may also leave the wall, at time t_3 , switching back to 3D motion. In this case, we drop the particle into a third bucket, A_2 , and continue with the remainder in B_1 until its list of particles is exhausted or the final time is reached. Particles in A_2 are handled similarly to those in B_1 , except using the 3D equation of motion. Particles striking the sea floor for a second time will drop into B_2 , and so on.

If the number of particles is too large for the ODE solver, bucket A_1 can be split and the process copied onto a second, or even a third or a fourth set of threads/batches.

Figure 9 shows an example of how the bucket-list algorithm will work using output from the benchmark calculation.

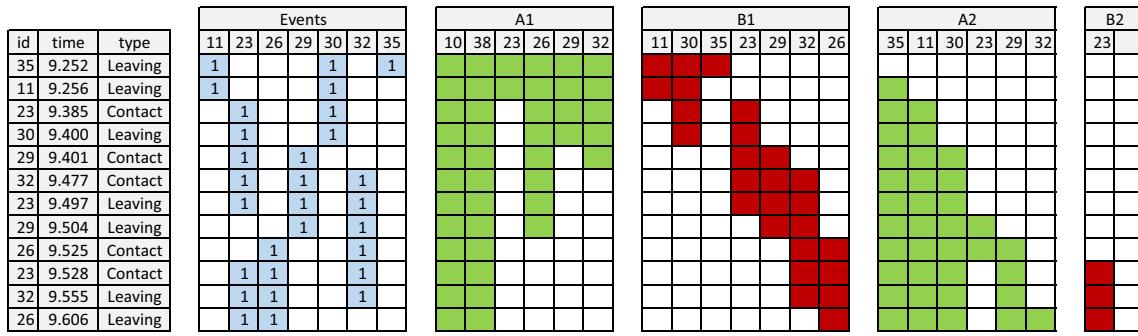


Fig. 9. Sample events in the benchmark calculation between 9.25 and 9.75 days. The figure only shows particles which are affected by an event. The subsequent panels show the steps in the different buckets (A_1, B_1, A_2, B_2). One sees particles ‘drop’ from one bucket to another and how the particles migrate through the buckets.

IV. ADDITIONAL PERFORMANCE IMPROVEMENTS

A. Parallel extraction

Stage 3 of the original code takes the particle positions, and extracts quantities like temperature and salinity along the trajectories. Here one can implement two parallelization steps. First the interpolation of these quantities can be done in a vectorized form over all particles. Second, as the different extractions are all independent, it is possible to run the extractions of different quantities in a `parfor` loop.

We performed experiments using the benchmark configuration with 40 particles to accelerate the stage 3 extraction. The letter (C) denotes ‘cold’ cache, while (W) corresponds to a ‘warm’ cache of the file system. The numbers reflect the different experiments, as follows:

- 1) C0 and W0 correspond to the original version of the code, using the I/O routine `rdmms()` written in MATLAB to read the model data, in a big endian form.
- 2) C1 and W1 use a simple binary `fread()`, after converting the model outputs to native little endian.
- 3) C2 and W2 use vectorized form of the T/S extraction, but read the model outputs sequentially with `fread()`.
- 4) C3 and W3 use an 8-way `parfor` loop in MATLAB to read the model outputs in parallel threads.

Results appear in Figure 10. These two steps altogether have speeded up the extraction by a factor of 5 for cold cache, and almost a factor of 10 for a warm cache. The speed at this point is largely limited by the disk I/O because experiments with several thousand particles yield essentially the same lapse time.

B. Improving the I/O performance

The next strategy for performance improvement is to exploit compressed data structures. Approximately half of the ocean model grid cells are below the ocean floor, varying with depth (at the surface this fraction is about 30%, while at the bottom it is almost 100%). In MATLAB one can easily address a large multi-dimensional array as a one-dimensional one. Using the elegant array indexing features of MATLAB, we can create a single vector of all the indices corresponding to the cells

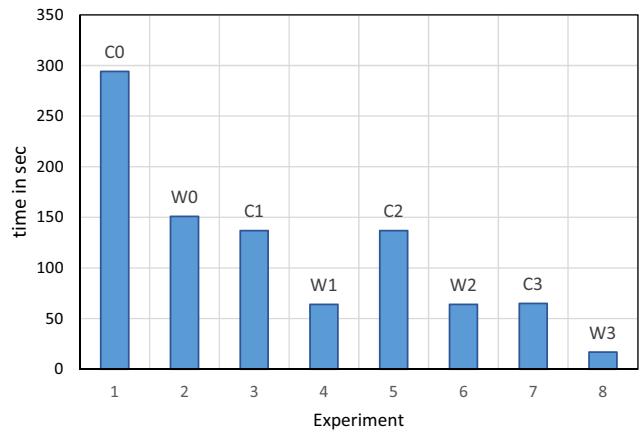


Fig. 10. Timing of the stage 3 extraction in different experiments. C (cold) and W (warm) indicate the state of the file system cache. 0 is the original code, 1 is `fread()` after native endian conversion, 2 is after vectorizing over the particles, 3 is handling the different outputs in parallel.

that are within or intersect the ocean floor. All model fields outside this set are ignored. We then store this sparse array sequentially, read it from the disk 50% faster, and copy it into place in an empty array in a single command. This approach has been tested, but not in the benchmark configuration yet.

Also, we can store the ocean model output in a database, using a cache-resilient representation, like space filling curves [10]. Several of us have developed a modular library for exactly such applications, using a plug-in architecture for various space-filling curves [11]. This enables a very fast access to subsets on various scales, while retaining a maximally sequential access pattern. We can track a bounding box of the particles and only retrieve subsets of the data as necessary, leaving the rest of the logic of the code still unchanged.

C. Increasing the precision of the integration

Until now both 3D and 2D integration paths have used the same ODE solver, with the same tolerance settings. We will explore using a higher precision method (`ode45` or `ode113`)

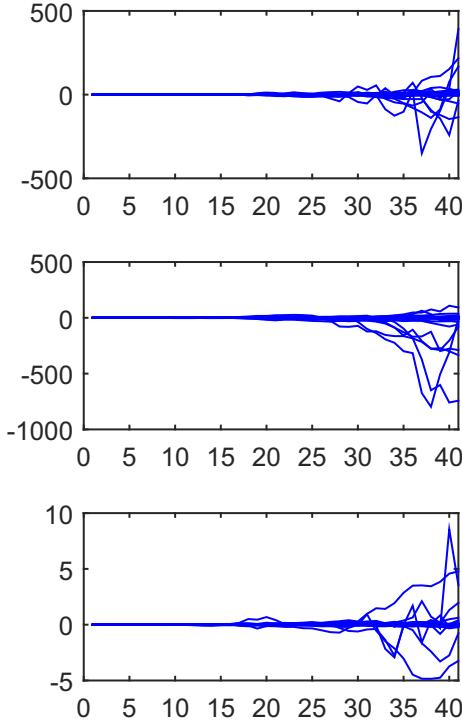


Fig. 11. The absolute value of the difference in trajectories for particles integrated using `ode113` and a factor of 10 difference in the tolerance parameters. The three rows show the differences (m) in longitude, latitude, and depth.

for the 2D wall-sliding part, and explore using different tolerances for the two branches. A sample of such stability is shown on Figure 11, where we deployed `ode113` with tolerances changed by a factor of 10. The largest differences in the trajectories were a few hundred meters. This work is still in progress, and we expect to increase our precision and stability substantially.

One idea is to use the lower accuracy but faster `ode23t` for the 3D interior particles and a higher precision one for the 2D wall-sliders, like `ode113`, with separate tolerances. This will require additional experimentation with the different tolerance settings, and will be addressed in the near future.

V. SUMMARY AND CONCLUSIONS

We are developing a new Lagrangian particle-tracking tool for application in physical oceanography. The algorithm is based on one that has already provided good results. It has been vectorized and parallelized to significantly improve its performance and enhance scalability. With the new algorithm large-scale and multi-time scale oceanic phenomena can be investigated from a Lagrangian perspective. This paper describes the current state of the code and algorithm development.

We have used several techniques to accelerate the particle-tracking simulation and make the algorithm scale better. First, stage 2 has been vectorized, so that all particle positions are interpolated simultaneously instead of sequentially. Second,

stages 2 and 3 have been parallelized so that multiple processors and threads can be used. Even before the bucket-list is fully implemented, our conservative end-to-end lapse time (including the various initialization steps) for our test case went from 631s to 156s, mostly dominated by I/O (about 130s).

Before we can fully vectorize the tracking of particles we need to interpolate many particle trajectories simultaneously. We were able to implement this through loading two full 3D velocity fields and performing 4D interpolation on the full-domain matrices. Using an improved interpolation method, this process is now 5 times faster even in the serial mode, before vectorization. The time of vectorized interpolation grows only very slowly with the number of particles: for 10^3 particles the interpolation takes 5 times longer than for a single particle, corresponding to a 200-fold increase in performance. For 10^4 particles this improvement is a factor of 675 on a CPU, and 1230 on a GPU. Finally, for 10^5 particles we get a 3300-fold speedup in the interpolation speed. We expect an excellent scaling behavior as the number of particles is increased.

We have developed a novel strategy for parallelization inspired by the natural division between interior and wall-bound particles. The simulation starts with all particles in one bucket. When a particle hits a solid boundary, it moves into another bucket, whose content is queued by the time of these events. When a particle subsequently moves back into the interior domain, it is moved to another bucket on a third thread and so on. This approach allows a maximum number of trajectories to be simulated simultaneously, because the interior and wall-sliding particles use a different part of the algorithm with a separate ODE solver and different event-detection strategy. This part of the algorithm is currently in development and it is too early to quote actual performance numbers. In any case we expect to run at least 10^5 particles in a single simulation—a huge improvement over the original serial algorithm with typically several hundred particles.

Parallelization of the stage 3 property-extraction part of the algorithm is based on time slab. Consecutive property time slabs loaded and all the particle positions are interpolated onto the 4D field simultaneously. Implementing a series of optimization steps, we have been able to improve the performance by a factor of 5 for cold cache and a factor 10 for warm cache.

Most of the algorithm is now ready to be used in production mode, but we envision a number of further improvements to make this tool widely available. We believe that the use of compressed data structures and possibly space-filling curves have great potential for further speedup of the I/O, as many of the ocean model grid cells are below the sea floor and thus never used.

This work is part of a larger effort to make ocean model output more easily accessible to both scientists and a wider audience. This Lagrangian particle-tracking tool is one tool that will be available to explore the ocean-model output. The tools will be made available through an ePortal, with different levels of complexity available for scientists and the more general public.

ACKNOWLEDGMENT

This work was supported by the Johns Hopkins University IDIES seed funding program. AS is also supported by a grant from the Gordon and Betty Moore Foundation.

REFERENCES

- [1] B. Blanke and S. Raynaud, "Kinematics of the Pacific Equatorial Undercurrent: An Eulerian and Lagrangian approach from GCM results," *Journal of Physical Oceanography*, vol. 27, pp. 1038–1053, 1997.
- [2] K. Döös, J. Kjellsson, and B. Jónsson, *Preventive methods for coastal protection: Towards the use of ocean dynamics for pollution control*. Springer, 2013, ch. TRACMASS-A Lagrangian trajectory model, pp. 225–249.
- [3] C. B. Paris, J. Helgers, E. van Sebille, and A. Srinivasan, "Connectivity Modeling System: A probabilistic modeling tool for the multi-scale tracking of biotic and abiotic variability in the ocean," *Environmental Modelling & Software*, vol. 42, pp. 47–54, 2013.
- [4] I. M. Koszalka, T. W. N. Haine, and M. G. Magaldi, "Fates and travel times of Denmark Strait Overflow Water in the Irminger Basin," *Journal of Physical Oceanography*, vol. 43, pp. 2611–2628, 2013.
- [5] W. J. von Appen, I. Koszalka, R. S. Pickart, T. W. N. Haine, D. Mastropole, M. G. Magaldi, H. Valdimarsson, J. Girton, K. Jochumsen, and G. Krahmann, "The East Greenland Spill Jet as an important component of the Atlantic Meridional Overturning Circulation," *Deep-Sea Research I*, vol. 92, pp. 75–84, 2014.
- [6] R. Gelderloos, I. M. Koszalka, T. W. N. Haine, and M. G. Magaldi, "Seasonal variability in warm water inflow towards Kangerdlugssuaq Fjord," manuscript in preparation.
- [7] MATLAB, the MathWorks, Inc., Natick, Massachusetts, United States.
- [8] P. Bogacki and L. F. Shampine, "A 3(2) pair of Runge-Kutta formulas," *Applied Mathematics Letters*, vol. 2, pp. 321–325, 1989.
- [9] A. Arakawa and V. Lamb, "Computational design of the basic dynamical processeses of the UCLA general circulation model," in *Methods in Computational Physics*. Academic Press, 1977, vol. 17, pp. 174–267.
- [10] H. Samet, "Multidimensional spatial data structures," in *Handbook of Data Structures and Applications*. Chapman and Hall/CRC, 2004.
- [11] G. Lemson, T. Budavari, and A. S. Szalay, "Implementing a general spatial indexing library for relational databases of large numerical simulations," in *Scientific and Statistical Database Management - 23rd International Conference, SSDM 2011, Portland, OR, USA, July 20-22, 2011. Proceedings*, J. B. Cushing, J. C. French, and S. Bowers, Eds. Springer, 2013, vol. 6809, pp. 225–249.

Data Management and Simulation Support Accelerating Carbon Capture through Computing

You-Wei Cheah, Joshua Boverhof,
 Abdelrahman Elbashandy,
 Deb Agarwal
 Lawrence Berkeley National Laboratory
 Berkeley, CA

Jim Leek, Thomas Epperly
 Lawrence Livermore National Laboratory
 Livermore, CA

John Eslick, David Miller
 National Energy Technology Lab
 Pittsburgh, PA

Abstract—The Carbon Capture Simulation Initiative (CCSI) project has developed and deployed scientific infrastructure called the CCSI Toolset. The CCSI Toolset provides state-of-the-art computational modeling and simulation tools to accelerate the commercialization of carbon capture technologies from discovery to development, demonstration, and ultimately the widespread deployment to hundreds of power plants. Carbon capture technologies have the potential to dramatically reduce the carbon emissions from power plants. The CCSI Toolset provides end users in industry with a comprehensive, integrated suite of leading-edge, scientifically validated models with simulation, uncertainty quantification, optimization, risk analysis and decision making support. The CCSI Toolset has at its core an integrated framework that enables execution of simulations and workflows including optimization and uncertainty parameter sweeps using a wide variety of computing platforms including desktops, clusters, Clouds, and HPC systems. The integration framework enables the running of a variety of commercial process simulation packages as well as custom simulators. Moreover, the framework enables scientists to run and manage thousands of concurrent simulations to perform optimizations and uncertainty quantification. Components of the CCSI Toolset are connected through the use of a data management system that stores data to a repository and enables the tracking of provenance for each simulation as well as its associated components. The data management system tracks all the configurations, models, simulations, and results created during the design of a carbon capture system and supports the design life-cycle as well as decision making. The primary contribution of this paper is thus the design and implementation of the integration framework within the CCSI Toolset, which provides both data management and simulation support for CCSI. This integration framework has been deployed and is in use by several groups of researchers and commercial entities.

Keywords—carbon capture, cyberinfrastructure, e-Science, data management, parallel simulation

I. INTRODUCTION

Carbon capture has the potential to significantly reduce greenhouse gas emissions from power plants. Development of cost effective carbon capture processes that can operate at the plant scale (typically around 650MW) is required for the widespread deployment of these technologies. Historically, the energy sector takes up to 15 years [1] to move from the laboratory to pre-deployment and on the order of 20 to 30 years for industrial scale deployment [2]. The U.S. Department

of Energy initiated the Carbon Capture Simulation Initiative (CCSI) in 2011 with the goal of developing a computational toolset that would enable industry to more effectively identify, design, scale up, operate, and optimize promising concepts (Miller et al., 2014). The CCSI project is a partnership among national laboratories, industry and academic institutions that is developing and deploying the computational modeling and simulation tools. The goal of the project is to deploy state-of-the art scientific computational capabilities to accelerate carbon capture technologies from discovery to development, demonstration, and ultimately the widespread deployment to hundreds of power plants.

The CCSI Toolset supports particle scale (computational fluid dynamics, CFD), device scale (process simulators), and plant scale (superstructure simulators) modeling and simulation. It enables multi-scale modeling and the features needed to evaluate, optimize and scale the models and to make decisions based on the results. By developing the CCSI Toolset, a comprehensive, integrated suite of validated science-based computational models, CCSI provides simulation tools that will increase confidence in designs, thereby reducing the risk associated with incorporating multiple innovative technologies into new carbon capture solutions. The scientific underpinnings encoded into the suite of models will also ensure that learning will be maximized from successive technology generations.

The requirements of the CCSI Toolset were developed through extensive interaction with the CCSI Industry Advisory Board (IAB) and active testing by industry partners during the development phase. User Experience design techniques were utilized during the project to solicit input on priorities and processes from the industry participants and to improve usability of CCSI Toolset modules [3], [4]. One of the results of our early interaction with the industry participants was the recognition that they already have extensive experience and libraries of models in existing process simulation packages including AspenPlus, Aspen Custom Modeller (ACM), Simulink, gProms, and custom simulation packages including some that are Excel-based. In addition, the users in industry are primarily using Windows systems for process simulations. Although users expressed a willingness to consider changing simulation environments, it was clear from the early interviews

that any change would require building up process model libraries and validation of these models.

Realizing the potential of CCSI depends on having an *integration framework* to provide the simulation and data management support for the multi-scale simulations, uncertainty quantification, decision support, and optimization of scientific models. The integration framework enables easy interoperability of the wide variety of process simulation packages in use in the industry with other components of the toolset and the ability to run 1000s of simulations and 100s of iterations. Simulations can be run on desktops, Cloud environments, clusters, or high-performance computing environments. The choice of simulation execution resources is based solely on the computational resources available to the user and the user's preferred runtime environment. Tracking all of the experiments, models, simulations, and results along with conclusions requires data management support to enable decisions. The primary contribution of this paper is the presentation of the design and implementation of the CCSI integration framework.

The remainder of the paper is organized as follows. We first discuss related work in Section II and present the CCSI Toolset architecture for computational tools in Section III. This is followed by an in-depth discussion of the implementation of these computational tools in Section IV. The implications of the tools and adoption are discussed in Section V. Lastly, we present our conclusions and future work in Section VI.

II. RELATED WORK

There are several scientific workflow frameworks with integrated data management and simulation capabilities. However, there is relatively little past work to implement either capability for process simulation environments. In this section, we will focus our discussion on related work for the CCSI Toolset's data management and parallel simulation support.

A. Data Management

Data management plays a key role in science cyberinfrastructure [5]. In the commercial industrial simulation space, ANSYS Engineering Knowledge Manager (EKM) provides an integrated solution providing simulation-based process data management capabilities based entirely on ANSYS tools [6]. It does not allow integration of simulators from other companies. The CCSI project must be able to also support AspenPlus, ACM, and SimuLink simulations as well as custom simulations. Various data management systems have been developed for scientific projects such as the ASCEM Data Management component [7] which relies on a combination of Velo [8], a package based on semantic wiki and Alfresco CMS capabilities, and traditional database technologies [9] for data management and Vistrails an open-source scientific workflow and provenance management system [10]. Earth System Grid is a peer-to-peer distributed data system supporting climate science simulations [11]. The high-energy physics experiments at CERN's Large Hadron Collider have also developed data management solutions. For example, the Atlas experiment uses Don Quixote 2 [12] and Panda [13]. Although a few of these

infrastructures have found acceptance beyond their specific domains, it is rare since each contains many customizations to make it useful to the original domain.

Prior research has investigated the properties needed in a science metadata system, eScience data management systems should possess a number of requirements. It is crucial that metadata is maintained for data objects [14] and their provenance needs to be tracked [15].

Versioning has been demonstrated to be useful in scientific workflows [10], [16]. In addition, data management systems should facilitate the sharing and collaboration amongst scientists. Hence, capabilities such as searching and version control are desirable for data management systems. A nice overview of much of the current work in data provenance and annotation can be found in [17]. In the CCSI Toolset, the Data Management Framework (DMF) fulfills the role for data management by incorporating the requirements for metadata, data provenance, versioning and also the facilitation of sharing and collaboration.

B. Parallel Simulations

Over the years, much research has been done on running simulations in parallel. With emergent trends such as the Cloud, studies have evaluated the effectiveness of deploying simulation runs on Cloud environments [18], [19]. When the project first started, the main distributed technologies involved MapReduce and Hadoop at that time. These technologies have proven to be useful for scaling up simulations [20]. Container managers such as Docker [21] and rkt [22] are relatively new to the scene but have revolutionized the development and deployment of large-scale simulations [23].

As technology has evolved, so has science. Many scientific domains are shifting towards running simulations in parallel. In material science, we have systems such as Fireworks [24], which is designed for high throughput applications. Similarly, GROMACS [25] is a system that does high-throughput molecular simulations. Qdo [26] is another system that supports batch execution of tasks. More recently, Apache Spark [27] and Amazon's AWS Flow Framework [28] provide support related to the CCSI Toolset's Turbine Science Gateway capabilities. However, they were not available when the project began. The CCSI Toolset is set to change the scientific landscape of carbon capture by allowing scientists to run simulations in parallel on distributed environments or Cloud resources.

III. CCSI TOOLSET ARCHITECTURE

The CCSI Toolset's concentration is on providing the capabilities needed to enable effective design and scale-up of carbon capture processes for installation in a power plant. A typical design process starts with bench-scale experiments in order to find the most promising capture process (chemical reaction), or in some cases CFD simulations for understanding how chemical reactions would operate in a device. This is normally followed by building a small device to test basic concepts, which is then followed by using process simulation

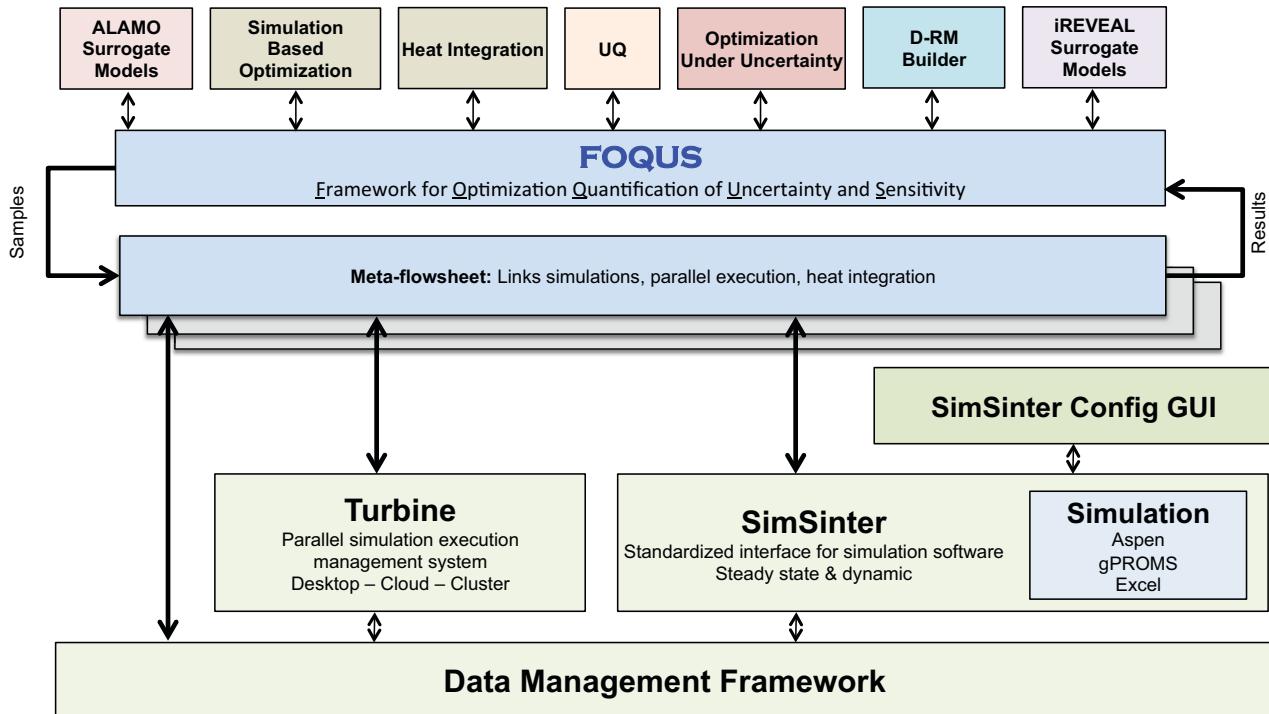


Fig. 1. Architecture of CCSI Toolset

to build successively larger devices to predict expected performance during the scale-up process. The design process like this typically involves many iterations at each scale.

The intent of the CCSI Toolset is to utilize the latest scientific computational tools and understanding of the design process to enable thorough study of the design before a physical device is built. This approach is expected to reduce the number of physical prototypes and increase the likelihood that devices built at each scale succeed the first time. As a result, this will maximize the learning at each level. In order to accomplish this goal, the existing process simulation and CFD capabilities need to be augmented with an integrated ability to perform uncertainty quantification and optimization. In addition, the process needs to be studied at all scales in the context of the design process and the plant.

To address these goals and to provide the required functionality, the CCSI Toolset is designed to be modular with a unified interface and support framework. The design leverages existing software components such as process simulators and CFD capabilities and allows the team to concentrate on building the new functionality required. Figure 1 shows a schematic of the components built for the toolset. The top layer identifies modules providing the core capabilities such as process scale modeling using surrogate models (ALAMO), optimization of device configurations, uncertainty estimation to understand sensitivities of models (using PSUADE [29]), dynamic reduced model development to enable understanding of behavior during changing conditions (D-RM Builder), and surrogate model development to enable CFD models to be

embedded in process simulations (iREVEAL). The next layer down is the Framework for Optimization Quantification of Uncertainty and Sensitivity (FOCUS), which is the brains of the Toolkit. The FOCUS layer orchestrates the execution of the core capability modules on behalf of the user. FOCUS relies on the integration framework components (the heart of the toolkit) to run simulations and store results. The process simulation framework consists of three components: 1) Turbine Science Gateway for simulation management, 2) SimSinter to connect process simulators to the framework through standard interfaces, and 3) the Data Management Framework to store and track all of the activities and data in the system.

The details of the core capability modules are beyond the scope of this paper, but an overview will be provided here to provide context and to motivate the process simulation framework capabilities. There are five major modules, they are: 1) Simulation-based optimization, 2) Uncertainty Quantification (UQ), 3) Optimization Under Uncertainty, 4) Heat Integration, and 5) D-RM Builder. Simulation-based optimization is a technique where models are evaluated by running 100's to 1000's of iterations of the process simulation software and using genetic algorithms to evaluate an objective function and also determine the optimal configuration of a device. The UQ module encompasses a rich selection of mathematical, statistical, and diagnostic tools for application users to perform UQ studies on simulation models. These studies are carried out by treating the simulation as a black box and varying the inputs while running 1000's of simulations and studying

their impact on the outputs. Optimization Under Uncertainty extends the simulation-based optimization by including studies of the contribution of model parameter uncertainties in the objective function. The Heat Integration tool maximizes heat utilization within the entire carbon capture process. And lastly, the D-RM builder enables the automatic, systematic generation of data-driven dynamic reduced models from high-fidelity dynamic models.

In addition to the five major modules, FOCUS also bundles surrogate models. Since CFD and process simulations are time consuming and may fail to converge, surrogate models are simplified (reduced) models that are meant to be executed in a much shorter time. Examples of these simplified model modules are ALAMO and iREVEAL. The ALAMO module is used to create algebraic surrogate models for supporting large-scale deterministic optimization. The iREVEAL module is an automated tool to create reduced models from CFD simulations and export them in a form that process simulators can use.

FOCUS provides a graphical user interface and uses a flowsheet as a means of describing a process. The flowsheet is a core concept of FOCUS and is akin to a workflow. Simulations are initiated through FOCUS and executed on the Turbine Science Gateway, the parallel simulation execution management system for the CCSI Toolset. SimSinter is a library that interfaces with the wide variety of process simulators to allow models to be executed using a standardized interface from Turbine Science Gateway through SimSinter. A graphical user interface is provided with SimSinter to allow configuration of simulations for use with SimSinter. The last of the integration framework tools is the Data Management Framework (DMF). It interfaces with all of the computational tools and acts as a repository for storing, organizing, and sharing of data and models in CCSI. In addition, the DMF tracks the provenance associated with data and models.

IV. IMPLEMENTATION

In this section, we discuss implementation details of the integration framework tools in the CCSI Toolset, namely, FOCUS, Turbine Science Gateway, SimSinter, and the Data Management Framework. These tools provide the data management and simulation support for accelerating carbon capture in CCSI.

A. FOCUS

FOCUS (The Framework for Optimization and Quantification of Uncertainty and Sensitivity) is an end-user application that serves as the primary computational platform in the CCSI Toolset. It provides a graphical user interface and standard platform for several CCSI tools (Figure 2).

For processes such as post-combustion carbon capture from a coal fired power plant, there are many very different subsystems involved. The boiler, steam cycle, pollution controls, carbon capture, and compression systems are all distinct parts and simulation of some of these parts may be better done separately in different software by experts in each system. A

critical feature of FOCUS is its ability to interface with and run commonly-used chemical engineering process modeling software. Through FOCUS, models constructed by many different people and using a variety of process simulation packages can be combined into a larger composite model referred to as a *flowsheet* and flowsheets can be combined into a *meta-flowsheet*. The meta-flowsheet allows connections between various flowsheets. For instance, the meta-flowsheet can be used to combine all of the carbon capture related system flowsheets into one large system, which can have recycle streams. FOCUS utilizes the tools SimSinter and the Turbine Science Gateway to run models using the external process simulation software. FOCUS provides a graphical interface to make linking simulations on Turbine into the flowsheet or meta-flowsheet relatively simple. The user can draw out their flowsheet where nodes represent simulations or calculations, and directed edges represent the flow of information between nodes. If a set of interdependent nodes is created (a flowsheet with recycle), FOCUS can automatically find an optimal set of tear edges (streams) and solve the problem iteratively.

The UQ portion of FOCUS provides an interface for the PSUADE tool, which does the underlying calculations. FOCUS first calls PSUADE to generate sets of samples to be run. This is then followed by FOCUS using Turbine and SimSinter to execute the simulations. FOCUS can take advantage of Turbine Science Gateways parallel execution capabilities to run flowsheets and meta-flowsheets in parallel; this provides a significant increase in speed when performing uncertainty quantification analysis. Once the results of the simulations are obtained, FOCUS can then use PSUADE to analyze results and generate relevant plots.

The simulation based optimization portion of FOCUS uses a plugin system that allows various derivative free optimization solvers to be added to FOCUS. The end user may write or modify derivative free optimization methods, or write FOCUS plug-in wrappers for libraries they have available. Many derivative free optimization solvers can benefit from running several simulations in parallel. For example, in evolutionary methods, an entire generation must usually be run before moving to the next iteration. Significant time savings can be achieved by running the simulations in parallel using Turbine Science Gateway.

The meta-flowsheet in FOCUS plays an important role in simulation based optimization. The optimum of a whole system may be different than independently optimizing sub systems. Since many of these models may not be compatible with any one process simulation package or may use a process simulation package that does not provide robust optimization methods, FOCUS's ability to link simulations and models built in different process simulators and run them efficiently is critical for uncertainty quantification and optimization of process-scale and plant-scale carbon capture process design.

B. Turbine Science Gateway

Turbine Science Gateway is a generic solution that can be extended to process modeling and simulation packages, and

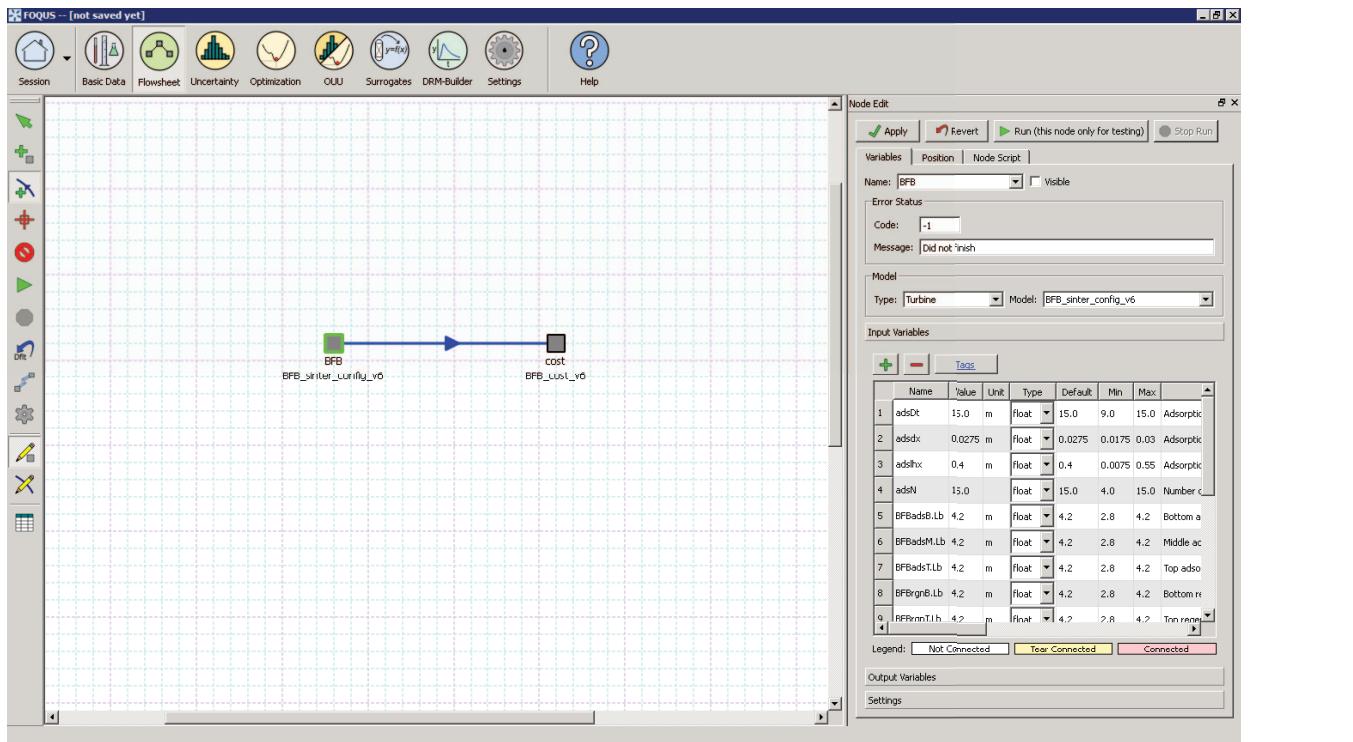


Fig. 2. Framework for Optimization and Quantification of Uncertainty and Sensitivity (FOCUS) with example Flowsheet construction

is essentially a batch system that provides staging of input and output files. The Turbine Science Gateway is composed of four components; a Turbine Web application, a Turbine Server providing an execution environment for running and managing scientific applications, a Turbine Database for storing results, and a Turbine Client interface that provides the local machine a connection to the other components. The Turbine Web allows users to retrieve, monitor, run, and manage scientific applications running on the back-end. The Turbine Server back-end execution framework can be configured for single machines, networked workstations, cluster, or Cloud computing resources such as Amazon Web Services EC2. The cluster and EC2 deployments allow for parallel application executions, where on EC2, the CCSI team has successfully executed Aspen Plus simulations using deployments of 400 concurrent instances. When deploying with EC2, the Turbine Science Gateway harnesses Amazon EC2 spot instances allowing simulation executions to happen using low costs. Through the use of spot instances, running simulations in the Cloud now become more economical than having a cluster of machines. In our experience, parallelization dramatically increases application throughput and decreases the time to solution from weeks to hours and months to days.

Turbine Science Gateway is designed to operate primarily in Windows since that is the platform of the process simulations and users. The Turbine Client, on the other hand, is a python library that is platform independent. Turbine Clients interact with the Turbine Web through a RESTful Web interface or directly through HTTTPs. The RESTful Web API defines 5

kinds of Web resources: Application, Simulation, Job, Consumer, and Session. Short descriptions of each Web resource is provided below:

1) *Application*: The Application resource is read-only, provides descriptions of the supported scientific applications, and specifies the files required for an execution.

2) *Simulation*: A Simulation resource references the target application, and contains an appropriate application model and sinter configuration file (see IV-C). These are specified as required staged-in files in the Application resource.

3) *Job*: Each simulation run is represented by a Job resource. This is essentially a single execution request. A Job must reference a Simulation resource and contains metadata concerning the progress of the job through the system (status, timestamps, etc), input data set, and output data set specified by the sinter configuration file. In addition, each Job resource contains a reference to the Consumer resource which identifies the computational resource on which the Job was executed.

4) *Consumer*: The Consumer resource represents workers that are running the scientific application. Consumers are each associated with a computational resource where each consumer process must be registered before running Jobs.

5) *Session*: Each Session resource is a set of Job resources that can be controlled as a group.

In addition to the REST API, Turbine Web also provides a Python library that is designed to be scriptable, returning structured JSON [30] output that can be easily consumed by other tools. When the Turbine Web receives a request to run a simulation or flowsheet it stores the request in the Turbine

database. It also triggers Turbine Server to launch the tasks.

The Turbine Server includes a Turbine Worker component which executes and directly manages each underlying simulation process through SimSinter. A Turbine Worker also requires a configuration string to connect to the database, which is set up during installation. Each supported process simulation package (i.e. Aspen Plus, ACM, gPROMS, Excel) has a corresponding Turbine Worker application that manages the scientific application through SimSinter.

A Turbine worker will start and connect to the database and query for a matching simulation request or job. For example, an ACM Turbine Worker will query for an ACM job. When the worker finds a job, it creates a working directory and then downloads and stores the input files there. It then utilizes SimSinter to manage the scientific application. One of the staged files placed in the working directory is the SimSinter configuration JSON formatted file. This file is required to use SimSinter, it contains information for executing the underlying scientific application, like the simulation file name (eg. ACMF). This file also specifies input parameters the user can change for a particular simulation, and output parameters which are retrieved from the underlying application after the job has completed. The worker manages most of the state changes of a job, moving it sequentially from submit, setup, running, finished or error. Additional workers can be added to form a Turbine Cluster by installing the Turbine Worker component on additional machines.

C. SimSinter

SimSinter is a standard interface library for driving single-process Windows based process simulation software. Through the use of SimSinter, the CCSI Toolset is able to provide extensible support with various simulation tools. Within the CCSI Toolset, SimSinter serves as a connector between the Turbine Science Gateway and each individual process simulation tool. SimSinter achieves this by providing a standardized execution request interface to Turbine Worker and uses a custom interface to interact with process simulation tools. It is designed to run simulations on both large-scale computers, such as on a cluster or in the Cloud, or on users local machine.

Since SimSinter supports commercial simulations, it must be executed on a machine that has the simulator and simulator licenses installed. For example, SimSinter can run Aspen Plus simulations on a desktop computer that has Aspen Plus installed and the necessary Aspen Plus licenses. In the case where users do not already have a local license copy, users can still use SimSinter remotely via the Turbine Science Gateway. However, the sinter configuration file will need to have already been created. The main purpose of a sinter configuration file is to identify simulation input and output variables that are available to users in FOCUS. Sinter configuration files are written in the JSON format and is typically created by the model creator. Outputs for SimSinter are also written using the JSON format. SimSinter can be called via the Turbine Worker, through Microsoft Excel, or standalone tools that are command-line based. As part of simplifying the configuration

process for users, a graphical user interface is bundled with SimSinter, called the SinterConfigGUI.

SimSinter leverages existing custom interfaces of process simulators and is based on .Net and COM. In current implementations, SimSinter currently supports four kinds of commercial simulators: Aspen Custom Modeler, Aspen Plus, gPROMS, and Microsoft Excel. A brief description of these simulators and how they are supported are provided below.

1) *Aspen Custom Modeler*: Aspen Custom Modeler (ACM) is an equation-oriented chemical process simulator developed by AspenTech. ACM provides a Microsoft COM interface for driving simulations. SimSinter uses the COM interface to launch ACM, inserts input variables according to the paths provided in the sinter configuration file, run the simulation, read out the output variables according to their paths, and finally closes the simulation.

2) *Aspen Plus*: Aspen Plus is a sequential-modular chemical process simulator also developed by AspenTech. Aspen Plus provides a Microsoft COM interface for driving simulations. SimSinter uses the COM interface to launch Aspen Plus, inserts the input variables according to the paths provided in the sinter configuration file, run the simulation, read out the output variables according to their paths, and closes the simulation at the end.

3) *gPROMS*: gPROMS is an equation-oriented chemical process simulator developed by Process Systems Enterprise (PSE). gPROMS provides a simulator called gORUN_XML that takes inputs and outputs from an XML file. SimSinter is able to support gPROMS by taking in an input JSON file and creating a gORUN_XML readable XML file, which is then executed using gORUN_XML. When the execution completes, SimSinter reads the results from the XML file and converts back to JSON. The simulation file provided to SimSinter must be in the .gENCRYPT format, as gORUN_XML will only run encrypted gPROMS simulations. SinterConfigGUI cannot currently be used with gPROMS, so the sinter configuration file currently must be written by hand.

4) *Microsoft Excel*: Microsoft Excel is a commonly used spreadsheet application. It is widely used by scientists for simple calculations such as costing. Excel provides a COM interface that SimSinter uses to access spreadsheet cells. Macros inside Excel are also supported by SimSinter.

D. Data Management Framework

The Data Management Framework (DMF) serves as a framework for tracking all files related to a carbon capture investigation. The framework provides support for collecting CCSI data files. Additionally, it also manages and stores metadata associated with these files. Furthermore, the Data Management Framework maintains data provenance and provides the necessary tools and interfaces to allow scientists to quickly identify outdated simulations using provenance traces. Consequently, simulations can be rerun when a model is updated and the time-consuming operations such as optimization can be performed using the same configurations as were used on the previous version. This capability enables

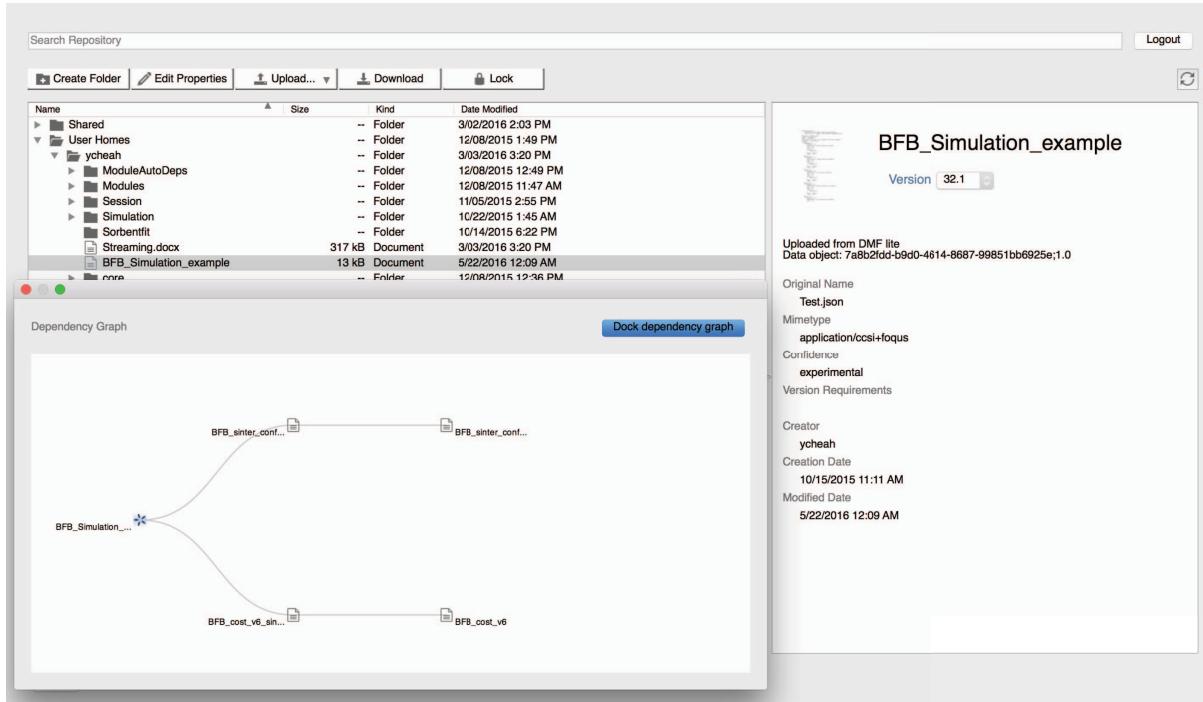


Fig. 3. CCSI Data Management Framework: DMF Browser with data provenance visualization

the design lifecycle and allows decisions to be made on information that was generated using the latest basic property and model information. The DMF backend currently comes in two flavors: *DMFServ*, a server-based feature-rich flavor and *DMF Lite*, a lightweight local machine solution. A separate client, *DMF Browser*, with a Graphical User Interface is also available and bundled with the FOQUS tool.

DMFServ is a server-based, central, network-accessible service that is suitable for deploying in an organization. This version of the Data Management Framework facilitates group interactions by providing a venue to share documents between multiple, geographically distributed individuals while tracking relationships between files. The server-based DMFServ is intended to store all the data related to a project from the gathering of experimental data, to the development of basic models, to the development of process models, and ultimately optimization and UQ studies and their results.

In addition to providing basic file sharing services, DMFServ is set up to store multiple revisions of each file in the system, allowing the DMFServ to provide a historical archive of the evolution of the project. Older versions of data and model files are stored to enable backtracking or to provide a means to compare the final product with earlier work.

Data provenance tracking is also provided with the DMFServ. As files are added to the framework, they are assigned unique IDs and versions. Users are able to specify relationships between individual files to indicate dependencies as needed. Collections of these dependency relationships help establish the provenance chain for each file in the system. Users are

thus able to see at a glance when dependencies of a simulation have been modified, indicating that the simulation may not be incorporating the best available data. These allows users to rerun existing simulations to obtain updated results.

The smaller lightweight DMF Lite provides a solution for users that tend to operate on just their local machine. Like the DMFServ, it is able to store multiple revisions of each file and provide provenance tracking. In our implementation the DMF Lite lacks the capability for sharing data directly and searching over files because it is meant to be complimentary with the DMFServ. Users of the DMF Lite are able to push their data onto the DMFServ if needed and this will open up better integrated collaborative capabilities for users.

A standalone client called the DMF Browser is also provided with the CCSI Toolset. This client can be operated both with or without the use of a graphical user interface. In the case of interfacing with other CCSI Toolset computational tools, the DMF Browser is often operated without the use of a graphical user interface. Alternatively, users can operate it in graphical user interface mode (Figure 3). The graphical user interface is in the form of a file browser and shows the metadata. These metadata includes basic file metadata like the name, author, creation time, versions of files, location of the files. In addition, users can also browse data provenance dependency graphs. The rendering of provenance graphs uses the D3 javascript library [31]. Users are able to quickly identify from the provenance graphs detailed information about data dependencies, version information, and also whether file dependencies are outdated. When the DMF Browser is used with the DMFServ

backend, additional functionality such as searching over the repository and setting basic file permissions are available. The DMF Browser can be used to browse multiple DMF backends if needed. A command-line client called the *DMF Basic Data Ingestor* was also developed to automate the process for basic data modelers to channel modeling data straight into the Data Management Framework.

The DMFServ uses the Alfresco Community Edition repository [32] as the backend repository. Alfresco was chosen as a backend repository due to its rich collaborative features, versioning and search features, and also support for a wide variety of data formats. In addition, Alfresco comes in open source and commercially supported versions allowing it to be deployed in large corporations and small contractors. DMF-Serv is implemented using a mixture of Python and Java. The DMF Lite, on the other hand, uses the Git version control system in place of Alfresco as its backend repository. The DMF Lite is written using only Python.

In the CCSI Toolset, the Data Management Framework is tightly integrated with FOCUS, Turbine Science Gateway, and SimSinter. With SimSinter, users can store their simulations right after creation or when making edits in the Data Management Framework. Users can also ingest simulations to the Data Management Framework through the FOCUS interface. When users are ready to run their experiments, the Data Management Framework allows users to choose simulations. These simulations will be synchronized with the Turbine Science Gateway before execution so that the correct versions of simulations are available before a run. When a run is completed, output files will be generated and users can choose to save the results of simulations to the Data Management Framework directly or via FOCUS after analysis.

V. DISCUSSION

In this section, we provide further information about the user experience work in the CCSI Toolset. We also present and discuss some of the real life use cases of the integration framework including ones where the Turbine Science Gateway executed thousands of simulations.

A. User Experience Design

The Industry Advisory Board (IAB) was involved in the CCSI project from the beginning and our user data collection started with informal interviews and a survey of project industry participants to better understand current simulation tools, new equipment design processes, and the industry working environment. The Industry Advisory Board's 20–30 members represent a wide variety of aspects of the power industry including power plant operators, equipment designers, and utilities. Thus their perspectives spanned the range of use cases we were targeting. One of the first things we learned was that the simulation tools in use cover a wide spectrum including many commercial, open source, and custom tools. In addition, many of the groups have developed extensive experience and confidence as well as large model libraries in these simulation tools. This survey along with the interviews

of IAB members made it clear that enabling existing process simulators was an important requirement to facilitate adoption of the CCSI Toolset by our target industry. In addition, survey results also made clear that the predominant group of users in the process simulation space were Windows users and that the CFD users were primarily using a UNIX variant. The process simulation teams also rarely had access to or experience with high performance computing platforms or even clusters. It had been our expectation before the survey that gaining access to parallel resources to execute simulations required for uncertainty and optimization studies would be easy. However, it became clear from the survey that availability of on-demand resources in the Cloud would be valuable to users with no other resources. The above examples are the key constraints learned from the users. Results from the survey and interviews also provided other more minor constraints used in designing the integration framework. In particular, the interviews provided insight into why and how CFD and process simulations are created and conducted at each industry. It also provided insight into the roles of contractors in the design process.

Throughout the CCSI project, each graphical user interface went through multiple rounds of design with experts in Human Computer Interaction. These rounds allowed us to refocus each interface from the perspective of the developer to an interface designed to aid the user of the software and fit with their design process. The reactions to the resulting graphical user interfaces have been quite positive and several of the industry members have adopted the CCSI Toolset for use in their local environment. In our follow-on project we will be supporting that use for development of multiple large-scale carbon capture demonstration projects.

B. Parallel Simulation Use Case

The CCSI Toolset integration framework has been used extensively in the last two years. In this subsection, we pick a particular four days of heavy usage to demonstrate the performance of the framework when executing thousands of simulations. We demonstrate this through use of the Turbine Science Gateway executing a ACM Hybrid Split Optimization experiment as shown in Figure 4.

The simulations were executed spanning more than four days (from Jan 21st to Jan 26th) using 50 Amazon EC2 m3-large Virtual Machines. The x-axis shows the start time, in this case the date of the day where the simulation was started. The y-axis measures the runtime in minutes for each simulation. In the figure, the blue circles refer to simulations completed successfully and the green circles indicate failed simulations. As expected, the blue circles often reflect a shorter runtime. In comparison, the simulations represented by green circles show longer runtimes and fail eventually. Of the 74131 simulations that were executed during this period of time, 62995 of the simulations completed successfully while 11135 of them failed. Turbine Science Gateway does detect failures and attempts a single restart after a failure.



Fig. 4. Plot of simulation runtime versus start time of simulation execution

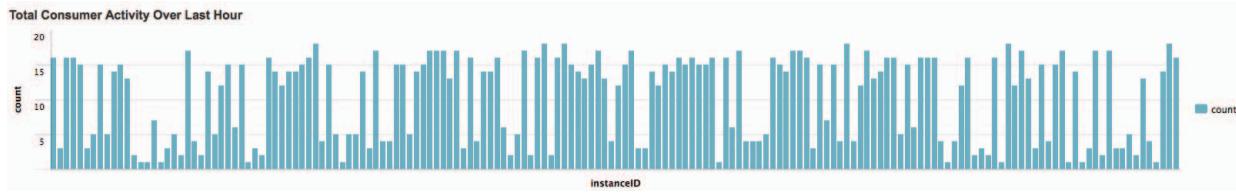


Fig. 5. Histogram of number of simulations executed by Virtual Machines

We are also able to run real-time log analysis on simulation runs. In Figure 5, we present a histogram of the number of simulations executed by Virtual Machines over an hour. Each bin in the histogram represents a single Virtual Machine (VM) and contains the number of simulations executed. This includes both simulations that succeeded and those that failed. Bins that have larger simulation counts are due to the VMs executing many simulations that complete successfully. On the other hand, some bins have lower counts due to executing non-converging simulations. Non-converging simulations have a time-out set at approximately 30 minutes.

With the deployment of the Turbine Science Gateway, CCSI has seen computational performances improved drastically. An integrated mass transfer model used to require optimization times of up to 12 hours for a single iteration on a single processor machine. The same optimization executed using the Turbine Science Gateway with 4–6 consumers now only require less than a third of the time (2 hours and 45 minutes).

VI. CONCLUSION

Traditionally, the end-to-end process for carbon capture technologies in the energy sector consumes tens of years. The Carbon Capture Simulation Initiative (CCSI) is a joint effort between national laboratories, industry and academia

with the goal of shortening the amount of time needed from discovery to deployment of carbon capture technologies. As part of CCSI, we have developed the integration framework in the CCSI Toolset consisting of computational tools with simulation models and optimization techniques. In this paper, we focus our discussion on the simulation framework and data management tools that are part of the CCSI Toolset. The simulation framework provides adapters to enable running a variety of commercial simulation packages using custom models. Moreover, the framework enables scientists to run and manage thousands of simulations to perform optimizations and uncertainty quantification. Components of the CCSI Toolset are connected through the use of a data management system that stores data to a repository and enables tracking of provenance for each simulation as well as its associated components. The CCSI Toolset has been deployed and is in use by several groups of researchers for development purposes and commercial entities for designing and deploying carbon capture equipment. The integration framework within the CCSI Toolset provides the capabilities needed to allow tasks like optimization and uncertainty quantification of carbon capture processes to be executed in hours instead of the weeks and months that were previously required. The integration framework also provides the data management tracking of

versions and results needed to support the lifecycle of a project and decision making.

Future work will concentrate on augmenting the existing CCSI Toolset with tools to help improve decision making. With the wealth of data that the CCSI Toolset collects, it is beneficial to present these data in a manner such that trial-and-error efforts are minimized and decisions can be more informed. This will likely involve the implementation of a dashboard that is able to present and integrate existing data in an effective manner.

ACKNOWLEDGMENT

The authors would like to thank Christine Moran from INRIA, Lavanya Ramakrishnan from Berkeley Labs and the broader CCSI team for providing suggestions and feedback for the writing of this paper. Special thanks to Keith Beattie and Paolo Calafiura for managing the CCSI toolset software releases. This project was funded by the US Department of Energy under the Carbon Capture Simulation Initiative through the following contracts: LBNL DE-AC02-05CH11231, LLNL FEW0180, and NETL RES-0004000.6.600.007.002.

DISCLAIMER

This paper was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

REFERENCES

- [1] J. Jenkins and S. Mansur, "Bridging the clean energy valleys of death," *Breakthrough Institute, November*, 2011.
- [2] R. S. Haszeldine, "Carbon capture and storage: how green can black be?" *Science*, vol. 325, no. 5948, pp. 1647–1652, 2009.
- [3] L. Ramakrishnan, S. Poon, V. Hendrix, D. G. ter, G. Pastorello, and D. Agarwal, "Experiences with user-centered design for the tigres workflow ap i," in *Proceedings of the 10th IEEE International Conference on e-Science*, Guaruja, Brazil, October 2014.
- [4] C. Aragon, S. Poon, G. Aldering, R. Thomas, and R. Quimby, "Using visual analytics to develop situation awareness in astrophysics," *Journal of Information Visualization*, 2009.
- [5] A. J. Hey, S. Tansley, K. M. Tolle *et al.*, *The fourth paradigm: data-intensive scientific discovery*. Microsoft research Redmond, WA, 2009, vol. 1.
- [6] ANSYS Engineering Knowledge Manager (EKM). [Online]. Available: <http://www.ansys.com/Products/Platform/ANSYS-EKM>
- [7] M. Freshley, S. Hubbard, G. Flach, V. Freedman, D. Agarwal, B. Andre, Y. Bott, X. Chen, J. Davis, B. Faybishenko *et al.*, "Advanced Simulation Capability for Environmental Management (ASCEM) Phase II Demonstration," Tech. Rep., 2012.
- [8] F. MD, G. Flach, H. Wainwright, M. Rockhold, V. Freedman, D. Moulton, P. Dixon, and J. Morse, "Applications using the advanced simulation capability for environmental management toolset," in *Waste Management Symposia*, Phoenix, Arizona, March 2016.
- [9] D. A. Agarwal, B. Faybishenko, V. L. Freedman, H. Krishnan, C. L. Gary Kushner *et al.*, "A science data gateway for environmental management," *Concurrency and Computation: Practice and Experience*, p. To appear, 2015, also appeared at GCE15: The 10th Gateway Computing Environments Workshop, 9/30 to 10/1, 2015, Boulder, CO.
- [10] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo, "Vistrails: visualization meets data management," in *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*. ACM, 2006, pp. 745–747.
- [11] H. I. and C. T. Ludwig, "Improving processes for user support in e-science," in *IEEE eScience 2014 Conference - Works in Progress session*, 2014.
- [12] V. Garonne, G. A. Stewart, M. Lassnig, A. Molfetas, M. Barisits, T. Beermann *et al.*, "The atlas distributed data management project: Past and future," *Journal of Physics: Conference Series*, vol. 396, no. 3, p. 032045, 2012. [Online]. Available: <http://stacks.iop.org/1742-6596/396/i=3/a=032045>
- [13] T. Maeno, "Panda: distributed production and distributed analysis system for atlas," *Journal of Physics: Conference Series*, vol. 119, no. 6, p. 062036, 2008. [Online]. Available: <http://stacks.iop.org/1742-6596/119/i=6/a=062036>
- [14] J. Gray, D. T. Liu, M. Nieto-Santisteban, A. Szalay, D. J. DeWitt, and G. Heber, "Scientific data management in the coming decade," *ACM SIGMOD Record*, vol. 34, no. 4, pp. 34–41, 2005.
- [15] S. Miles, P. Groth, M. Branco, and L. Moreau, "The requirements of using provenance in e-science experiments," *Journal of Grid Computing*, vol. 5, no. 1, pp. 1–25, 2007.
- [16] R. S. Barga, Y. L. Simmhan, E. Chinthaka, S. S. Sahoo, J. Jackson, and N. Araujo, "Provenance for scientific workflows towards reproducible research," *IEEE Data Eng. Bull.*, vol. 33, no. 3, pp. 50–58, 2010.
- [17] B. Ludäscher and B. Plale, *Provenance and Annotation of Data and Processes: 5th International Provenance and Annotation Workshop, IPAW 2014, Cologne, Germany, June 9–13, 2014. Revised Selected Papers*. Springer, 2015, vol. 8628.
- [18] R. M. Fujimoto, A. W. Malik, and A. Park, "Parallel and distributed simulation in the cloud," *SCS M&S Magazine*, vol. 3, pp. 1–10, 2010.
- [19] G. D'Angelo, "Parallel and distributed simulation from many cores to the public cloud," in *High Performance Computing and Simulation (HPCS), 2011 International Conference on*. IEEE, 2011, pp. 14–23.
- [20] L. Wang, J. Tao, R. Ranjan, H. Marten, A. Streit, J. Chen, and D. Chen, "G-hadoop: Mapreduce across distributed data centers for data-intensive computing," *Future Generation Computer Systems*, vol. 29, no. 3, pp. 739–750, 2013.
- [21] Docker. [Online]. Available: <https://www.docker.com/>
- [22] CoreOS rkt. [Online]. Available: <https://coreos.com/rkt/>
- [23] W. Gentsch, "Linux containers simplify engineering and scientific simulations in the cloud," in *Information and Computer Technology (GOCICT), 2014 Annual Global Online Conference on*. IEEE, 2014, pp. 22–26.
- [24] A. Jain, S. P. Ong, W. Chen, B. Medasani, X. Qu, M. Kocher, M. Brafman, G. Petretto, G.-M. Rignanese, G. Hautier *et al.*, "Fireworks: a dynamic workflow system designed for high-throughput applications," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 17, pp. 5037–5059, 2015.
- [25] S. Pronk, S. Páll, R. Schulz, P. Larsson, P. Bjelkmar, R. Apostolov, M. R. Shirts, J. C. Smith, P. M. Kasson, D. van der Spoel *et al.*, "Gromacs 4.5: a high-throughput and highly parallel open source molecular simulation toolkit," *Bioinformatics*, vol. 29, no. 7, pp. 845–854, 2013.
- [26] Qdo. [Online]. Available: <https://bitbucket.org/berkeleylab/qdo>
- [27] Apache Spark. [Online]. Available: <http://spark.apache.org>
- [28] Amazon Flow Framework. [Online]. Available: <https://aws.amazon.com/swf/details/flow/>
- [29] C. Tong, "The psuade software package version 1.0," *LLNL code release UCRLCODE-235523*, 2007.
- [30] Javascript Object Notation. [Online]. Available: <http://www.json.org/>
- [31] D3 Javascript Library. [Online]. Available: <https://d3js.org/>
- [32] Alfresco Content Management Repository. [Online]. Available: <https://www.alfresco.com>

OMUSE: Oceanographic Multipurpose Software Environment

Inti Pelupessy^{*†}, Ben van Werkhoven[†], Arjen van Elteren[†], Jan Viebahn^{*},
Adam Candy[§], Simon Portegies Zwart[†], Henk Dijkstra^{*}

^{*}Institute for Marine and Atmospheric research Utrecht, Utrecht, the Netherlands

[†]Netherlands eScience Center, Amsterdam, the Netherlands

[‡]Leiden Observatory, Leiden University, Leiden, the Netherlands

[§]Delft University of Technology, Delft, the Netherlands

Abstract—This talk will give a brief introduction to OMUSE, the Oceanographic Multipurpose Software Environment, which is currently being developed. OMUSE is a Python framework that provides high-level object-oriented interfaces to existing or newly developed numerical ocean simulation codes, simplifying their use and development. In this way, OMUSE facilitates the efficient design of numerical experiments that combine ocean models representing different physics or spanning different ranges of physical scales, for example coupling a global open ocean simulation with a regional coastal ocean model. OMUSE enables its users to write high-level Python scripts that describe simulations. The functionality provided by OMUSE takes care of the low-level integration with the code and deploying simulations on high-performance computing resources, allowing its users to focus on the physics of the simulation. We give an overview of the design of OMUSE and the modules and model components currently included. In particular, we will discuss the process of creating a new OMUSE interface to an existing code, and explain how OMUSE keeps track of the internal state of a running simulation. In addition, we will discuss the grid data types and grid remapping functionality that OMUSE provides. We also give an example of performing online data analysis on a running simulation, which is becoming increasingly important as models simulate a broader range of scales, generating large datasets that cannot be fully stored for offline analysis.

ACKNOWLEDGMENT

The ABC-MUSE project is funded by Netherlands eScience Center (file number 027.013.701, 2013-2016).

REFERENCES

- [1] CDO 2015: Climate Data Operators. Available at: <http://www.mpimet.mpg.de/cdo>, 2015.
- [2] Drost, N., Maassen, J., Van Meersbergen, M. A., Bal, H. E., Pelupessy, F., Zwart, S. P., Kliphuis, M., Dijkstra, H. A., and Seinstra, F. J.: High-performance distributed multi-model/multi-kernel simulations: A case-study in jungle computing, in: Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International, pp. 150–162, IEEE, 2012.
- [3] Gregersen, J. B., Gijsbers, P. J. A., and Westen, S. J. P.: OpenMI: Open modelling interface, Journal of Hydroinformatics, 9, 175–191, 2007.
- [4] Griffies, S. M., Winton, M., Anderson, W. G., Benson, R., Delworth, T. L., Dufour, C. O., Dunne, J. P., Goddard, P., Morrison, A. K., Rosati, A., Wittenberg, A. T., Yin, J., and Zhang, R.: Impacts on Ocean Heat from Transient Mesoscale Eddies in a Hierarchy of Climate Models, Journal of Climate, 28, 952–977, 2015.
- [5] Hill, C., DeLuca, C., Suarez, M., Da Silva, A., and others: The architecture of the earth system modeling framework, Computing in Science & Engineering, 6, 18–28, 2004.
- [6] IPCC, 2013: Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change (IPCC) [Stocker, T.F. et al. (eds)], Cambridge University Press, Cambridge, UK and New York, NY, also available from www.ipcc.ch, 2013.
- [7] Jones, P. W.: First-and second-order conservative remapping schemes for grids in spherical coordinates, Monthly Weather Review, 127, 2204–2210, 1999.
- [8] Larson, J.: The Model Coupling Toolkit: A New Fortran90 Toolkit for Building Multiphysics Parallel Coupled Models, International Journal of High Performance Computing Applications, 19, 277–292, 2005.
- [9] Le Bars, D., De Ruijter, W. P. M., and Dijkstra, H. A.: A New Regime of the Agulhas Current Retroflection: Turbulent Choking of IndianAtlantic leakage, Journal of Physical Oceanography, 42, 1158–1172, 2012.
- [10] Maltrud, M., Bryan, F., and Peacock, S.: Boundary impulse response functions in a century-long eddying global ocean simulation, Environmental Fluid Mechanics, 10, 275–295, 2010.
- [11] Mason, E., Pascual, A., and McWilliams, J. C.: A new sea surface height-based code for oceanic mesoscale eddy tracking, Journal of Atmospheric and Oceanic Technology, 31, 1181–1188, 2014.
- [12] Peckham, S. D., Hutton, E. W., and Norris, B.: A component-based approach to integrated modeling in the geosciences: The design of CSDMS, Computers & Geosciences, 53, 3–12, 2013.
- [13] Pelupessy, F. I., van Elteren, A., de Vries, N., McMillan, S. L. W., Drost, N., and Portegies Zwart, S. F.: The Astrophysical Multipurpose Software Environment, Astronomy and Astrophysics, 557, 84, 2013.
- [14] Portegies Zwart, S., McMillan, S. L. W., van Elteren, E., Pelupessy, I., and de Vries, N.: Multi-physics simulations using a hierarchical interchangeable software interface, Computer Physics Communications, 183, 456–468, 2013.
- [15] Seinstra, F. J., Maassen, J., Van Nieuwpoort, R. V., Drost, N., Van Kessel, T., Van Werkhoven, B., Urbani, J., Jacobs, C., Kielmann, T., and Bal, H. E.: Jungle computing: Distributed supercomputing beyond clusters, grids, and clouds, in: Grids, Clouds and Virtualization, pp. 167–197, Springer, 2011.
- [16] Smith, R. D., Jones, P. W., Brueggle, B., Bryan, F., Danabasoglu, G., Dennis, J., Dukowicz, J., Eden, C., Fox-Kemper, B., Gent, P., and others: The Parallel Ocean Program (POP) reference manual, Los Alamos National Laboratory, LAUR-10-01853, 2010.
- [17] Valcke, S.: The OASIS3 coupler: a European climate modelling community software, Geoscientific Model Development, 6, 373–388, 2013.
- [18] van Werkhoven, B., Maassen, J., Kliphuis, M., Dijkstra, H., Brunnabend, S., van Meersbergen, M., Seinstra, F., and Bal, H.: A distributed computing approach to improve the performance of the Parallel Ocean Program (v2. 1), Geoscientific Model Development, 7, 267–281, 2014.
- [19] Viebahn, J. and Dijkstra, H. A.: Critical Transition Analysis of the Deterministic Wind-Driven Ocean Circulation A Flux-Based Network Approach, International Journal of Bifurcation and Chaos, 24, 1430007, 2014.
- [20] Viebahn, J. and Eden, C.: Towards the impact of eddies on the response of the Southern Ocean to climate change, Ocean Modelling, 34, 150–165, 2010.
- [21] Zijlema, M.: Computation of wind-wave spectra in coastal waters with SWAN on unstructured grids, Coastal Engineering, 57, 267–277, 2010.

Automating Environmental Computing Applications with Scientific Workflows

Rafael Ferreira da Silva*, Ewa Deelman*, Rosa Filgueira[†], Karan Vahi*, Mats Rynge*
Rajiv Mayani*, Benjamin Mayer[§]

* University of Southern California, Information Sciences Institute, Marina Del Rey, CA, USA

[†] School of Informatics, University of Edinburgh, Edinburgh EH8 9LE, UK

[§] Oak Ridge National Laboratory, Oak Ridge, TN, USA

{rafsilva,deelman,vahi,rynge,mayani}@isi.edu, rosa.filgueira@ed.ac.uk, mayerbw@ornl.gov

Abstract—Computational environmental science applications have evolved and become more complex over the last decade. In order to cope with the needs of such applications, computational methods and technologies have emerged to support the execution of these applications on heterogeneous, distributed systems. Among them are workflow management systems such as Pegasus. Pegasus is being used by researchers to model seismic wave propagation, to discover new celestial objects, to study RNA critical to human brain development, and to investigate other important research questions. This paper provides an introduction to scientific workflows and describes Pegasus and its main features. The paper highlights how the environmental science community has used Pegasus to automate their scientific workflow executions on high performance and high throughput computing systems by presenting three use cases: two Earth science workflows, and a climate science workflow.

Index Terms—Environmental Computing; Scientific workflows; Pegasus Workflow Management System

I. INTRODUCTION

Scientific computing is the mainstream method for scientists that want to extract the maximum information out of their data—which are often obtained from scientific instruments such as the USArray Transportable Array [1] (seismic instruments). These computations often require the processing and analysis of vast amounts of data to understand complex system behaviors and interactions. In order to support the computational and data needs of today’s science, scientists have used distributed computing infrastructures (e.g., grids, clouds, campus clusters, and supercomputers) to efficiently compute their analyses in a reliable and scalable way. This is the case in fields such as seismology, climate and ocean modeling, hydrology, astronomy, bioinformatics, and physics.

In the meantime, scientific workflows have emerged as a flexible representation to declaratively express complex applications with data and control dependencies. As a result, many scientific workflow management systems (WMSs) have been developed [2]–[5], and they have been intensively used by various research communities [6]. Workflows allow researchers to easily express multi-step computational pipelines involving simulation, data analysis, and visualization; and to abstract the specificities of the computing environment allowing scientists to focus on the experiment definition and the analysis rather than the configuration and deployment of the

execution environment. Additional advantages of workflows include reusability (aids reproducibility), data provenance, fault-tolerance, parallel distributed computations, among others.

In this paper, we provide an overview of scientific workflows, presenting their main characteristics and applicability, as well as examples of workflow management systems (WMSs) that address different classes of problems. We describe the Pegasus WMS [2], a well-established workflow system that is widely used by the scientific community to seamlessly manage the execution of computations across distributed, heterogeneous systems. Then, we illustrate three use cases from the Earth science (CyberShake [7] and Seismic Ambient Noise Cross-Correlation [3]) and the climate science (ACME [8]) domains, highlighting how these communities have benefited from the use of scientific workflows, in particular Pegasus, for running small- and large-scale computations in the cloud and on national cyberinfrastructures.

This paper is structured as follows. Section II describes previous works, and highlights selected applications. Section III presents an overview of scientific workflows and workflow management systems. Section IV describes Pegasus, its properties, and its main capabilities. Section V describes the three environmental computing workflows for the Earth science and climate science domains, and Section VI concludes the paper.

II. RELATED WORK

Scientific workflows have been used for over a decade, and a plethora of workflow management systems have been developed to attend the different needs of computational science applications [2]–[5]. As a result, workflow management systems became the focus of many characterization studies and surveys [9], [10]. In environmental computational science, most of the applications that use workflows target large-scale analyses on distributed systems [3], [7], [8]. Workflows are also extremely useful to manage the execution of dispersed analyses on heterogeneous systems [11]. Therefore, in this paper we provide an introduction to scientific workflows highlighting their main properties and functionalities, and how they may fit the requirements of environmental computing applications.

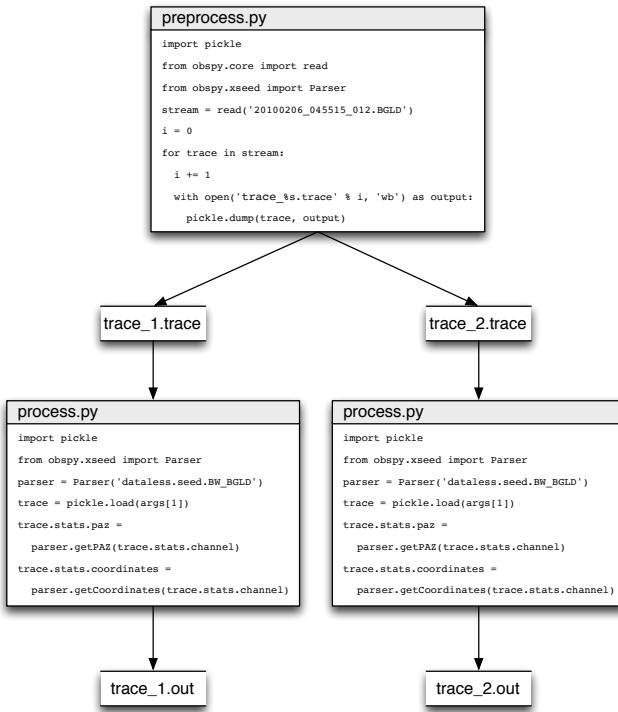


Fig. 1. An example of a workflow that processes seismology data. The workflow is composed of two task types (`preprocess.py` and `process.py`), which are Python programs.

III. SCIENTIFIC WORKFLOWS

Scientific workflows are an important abstraction for the composition and automation of complex scientific applications in a broad range of domains. They provide an abstraction above the individual application components and often abstract the details of the execution environment. Workflows allow scientists to easily express multi-step computational tasks, for example retrieve data from an instrument or a database, reformat the data, and run an analysis; while automating data movement between workflow processing stages. Typically, a workflow is described as a directed acyclic graph (DAG), where the nodes are tasks and the edges are dependencies (often data dependency). Workflows may also contain conditional statements (`if-then-else` statements, exception handling) or loops. Fig. 1 shows an example of a workflow composed of two tasks (`preprocess.py` and `process.py`), where dependencies are expressed as data files. In this example, the `process.py` tasks only start their execution, once their input data (`trace_1.trace` and `trace_2.trace`) are available, i.e. the task `preprocess.py` is successfully completed.

Task. In scientific workflows, a task often represents a program (or a script) written in any programming language, which is seen as a black box by the workflow engine—the workflow system is not aware of the intrinsics of the application code, it is limited to the invocation of the application with given parameters. Few workflow management systems require the

implementation of the workflow mechanism as part of the application code (white box application). This is typically the case of shared memory applications. A task may also represent a local or remote Web service, sub-workflows, etc. In addition, a single task execution may require a single CPU (embarrassingly parallel, no communication between processes), or multiple CPUs (e.g., MPI-like, i.e. tightly coupled processes—there is communication between processes).

Dependency. Workflow dependencies define the order that tasks should be executed in a workflow—the outcome of parent tasks are the input for child tasks. Typically, dependencies are based on flow of data between tasks (as shown in Fig. 1). However, they can also be expressed as any kind of event that could trigger the execution of the subsequent tasks or advance the workflow execution. Other types of dependencies include conditional statements, exceptions (error handling), user triggered action (the workflow waits for the user's input to proceed with the execution), among others.

A. Workflow Composition

Scientific workflows are described as high-level abstraction languages which mask the complexity of execution infrastructures to the scientist. The example shown in Fig. 1 describes the problem that the scientist is aiming to solve as a multi-step computational analysis. Note that this representation does not provide any detail about the underlying execution platform, i.e., it is an *abstract workflow*. During execution, the workflow system concretizes the abstract workflow by mapping it to the execution infrastructure, which therefore is now called the *executable workflow*. The mapping phase may add additional tasks to the workflow to enable its execution on the computing platform (e.g., stage in input data), and performs static workflow optimizations (e.g., data cleanup, task clustering, workflow reduction, etc.).

Besides the execution environment abstraction, abstract workflows also foster reproducibility through workflow reuse and data provenance. The abstract workflow can run in different execution environments with no alterations. For instance, scientists can share their workflows within the scientific community, and other scientists can reproduce an equivalent analysis following the same recipe. Data provenance (the process of tracing and recording the origins of data and its movement) allows scientists to easily trace the origin of the data produced.

Workflow Patterns and Design. The shape of a workflow may assume a variety of different forms. For instance, in neuroinformatics and bioinformatics, most of the workflows are structured as pipelines (sequential tasks, no parallelization). More complex workflows are composed of parallel splits and merges, decision points (conditions), etc. [12]. As a result, several workflow languages were developed to describe workflows. A workflow language is a formalism expressing the causal/temporal dependencies among a number of tasks to execute. Currently, there is no standard way to describe a workflow (most workflow systems defined their

own language), although there are some efforts in trying to develop a common workflow language [13]. Current workflow management systems provide either a graphical interface to build workflows (drag-and-drop) or command line interfaces (via APIs), or both. While graphical interfaces provide easy mechanisms to create workflows, they may limit the versatility of workflows to handle large-scale complex applications—it is challenging to visualize and manipulate a graph with hundreds of thousands of nodes. On the other hand, command line interfaces require programming skills. However, they easily scale to tackle large-scale complex problems.

B. Workflow Execution

Workflow interpretation and execution are handled by a workflow engine that manages the execution of the application on the computational infrastructure. As aforementioned, several workflow management systems (WMSs) have been developed to address different requirements of diverse domain applications. For example, workflow systems differ in support of task-oriented or stream-based workflows. Task-oriented WMSs typically support parallel computations, where tasks may run at the same time and there is no dependency between them (e.g., Pegasus [2], Makeflow [4], etc.). On the other hand, stream-based WMSs provide concurrent task execution, i.e. a task (with precedence constraint) can start its execution before its parent tasks have completed (e.g., dispel4py [3], Nextflow [5], etc.).

Workflows may simply run in the local user's environment (e.g., laptop or desktop machine), or in large, diverse, heterogeneous, distributed systems. The scale of the system depends mostly on the complexity of the applications, and on the capabilities of the systems. For example, parameter sweep studies are often composed of independent small tasks that can significantly benefit from grid or cloud computing environments. Parallel applications often require specialized systems such as clusters and supercomputers (for large scale applications), which can provide high connectivity with low latency (required for the high frequency of message exchanges). Note that a workflow may be composed of different application types, where some of the workflow tasks are optimized to run in high throughput systems (e.g., grids), and the remainder require high performance computing platforms (e.g., supercomputers). The data movement between the different computing environment can then automatically handled by the WMS.

Fig. 2 shows an example of a workflow that runs in different computing infrastructures. The workflow is composed of a `split.sh` job, which divides the input data into several small chunks of data that can be processed independently (e.g., data preparation, filtering, etc.). These tasks (`process.R`) can then run in high throughput systems. The following step (`merge.c`) is a parallel task (MPI application) that will merge the contents of the processed data (e.g., cross-correlation analysis), and requires high performance systems such as clusters or supercomputers. Finally, the outcome of the analysis may be visualized locally on the scientist's desktop machine.

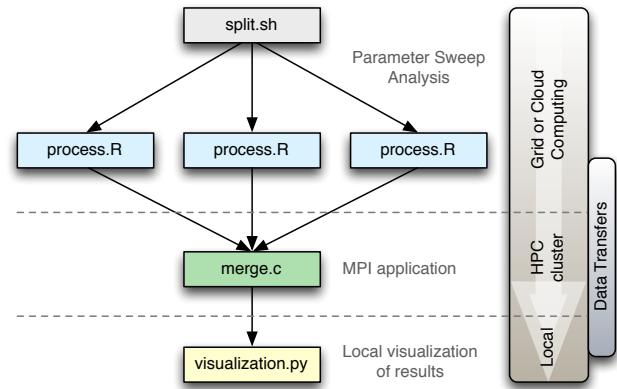


Fig. 2. An illustration of a workflow to that runs in a heterogeneous computing environment. The top part of the workflow performs a parameter sweep analysis and runs on grids or cloud environments. The middle part is an MPI-like application and requires HPC resources. The bottom part is a simple visualization script that runs locally in the scientist's computer. Data transfers between different computing environments are automatically handled by the workflow system.

IV. PEGASUS WORKFLOW MANAGEMENT SYSTEM

In this section, we present Pegasus [2], a well established scientific workflow management system for automating, recovering, and debugging scientific computations. Pegasus focuses on scalable, reliable, and efficient workflow execution on a wide range of systems, from the user's desktop to state-of-the-art high-throughput and high-performance computing systems.

Pegasus bridges the scientific domain and the execution environment by automatically mapping high-level abstract workflow descriptions onto distributed heterogeneous resources. It can manage data on behalf of the user, infer the required data transfers, register data into catalogs, and capture performance information while maintaining a common user interface for workflow submission. In Pegasus, workflows are described abstractly as directed acyclic graphs (DAGs), where nodes represent individual computational tasks and the edges represent data and control dependencies between tasks. The abstract workflows do not have any information regarding physical resources or physical locations of data and executables. The abstract workflow description is represented as a DAX (DAG in XML), describing all tasks, their dependencies, their required inputs, their expected outputs, and their invocation arguments.

A. Overview of Pegasus Functionalities

Pegasus has a number of features to facilitate and automate workflow executions, and foster collaboration and reproducible research:

- 1) *Versatility*: Pegasus defined workflows can run in different computing environments without modifications to

the abstract workflow. The same workflow can run on a single system or across a heterogeneous set of resources.

- 2) *Data Management*: Pegasus automatically handles data transfers. It autonomously determines the best available replica of the input data; performs data movement between tasks (within the same computer system, or across different computing platforms); and records output data into data catalogs.
- 3) *Fault-tolerance*: Pegasus provides several mechanisms to mitigate faults. Tasks and data transfers are automatically retried in case of failures, alternative data sources are used to stage the data, etc. Automated storage constraints mechanisms ensure that storage limits are honored by removing unnecessary data during the workflow execution. Pegasus also provides workflow recovery support through workflow-level checkpointing, and by providing a *rescue* workflow, a reduced version of the original workflow containing only the tasks that were not executed.
- 4) *Performance Optimizations*: Pegasus performs automatic workflow optimizations for improving the efficiency of the workflow execution. Tasks are reordered, grouped, and prioritized in order to increase the overall workflow performance. Pegasus determines existing workflow's output data, and prunes the workflow to prevent unnecessary re-computation.
- 5) *Scalability*: Pegasus can run workflows composed of a few tasks up to millions of tasks. It also provides scaling capabilities to dynamically adapt to the number of available resources at runtime.
- 6) *Provenance*: Pegasus enacts reproducibility through its ability to systematically capture provenance information for the derived data products. In addition to data provenance, Pegasus captures task performance data, which aids debugging.

B. Creating Pegasus Workflows

Workflow composition within Pegasus is done via APIs. Pegasus provides application programming interfaces in Python, Java, R, and Perl to generate the abstract workflow as a directed acyclic graph in XML (DAX). Fig. 3 shows the pseudocode using the Python API for generating the seismology example workflow presented in Fig. 1. The advantages of using an API include the flexibility provided by programming languages for defining large-scale workflows. For instance, parameter sweep studies can be easily defined using a simple *for* loop; a variety of workflows can be generated based on conditional statements evaluated using the workflow's input data (i.e., *if* statements may define whether a set of tasks would be part of the workflow). A complete description of the APIs can be found in the Pegasus' documentation [14]. Note that the abstract workflow does not require any information about the files locations nor details about the computing platform.

Pegasus also provides mechanisms to define dynamic workflows, i.e., workflows that are generated at runtime. Dynamic workflows can be expressed as sub-workflows—a task of the

```

#!/usr/bin/env python
from Pegasus.DAX3 import *

# Create a DAX
workflow = ADAG('seismology_workflow')

# Add a preprocess job
preprocess = Job('preprocess')
trace1 = File('trace_1.trace')
trace2 = File('trace_2.trace')
preprocess.uses(trace1, link=Link.OUTPUT)
preprocess.uses(trace2, link=Link.OUTPUT)
workflow.addJob(preprocess)

# Add left process jobs
process1 = Job('process')
out1 = File('trace_1.out')
process1.uses(trace1, link=Link.INPUT)
process1.uses(out1, link=Link.OUTPUT, transfer=True)
workflow.addJob(process1)

# Add right process jobs
process2 = Job('process')
out2 = File('trace_2.out')
process2.uses(trace2, link=Link.INPUT)
process2.uses(out2, link=Link.OUTPUT, transfer=True)
workflow.addJob(process2)

# Add dependencies
workflow.depends(parent=preprocess, child=process1)
workflow.depends(parent=preprocess, child=process2)

```

Fig. 3. Example of a DAX generator for the seismology example workflow shown in Fig. 1.

main workflow is actually an invocation to a DAX generator, that will create a sub-workflow during execution based on the outcomes of the previous tasks (parent tasks).

C. Executing Pegasus Workflows

For running workflows, Pegasus requires the specification of three different types of information catalogs: the (1) *site catalog*—describes the computing sites where the workflow tasks can be executed; the (2) *transformation catalog*—describes all of the executables (programs, also called *transformations*) used by the workflow tasks; and the (3) *replica catalog*—describes all of the input data stored on external servers. Pegasus uses the information provided in the catalogs to automatically map the abstract workflow onto an executable workflow (with all additional tasks to perform data stage in/out and optimizations).

Execution Environments. The simplest execution mode is the *local* method, where the workflow runs entirely in the scientist's machine. This method is particularly useful for development tests, or for scientists who want to automate small-scale analyses. Large-scale computations may be executed on grid computing platforms (e.g., Open Science Grid [15]), academic (e.g., NSF Chameleon [16]) and commercial (e.g., Amazon EC2 [17], Google Cloud [18], etc.) cloud computing platforms [19], campus clusters and national cyberinfrastructures (e.g., XSEDE [20]). During execution, the workflow is

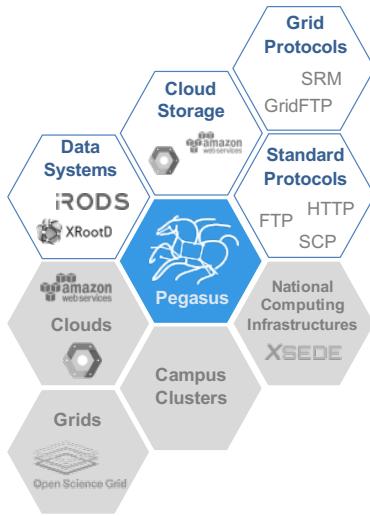


Fig. 4. Overview of execution environments (bottom grey hexagons), and data management protocols and services (top white hexagons) supported by Pegasus.

orchestrated to release tasks as they become available (parent tasks are completed), and data transfers are automatically performed within the same platform or across computing infrastructures. Note that the complexity of running workflows may increase depending on the selected infrastructure. For example, high-performance computing systems (HPC) often have stricter access, thus it may require additional steps to setup the workflow execution environment. The workflow execution complexity may also be impacted by the number of different heterogeneous computational platforms the workflow runs on. On the other hand, this may significantly improve the efficiency of the workflow execution.

Data Management. During the mapping process, Pegasus performs several optimizations as highlighted above. It provides mechanisms to automate data transfers from a number of different data protocols and services. Pegasus handles from simple files copies (using the basic linux command `cp` or `symlink`) within the same resource, up to remote data transfers using standard protocols such as HTTP, FTP, and SCP. For workflow runs conducted in grid environments, Pegasus uses SRM and GridFTP protocols, the most used transfer protocols available in these systems. For runs in cloud environments, Pegasus also interacts with cloud storages such as Amazon S3 and Google Cloud Storage using their data transfer clients. Additionally, Pegasus also provides mechanisms to interact with advanced data system such as iRODS [21] and XRootD [22].

Fig. 4 shows an overview of all execution environments and data management services and protocols supported by the Pegasus system. Note that a workflow may be composed of several different environments, and data may be staged in/out from/to different protocols or services.

Monitoring. Workflow monitoring is fundamental for debugging and following the progress of the workflow execution. Pegasus provides a suite of tools for monitoring and debugging

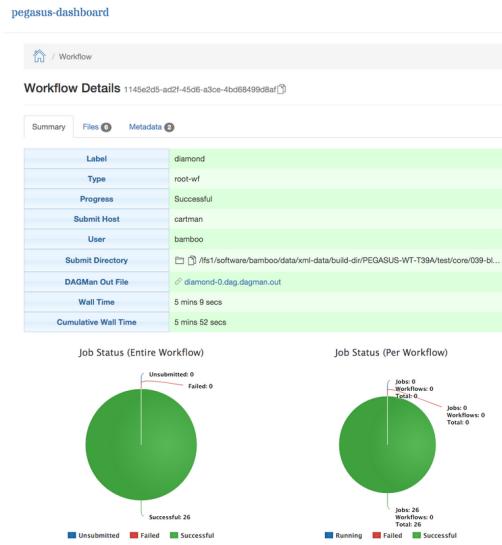


Fig. 5. Screenshot of the Pegasus monitoring dashboard.

workflows at runtime or when the workflow execution is completed. The Pegasus Dashboard is a web-based application that provides real-time monitoring of workflow executions. It shows the status of the workflow and its tasks, task characteristics, statistics, and performance metrics (Fig. 5). In addition to the dashboard, Pegasus provides command-line tools (e.g., `pegasus-statistics`, `pegasus-analyzer`) to inquire specific issues or characteristics of the workflow.

V. USE CASES: ENVIRONMENTAL COMPUTING WORKFLOW APPLICATIONS

In the past years, the environmental science community has used scientific workflows to address the simulation of complex phenomena such as climate changes and earthquake hazards. Below, we describe three workflow applications that have used Pegasus as a mean to seamlessly automate the execution of environmental computing workflows in distributed systems.

A. Cybershake

As part of its research program of earthquake system science, the Southern California Earthquake Center (SCEC) has developed CyberShake [7], a high-performance computing software platform that uses 3D waveform modeling to calculate physics-based probabilistic seismic hazard analysis (PSHA) estimates for populated areas of California. A CyberShake hazard curve computation can be divided into two phases. In the first phase, a 3D mesh of approximately 1.2 billion elements is constructed and populated with seismic velocity data. This mesh is then used in a pair of wave propagation simulations that calculates and outputs strain Green tensors (SGTs). The SGT simulations use parallel wave propagation codes and typically run on 4,000 processors. In the second phase, individual contributions from over 400,000 different earthquakes are calculated using the SGTs, then these hazard contributions are aggregated to determine the overall seismic hazard. These second phase calculations are loosely

coupled, short-running serial tasks. To produce a hazard map for Southern California, over 100 million of these tasks must be executed. The extensive heterogeneous computational requirements and large numbers of high-throughput tasks necessitate a high degree of flexibility and automation; as a result, SCEC utilizes Pegasus workflows for execution [23].

Recently, SCEC has conducted a study to run 336 Cifershake workflows over 38 days on Blue Waters (NCSA) and Titan (OLCF). Pegasus managed the execution of the workflows, which consumed about 1.4 million node-hours, and automated the management and movement of 550 TB of data within the workflows, where 197 TB were transferred from Titan to Blue Waters, and 9.8 TB staged back to USC HPC (\sim 7M files) [24]. The scalability required to achieve the science goal underscores the need for Pegasus to automate the execution and data management of SCEC's large-scale workflows.

B. Seismic Ambient Noise Cross-Correlation

Recently, the Pegasus team and the dispel4py [3] (a stream-based workflow system) team have collaborated to enable automated processing of real-time seismic interferometry and earthquake *repeater* analysis using data collected from the IRIS database [25]. Such analyses require a large number of waveform cross-correlations, which is computationally intensive. The workflow (Seismic Ambient Noise Cross-Correlation) periodically reads data from the repository (about every hour), and performs waveform cross-correlations analyses through thousands of computational tasks. The workflow consists of two main phases:

- Preprocess—each continuous time series from a given seismic station (called a trace), is subject to a series of treatments. The processing of each trace is independent from other traces, making this phase embarrassingly parallel (complexity $O(n)$, where n is the number of stations); and
- Cross-Correlation—pairs all of the stations and calculates the cross-correlation for each pair (complexity $O(n^2)$).

The workflow is implemented as a hybrid workflow approach to enable the execution of data-intensive stream-based workflow applications across different e-Infrastructures. The workflow execution is performed in heterogeneous computing platforms described as Docker containers, which can be deployed and executed in cloud computing environments. Fig. 6 shows an illustration of the *Seismic Ambient Noise Cross-Correlation* workflow implementation. A single (non-stream) run of the workflow requests an hour data from IRIS services (USArrayTA [1], data from 394 stations), and executes the Preprocess phase in about 8 minutes (using the OpenMPI cluster deployed in Container 2), while the Cross-correlation phase requires about 2 hours for processing (using the Apache Storm [26] cluster deployed in Container 3). A stream-based version of the workflow reads data from IRIS services every 2 hours.

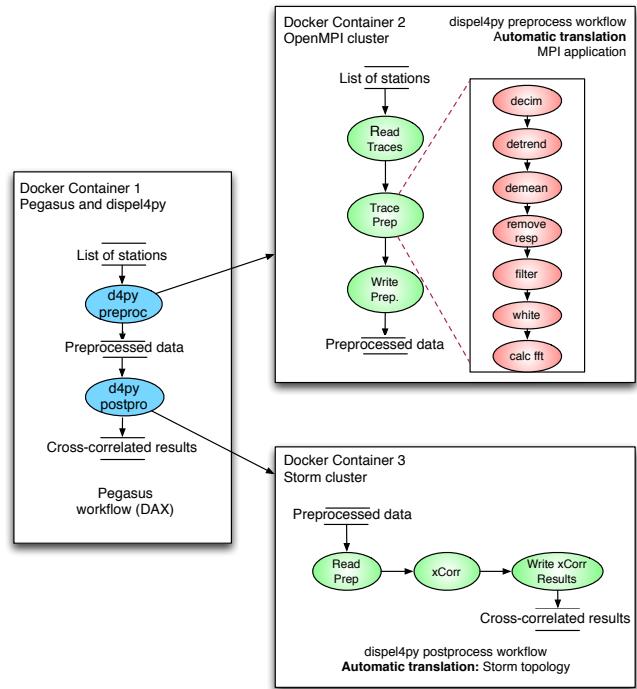


Fig. 6. Overview of the Seismic Ambient Noise Cross-Correlation workflow using Pegasus and dispel4py to automate computation on distributed resources (OpenMPI and Apache Storm clusters). Pegasus orchestrates the workflow execution, and manages data movement between different computing environments, while dispel4py executes stream-based workflows within a container.

The stream-based executions of the *Seismic Ambient Noise Cross-Correlation* workflow are managed by dispel4py, while the data movement between different execution infrastructures, and the coordination of the application execution are automatically managed by Pegasus. This approach enables the distributed allocation of stream-based workflows (represented as tasks of the Pegasus workflow) to the appropriate computing resources—allocate the workflow to the resource that could execute it in the most efficient way.

C. ACME

The Accelerated Climate Modeling for Energy (ACME) project is using coupled models of ocean, land, atmosphere and ice to study the complex interaction between climate change and societal energy requirements. The ACME workflow [8], [27] automates the manual effort involved in monitoring and resubmitting the model code in case of failures, and provides periodic reporting for validation of science outputs. The workflow divides a large climate simulation into several stages (Fig. 7). Each stage completes a portion of the total target simulation time. Each stage also produces history files, which are used by the workflow to automatically compute summary data called climatologies. This climatology data can be reviewed periodically by project scientists to ensure that the simulation is progressing as expected, so that problems can be identified, and corrections made, before computing resources are wasted. The workflow has been executed in

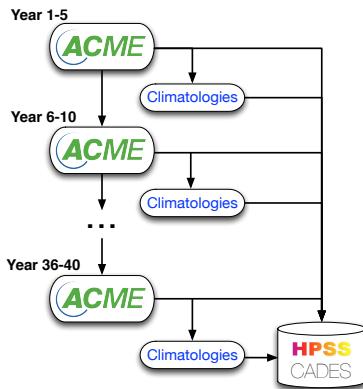


Fig. 7. Overview of the Accelerated Climate Modeling for Energy (ACME) workflow.

high performance computing infrastructures due to its complex large-scale parallel computations. The number of nodes (or cores) needed is determined from the size of the grid problem. A simple workflow execution on Titan (OLCF) with four different parameter values for a number of simulation time units (5 years per stage), consumes over 50,000 CPU hours.

VI. CONCLUSION

In the past years, workflows have been used by several science domains to efficiently manage their computations on a number of different resources: from the scientist's desktop to complex, state-of-the-art systems. For instance, the environmental science community has used workflows to create earthquakes hazard maps, or to compute seismic noise cross-correlations. In this paper, we have presented an introduction to scientific workflows, emphasizing their main properties and how scientists may model their experiments as workflows. We described the Pegasus workflow management system, highlighting three use-cases where the environmental scientists have used Pegasus to advance their research methods.

Although workflows have been used by a small group of environmental scientists, we believe that more researchers can benefit from these technologies. The main goal of this paper is to provide practical guidance on how scientists, in particular environmental scientists, may benefit from the use of workflows. On the other hand, new workflow requirements such as in-situ analysis and visualization or real-time data stream analyses, also challenge the workflow research community to provide efficient and advanced tools and resources [28].

ACKNOWLEDGEMENTS

This work was funded by DOE under the contract number #DESC0012636, “Panorama–Predictive Modeling and Diagnostic Monitoring of Extreme Science Workflows”; and by the National Science Foundation under the SI²–SSI program, award number 1148515.

REFERENCES

- [1] “USArray TA,” <http://www.usarray.org/researchers/obs/transportable>.
- [2] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, and K. Wenger, “Pegasus, a workflow management system for science automation,” *Future Generation Computer Systems*, vol. 46, pp. 17–35, 2015.
- [3] R. Filgueira, A. Krause, M. Atkinson, I. Klampinos, and A. Moreno, “dispel4py: A python framework for data-intensive scientific computing,” *International Journal of High Performance Computing Applications (IJHPCA)*, 2016.
- [4] M. Albrecht, P. Donnelly, P. Bui, and D. Thain, “Makeflow: A portable abstraction for data intensive computing on clusters, clouds, and grids,” in *Proceedings of the 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies*. ACM, 2012, p. 1.
- [5] “Nextflow,” <http://www.nextflow.io/index.html>.
- [6] I. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields, *Workflows for e-Science: scientific workflows for grids*. Springer Publishing Company, Incorporated, 2014.
- [7] R. Graves, T. H. Jordan, S. Callaghan, E. Deelman, E. Field, G. Juve, C. Kesselman, P. Maechling, G. Mehta, K. Milner *et al.*, “Cybershake: A physics-based seismic hazard model for southern California,” *Pure and Applied Geophysics*, vol. 168, no. 3-4, pp. 367–381, 2011.
- [8] E. Deelman, C. Carothers, A. Mandal, B. Tierney, J. S. Vetter, I. Baldin, C. Castillo, G. Juve, D. Krol, V. Lynch, B. Mayer, J. Meredith, T. Profen, P. Ruth, and R. Ferreira da Silva, “PANORAMA: An approach to performance modeling and diagnosis of extreme scale workflows,” *International Journal of High Performance Computing Applications*, 2015.
- [9] L. Adhianto, S. Banerjee, M. Fagan, M. Krentel, G. Marin, J. Mellor-Crummey, and N. R. Tallent, “Hpc toolkit: Tools for performance analysis of optimized parallel programs,” *Concurrency and Computation: Practice and Experience*, vol. 22, no. 6, pp. 685–701, 2010.
- [10] E. M. Bahsi, E. Ceyhan, and T. Kosar, “Conditional workflow management: A survey and analysis,” *Scientific Programming*, vol. 15, no. 4, pp. 283–297, 2007.
- [11] R. Filgueira, R. Ferreira da Silva, A. Krause, E. Deelman, and M. Atkinson, “Asterism: Pegasus and dispel4py hybrid workflows for data-intensive science,” in *The Seventh International Workshop on Data-Intensive Computing in the Clouds (DataCloud)*, submitted, 2016.
- [12] W. M. van Der Aalst, A. H. Ter Hofstede, B. Kiepuszewski, and A. P. Barros, “Workflow patterns,” *Distributed and parallel databases*, vol. 14, no. 1, pp. 5–51, 2003.
- [13] “Common workflow language,” <http://www.commonwl.org>.
- [14] “Pegasus documentation,” <https://pegasus.isi.edu/documentation>.
- [15] “Open science grid,” <https://www.opensciencegrid.org>.
- [16] “Chameleon cloud,” <https://www.chameleoncloud.org>.
- [17] “Amazon elastic compute cloud,” <http://aws.amazon.com/ec2>.
- [18] “Google cloud platform,” <https://cloud.google.com>.
- [19] E. Deelman, K. Vahi, M. Rynge, G. Juve, R. Mayani, and R. Ferreira da Silva, “Pegasus in the cloud: Science automation through workflow technologies,” *IEEE Internet Computing*, vol. 20, no. 1, pp. 70–76, 2016.
- [20] “XSEDE,” <https://www.xsede.org>.
- [21] A. Rajasekar, R. Moore, C.-Y. Hou, C. A. Lee, R. Marciano, A. de Torcy, M. Wan, W. Schroeder, S.-Y. Chen, L. Gilbert *et al.*, “iRODS primer: integrated rule-oriented data system,” *Synthesis Lectures on Information Concepts, Retrieval, and Services*, vol. 2, no. 1, pp. 1–143, 2010.
- [22] L. Bauerdtick, D. Benjamin, K. Bloom, B. Bockelman, D. Bradley, S. Dasu, M. Ernst, R. Gardner, A. Hanushevsky, H. Ito *et al.*, “Using xrootd to federate regional storage,” in *Journal of Physics: Conference Series*, vol. 396, no. 4. IOP Publishing, 2012, p. 042009.
- [23] S. Callaghan, E. Deelman, D. Gunter, G. Juve, P. Maechling, C. Brooks, K. Vahi, K. Milner, R. Graves, E. Field *et al.*, “Scaling up workflow-based applications,” *Journal of Computer and System Sciences*, vol. 76, no. 6, pp. 428–446, 2010.
- [24] “SCEC Cybershake: Using Pegasus to run across OLCF Titan, Bluewaters and HPCC,” <https://pegasus.isi.edu/2016/02/18/pegasus-scec-2015/>.
- [25] “IRIS: Incorporated research institutions for seismology,” <https://www.iris.edu>.
- [26] “Apache storm,” <http://storm.apache.org>.
- [27] B. Mayer, P. Worley, R. Ferreira da Silva, and A. Gaddis, “Climate science performance, data and productivity on Titan,” in *Cray User Group Conference*, 2015.
- [28] R. Ferreira da Silva, W. Chen, G. Juve, K. Vahi, and E. Deelman, “Community resources for enabling and evaluating research on scientific workflows,” in *10th IEEE International Conference on e-Science*, ser. eScience’14, 2014, pp. 177–184.

Utilising Amazon Web Services to provide an on demand urgent computing facility for climateprediction.net

Peter Uhe^{*†}, Friederike E. L. Otto*, Md Mamanur Rashid[†], David C.H. Wallom[†]

* Environmental Change Institute,
University of Oxford,
Oxford, United Kingdom

† Oxford e-Research Centre,
University of Oxford,
Oxford, United Kingdom

Email: Peter.Uhe@ouce.ox.ac.uk

Abstract—climateprediction.net has traditionally been an activity that requires a large amount of computing resources from its volunteer network, whilst allowing a time-frame of weeks to months for simulations to be returned for each project. However, there is an increasing trend of projects requiring results in shorter and shorter timescales. Under no project is this clearer than in the World Weather Attribution (WWA) initiative, where we are aiming to provide in near to real-time an answer to how anthropogenic climate change has altered the frequency of occurrence of a particular type of extreme weather event, either as it happens or as soon after as is practical. As such we need the ability to run simulations on alternate resources when volunteer resources will not provide results within the necessary timeframe.

This paper describes a workflow to distribute ensembles of climateprediction.net simulations in the Amazon Elastic Compute Cloud, to provide urgent compute capability for projects such as WWA. We propose a method of optimizing the use of cloud resources to minimize cost while maximizing throughput. A case study is presented to provide a proof of concept of this methodology. As such, this is a clear example of beneficial utilisation of cloud resources to supplement those available through our volunteer community.

I. INTRODUCTION

A. World Weather Attribution and the need for urgency

climateprediction.net (CPDN, [1]) is a climate modelling project using volunteer computing to run very large ensembles of climate simulations. It is run within the Berkeley Open Infrastructure for Network Computing (BOINC) framework, and tens of thousands of climate simulations are computed every year, with the results sent back to CPDN servers to be analysed by the researchers on the project. Using volunteer computing has been extremely successful since the start of CPDN in 2003. However, we are seeing increased interest in different aspects of climate change and its impacts, and new applications of CPDN are required to provide many different scenarios. This includes time critical operations that were

never part of the design of BOINC and are a challenge for systems that rely on the voluntary nature of resource donation.

The most prominent time critical activity currently undertaken in the CPDN framework is the international World Weather Attribution (WWA) initiative¹. The aim of this project is to answer questions about the role of anthropogenic climate change in the likelihood and intensity of extreme weather events, within one or two weeks after the event. This provides scientific evidence when these events are subject to international debates in the media, which may be otherwise driven by opinion.

In order to be able to provide such real-time event attribution analysis, model simulations are performed ahead of an extreme weather event occurring by utilising seasonal forecast sea surface temperatures [2]. For some events, very large ensembles of simulations are required to identify subtle but robust changes in the occurrence of, e.g., extreme rainfall events. However, we do not have the capacity to run high resolution simulations over the whole globe in advance. Depending on the nature of the event and where it has occurred, we may need to run additional simulations over a specific region to improve our statistics on the event.

In addition to the computational requirement of this activity, we need to consider the aim of addressing the questions people impacted by the event are asking. Knowing what the impacts are requires information to be provided while the event is unfolding (this can be as short as a couple of days, up to weeks). Therefore, for events where we want to look at specific impacts which are not part of our standard model output, we may want to submit new simulations with output designed to address those impacts, and hence enhance the public discussion about extreme weather events and the role

¹<http://www.climateprediction.net/weatherathome/world-weather-attribution/>

of anthropogenic climate change. These types of submissions fit into the urgent computing paradigm as we have a hard deadline of when these results are made public and the event is still fresh in the public consciousness, and after which time new results will not be used.

B. Conventional solutions to provisioning resources for urgent work

For the case where we require additional simulations and volunteer computing is unable to produce results within the required timeframe, we can look to commonly applied approaches to urgent computing.

The most common method by which any particular domain is able to provide an urgent computing capable facility is either through the provision of dedicated systems for this task or the over provisioning of a system that is in use for other non-urgent tasks [3]. An example of where this occurs is in the area of disaster response and management such as earthquake monitoring. This has a number of disadvantages around efficiency, as it is expensive to maintain a dedicated resource that is infrequently used. This may be appropriate for life or death disaster management, but is not suitable for a research project like CPDN.

Alternatively supercomputer time on clusters that are available for academic research may be purchased to run these urgent simulations. However, supercomputers generally run tasks using queues. If the cluster is utilized at or near capacity it may be difficult to submit thousands of tasks simultaneously without incurring significant queuing time, unless they are given priority over other users [3]. Additionally, supercomputers are generally optimized for high-performance computing (HPC) problems, such as large tasks using many CPU cores in a connected manner rather than many small individual tasks like CPDN.

Within the supercomputing domain there have been a number of different methods applied to attempt to bypass the problems of blocking caused by standard queueing methodologies. Examples include SPRUCE [4], where a token type of system was applied, allowing users to request elevated priority status on TeraGrid [5] connected resources. This pre-empted other tasks within the system, ensuring a high rate of throughput. This system was used a number of times for different urgent computing problems but still required a large infrastructure to manage the urgent and pre-empted tasks, to ensure that all tasks were run to completion. Alongside this are other specialised infrastructure configurations that are specifically designed to support urgent computing rather than as an addition to other types of service. An example of this is the CLAVIRE platform [6] which has constructed a specialised middleware system to support urgent computing. The biggest problem with both of these systems is though that they use large scale extremely costly computing infrastructure rather than the volunteer network with which we have had such success in CPDN.

One resource that fits the many small task paradigm is cloud computing. Commercial cloud providers provide highly scal-

able resources which are designed for use by many separate users and hence are more closely aligned with running many small jobs than large compute jobs as may be required on a supercomputer. Cloud resources are also scalable and on demand which meets our requirement for urgent computing. An example of urgent cloud computing work is in dust storm modelling where low resolution forecasts are run regularly, but then during a severe event, this is scaled up to run high resolution forecasts [7]. In addition, the scalable and on demand nature of the cloud means that there is no need to suspend other tasks to allow for our tasks to run (which may be the case on dedicated/traditional HPC machines). So we can optimise simulation run time and cost independently of others demands.

C. CPDN configuration using BOINC

The standard BOINC system relies on a set of central servers with a network of volunteer systems (hosts) operating as computational engines. Within this network, workunits (the unit of computational activity within BOINC systems) are generated within the central services, distributed and then received back in these services. There is also a degree of fault tolerance within the BOINC system where if a workunit crashes on a given computer it is resent out to another host to run (up to a maximum number of failures for a given workunit).

Any of the resources above could be used as hosts that compute simulations for CPDN. The main requirement is that the client software can be installed and the clients can communicate with the central CPDN servers. In the following sections, we introduce how the CPDN workflow can be utilised in a cloud environment and give an example of how cloud computing could supplement the existing CPDN set-up. The paper concludes with an estimation of funds required to optimally combine the advantages of a volunteer user base and cloud resources, to meet the computing challenges of a time critical research project in the focus of international media.

II. WORKFLOW FOR USING CLOUD RESOURCES FOR CPDN

For ease of deployment, avoiding costly development of new systems and integration with the current workflow, we will only consider the deployment of cloud resources in conjunction with the existing BOINC design. The cloud resource considered here is the Amazon Web Services (AWS), Elastic Compute Cloud (EC2) service². EC2 has previously be tested running CPDN simulations in [8].

As the BOINC infrastructure is already in place, the simplest mechanism for running CPDN in a cloud system is to configure cloud virtual machine instances as BOINC clients. These run as per any other volunteer and upload results back to CPDN servers. This avoids setting up an alternate control system for the CPDN simulations, but requires management of cloud resources to spin up the required computing power to run a certain number of workunits, then shut them down once they are no longer required.

²<https://aws.amazon.com/ec2/>

A. Deploying a mixed cloud and volunteer solution or stand alone system

We have a choice of whether to use cloud resources attached to the existing project or to set up a dedicated BOINC project for simulations sent to the cloud. Either of these options may be preferable depending on the particular use case.

A shared project is most appropriate in the case where we are supplementing free volunteer resources with some additional cloud resources to provide a guaranteed minimum return of simulations in a tight timeframe. However, with a shared system, we may not be able to make optimum use of Amazon EC2 resources due to BOINC mechanisms preventing volunteers from requesting work too frequently resulting in time when EC2 instances are idle. There are also costs to upload data out of Amazon (currently around \$90 per TB). However, this should be able to be waived if sending results back to a university server via a National Research and Education network³.

If we were to run a stand-alone system within AWS instead, we could also benefit by running the BOINC infrastructure within the cloud. The download and upload servers should be straightforward to set up in the cloud [8]. This would speed up data transfers and minimise transfer costs out of AWS. With data uploaded to AWS, we also have the potential for implementing instances within Amazon EC2 to analyse the results of the simulations, with direct access to the data. It would also be possible to make simulation results publically available directly from the cloud and thus simplify data access requirements enjoined by publicly funded research projects and many scientific publishers.

B. Design for controlling deployment of supplementary workunits using existing boinc infrastructure

Within Amazon EC2, spot fleets are the most cost-effective tool to manage resources⁴. These use EC2 spot instances which are significantly cheaper than on demand instances. Spot instances operate on a market price which is based on demand and differs across AWS regions and by availability zones (effectively which data centre the instances are housed in).

The main downside of spot instances is that if there is high demand and the price rises higher than the amount you bid, your instances can be terminated. However, the design of CPDN doesn't require specific simulations to be returned so can cope with a small percentage of instances terminating. When submitting simulations, a greater number are sent than required, to account for this loss. With this in mind, the chance of a simulation being interrupted is greater the longer it runs, so shorter simulations are better suited to use with spot instances. An alternate option to account for spot instance terminations is to save the state of terminating instances and restore the state into a new VM to continue the simulation.

³<https://aws.amazon.com/blogs/publicsector/aws-offers-data-egress-discount-to-researchers>

⁴<https://aws.amazon.com/blogs/aws/amazon-ec2-spot-fleet-api-manage-thousands-of-instances-with-one-request/>

However, this option introduces a lot of additional complexity and is not particularly suited for our use case.

Spot fleets maintain a set capacity for a certain amount of time, and if instances are terminated, the spot fleet will automatically spin up new instances of whichever type and availability zone is cheapest at the time (spot fleets are confined to a single AWS region though).

It is important to note though that because of the massively scalable nature of cloud resources, if we want to run a large number of simulations as quickly as possible, it is more advantageous to run many instances simultaneously than running a smaller number of instances and have each instance completing simulations sequentially. Our chosen methodology for urgent tasks is therefore to spin up a spot fleet with the capacity to run a whole ensemble in one go. On each instance we start a set of workunits to fully utilize the instance, but prevent them from requesting additional workunits. The instances are terminated once all of its simulations are completed.

If additional simulations are required, new instances will be spun up (possibly a different instance type if the spot price has changed since the initial submission). The spot fleets capacity can be dynamically modified at any time to request more or less instances. The request can also be cancelled with or without terminating running instances (depending on whether we have reached a deadline after which new simulations will not be analyzed).

III. BENCHMARKING AWS INSTANCE TYPES FOR THE CPDN WORKLOAD

Before deploying CPDN workunits to AWS through BOINC, we first want to optimize the use of resources. To do that we benchmark different Amazon EC2 instance types⁵. Amazon EC2 instance types are specified with a name such as 'c4.large' or 'm3.xlarge'. The first part of the name indicates the hardware configuration of the CPU e.g. c4 indicates a 4th generation compute optimised instance, m3 indicates a 3rd generation general purpose instance. The second part of the name indicates the number of vCPUs (hyper-threads) and amount of memory assigned to the instance e.g. 'c4.large' has 2vCPUs and 3.75GB memory, 'm3.xlarge' has 4 vCPUs and 15GB memory. We expect that all instances of the same type will have the same hardware configuration and hence similar performance.

The benchmarks were carried out running multiple copies of a stand-alone CPDN test workunit in parallel, with the number of simulations matching the number of vCPUs available on each instance type. The test workunit uses the weather@home model e.g. [9] with a 25km resolution regional domain over Europe, running a single day of simulation. This setup is consistent with a normal CPDN simulation except the test simulation does not connect to the CPDN servers and is a shorter length. For each instance types at least 4 benchmarks were run.

⁵<https://aws.amazon.com/ec2/instance-types/>

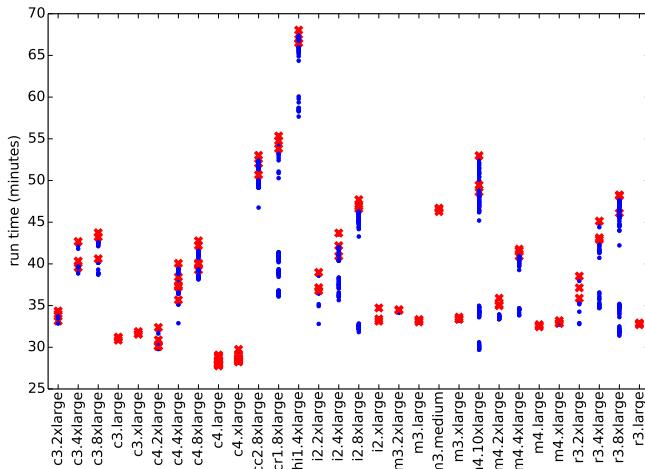


Fig. 1. Time taken to complete 1 day benchmark climate simulation using different EC2 instance types. Blue dots are completion times of each individual simulation and red crosses are the longest run time on each instance (e.g. how long that instance ran for).

On each instance, the individual simulations take different times to complete (depending on how much CPU time each process gets on the underlying physical machines). Fig. 1 shows the spread of run times of simulations (blue dots) with the run time of the instances (e.g. slowest running simulation on that instance) shown as a red cross. Optimising the balancing of load across the simulations may improve performance but is beyond the scope of this exercise.

Table I shows a list of instance types, the number of vCPUs and average run time required to complete all simulations on the instance (average time of crosses in Fig. 1). These combine to give the number of days of model simulation we can expect to compute in one instance hour. This shows a general trend of larger instances completing simulations more slowly. We assume this is to do with the way instances share the underlying CPU with other EC2 users. With smaller instances there is a greater chance the CPU is running at a lower utilization and our instances can scavenge extra CPU cycles.

Using the model days simulated per instance hour and the spot price of each instance, we can estimate the cost of running a year of simulation. Shown in Fig. 2 is this cost for a subset of instance types in each US region (where there are multiple Availability Zones in a region, the cheapest for each instance type is chosen).

When we submit simulations with a hard deadline, we also need to take into account the run length and ensure that on all instance types requested in our spot fleet the simulations will complete within the required time. In the analysis of Fig. 2, we discarded instance types which had a run time greater than 45 minutes in Table I (this will differ in real use cases). We also discarded the i2 instances which were significantly more expensive than other instance types.

TABLE I
LIST OF INSTANCE TYPES: NUMBER OF vCPUS, RUN TIMES (AVERAGE OF INSTANCE RUN TIME) AND MODEL DAYS COMPLETED PER INSTANCE HOUR BASED ON THIS RUN TIME AND NUMBER OF vCPUS.

Instance type	vCPUs	run time (min)	model days per instance hour
m3.medium	1	46.51	1.29
m3.large	2	33.20	3.61
r3.large	2	32.81	3.66
m4.large	2	32.59	3.68
c3.large	2	31.07	3.86
c4.large	2	28.19	4.26
i2.xlarge	4	33.75	7.11
m3.xlarge	4	33.45	7.17
r3.xlarge	4	33.39	7.19
m4.xlarge	4	32.99	7.28
i2.2xlarge	4	31.68	7.57
c3.xlarge	4	31.68	7.57
c4.xlarge	4	28.64	8.38
i2.2xlarge	8	37.64	12.75
r3.2xlarge	8	37.18	12.91
m4.2xlarge	8	35.33	13.58
m3.2xlarge	8	34.48	13.92
c3.2xlarge	8	33.83	14.19
c4.2xlarge	8	30.87	15.55
hi1.4xlarge	16	67.16	14.29
r3.4xlarge	16	43.73	21.95
i2.4xlarge	16	42.27	22.71
m4.4xlarge	16	41.42	23.18
c3.4xlarge	16	40.86	23.49
c4.4xlarge	16	37.69	25.47
cr1.8xlarge	32	54.64	35.14
cc2.8xlarge	32	51.92	36.98
r3.8xlarge	32	47.51	40.41
i2.8xlarge	32	47.08	40.78
c3.8xlarge	32	42.53	45.15
c4.8xlarge	36	40.67	53.11
m4.10xlarge	40	50.35	47.67

A. Optimizing cost of simulations

The model days per instance hour in Table I is used as a weighting factor by the AWS spot fleet to determine which instance type is the most cost effective. The spot fleet is configured with the allocation strategy: ‘lowestPrice’, so when starting new instances, the spot fleet will dynamically determine which instance type is the cheapest (as in Fig. 2) and spin up instances of that type.

The method above takes a simple view of optimization and chooses the cheapest instance type at the time of submission. This is most appropriate when the run times of simulations are short relative to the rate of change in the spot market. For longer simulations or where the volatility in the spot market is high, we may encounter issues where instances are terminated or result in a significantly higher cost than expected.

In these cases, the selection criteria would need to be modified to take the volatility into account. Possible alternate measures for instance type selection are lowest average price over the previous day (or appropriate time period). If terminations are more of a problem then choosing instance types with the lowest maximum spot price over a recent time period may be more appropriate. However, as spot-fleets only offer an allocation strategy which uses the instantaneous lowest price, these measures would also require developing more sophisticated control software to spin up and manage the instances.

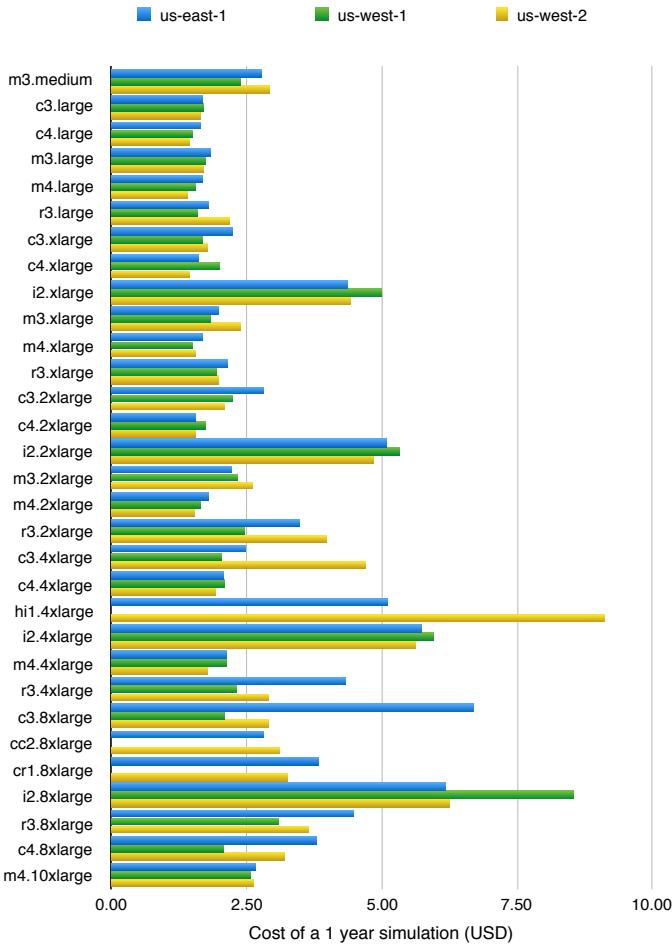


Fig. 2. Cost of 1 year of simulation in USD, by instance type and region (assuming using the cheapest availability zone in each region). Three AWS regions are shown: us-east-1 (blue), us-west-1 (green) and us-west-2 (yellow). Prices used are from the spot market as on 13/06/2016 13:45

All calculations above are made under the assumption that we are not a large enough user to use a significant proportion of the overall cloud and hence are not impacting the spot market price or instance availability (this is helped by considering as many as possible different instance types and using multiple regions).

B. Optimizing instance load

Another aspect to consider when running these instances is that each vCPU on an Amazon EC2 instance is actually a hyper-thread rather than a CPU core. So running an instance utilizing only half the hyper-threads may allow each simulation to have access to the full CPU cycles of one core and speed up the model run. The run times of simulations using c4 instance types under different loads are shown in Fig. 3. Utilizing an instance using 50% of the vCPUs or less results in run time of roughly 20 minutes which is significantly faster than using 100% of the vCPUs. However, as discussed in section III, Amazon EC2 instances are on hardware shared with other users which may be underutilizing the CPU. Because of this, the smaller instance types see a significant benefit of running

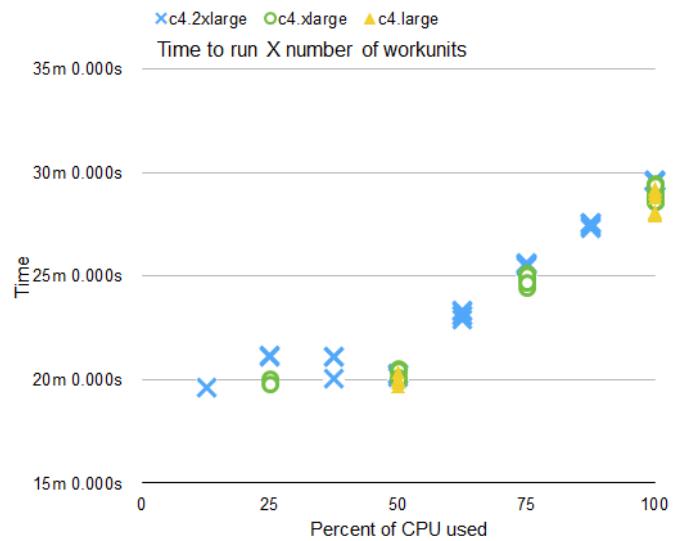


Fig. 3. Dependence of model run time on CPU utilization. This shows the change in run time as the percent of vCPUs used is increased. Three instance types are shown: c4.large, c4.xlarge and c4.2xlarge

on 100% of the vCPUs e.g. looking at Fig. 3 we see that a c4.2xlarge instance may run 4 simulations in 20 minutes using 50% of the vCPUs (and hence 8 simulations in 40 minutes), compared to 8 simulations in 30 minutes using 100% of the vCPUs. Here we have the notable exception of the c4.8xlarge which takes up a whole CPU and running on 100% of the vCPUs computes a simulation in roughly 40 minutes which is twice the time compared to the simulations using 50% of the vCPUs in Fig. 3. In that case it should be just as cost effective to run 18 simulations using half of the vCPUs as 36 simulations running at half the speed on all of the vCPUs.

However, if urgent computing takes a higher priority than cost (e.g. in the case where the deadline is too short for simulations to be completed when utilizing all the vCPUs), it would be necessary to run less workunits on each instance. Running on 50% of the vCPUs available, so each simulation has a dedicated CPU core, should result in the best performance in terms of simulation speed. If submitting simulations in this configuration, different benchmarks would be required to reflect the instance performance under this load.

IV. CASE STUDY OF A CPDN ENSEMBLE SENT TO AMAZON EC2

As a proof of concept, we have sent an ensemble of 750, 13 month simulations to Amazon EC2 instances. These simulations cover a different region to the benchmarks (South America at 50km resolution compared to Europe at 25km resolution) so the run times of these longer simulations are not comparable to the benchmarks above. However we assume that the relative performance of the different instance types will be the same. We also note that the simulation results were uploaded to a climatepredicton.net server rather than remaining in AWS.

These simulations were distributed across two AWS regions: North Virginia and Oregon. The configuration of the spot fleets sent were as follows. The spot fleet instance weightings used the ‘model days per instance hour normalised with the ‘c4.large (fastest instance with 2 vCPUs) having a value of 2. This allowed the requested capacity of the spot fleet to be roughly equivalent to the number of vCPUs e.g. a requested capacity of 20 will result in at least 20 vCPUs being available. The maximum bid price was set for each instance type as 5 times the spot price at the time of submission to reduce the likelihood of terminations.

The instances were configured using a startup script which installed the relevant software and libraries and connected to the climatprediction.net BOINC project. This script then continued to monitor the state of the BOINC project and cleaned up and terminated the instances shortly after the simulations were complete.

In total 318 instances were spun up. The capacity of the spot fleets in each region were gradually increased over periods of roughly half an hour. This was aimed to reduce the likelihood of all instances being spun up with the same instance type in the same availability zone. There were two simulations which crashed about a tenth of the way through due to model instabilities. There were also 12 c3.large instances which were terminated within a few hours of the submission (this appears due to spikes in the spot market where the price reached 10 times the on demand price), but the remainder of instances continued to completion. An outcome of this case study is that for future ensembles, we will need to filter instance types to remove those which have recently had spikes in price. A simple solution to this would be to remove those instance types from the spot-fleet, however it may also be beneficial to investigate other strategies such as discussed in section III-A. Three instance types had simulations that ran to completion: c4.large, c4.xlarge and m3.large. The run times for these instances were: c4.large 101.6 hours (range: 100.3 to 104.75), c4.xlarge 102.493902439 hours (range: 91.1 to 106.9), m3.large 119.6 hours (range: 118.4 to 120.6). We note that the outlier of 91.1 hours for the c4.xlarge instance type occurred where one of the four simulations had crashed, leading to the remaining three simulations to complete more quickly. This highlights the consequence of underfilling instances on performance.

Ratios: c4.xlarge 1.008 times c4.large, m3.large 1.18 times c4.xlarge. This compares to from the benchmarks: c4.xlarge 1.016 times c4.large, m3.large 1.18 time c4.large. So for this example, the average run lengths of our longer tests give good agreement of the relative performance of instance types. In addition, from monitoring the usage of these instances, they appear to consistently use 100% of the CPU. As the CPU is the bottleneck for the model computation, we expect that the timings of our simulations to scale linearly with run length, further justifying the use of one day simulations as benchmarks.

V. USE CASE FOR URGENT CPDN ENSEMBLE IN AMAZON EC2

The methodology for the attribution of extreme weather and climate-related events is based on the simulation of large ensembles of possible weather under present day and counterfactual climate conditions using a general circulation climate model [10]. Within the WWA project we typically run three month long simulations (at least one month before the event and one month after). These simulations use GloSEA5 seasonal forecast [11] sea surface temperatures as boundary conditions so it is possible to run them ahead of time before an event occurs (see [2]). However, as introduced above, we may not have simulations that describe the event in the most impact relevant way, i.e. an additional set of climate variables is required, or the ensemble is too small to obtain robust results. Hence cloud resources can be utilised to supplement the simulations with additional ensembles.

We can estimate the cost of running an ensemble such as this in Amazon EC2, using the benchmarks above. We take the cheapest instance type from Fig. 2 (m4.large in us-west-2). This takes 33 minutes per day of simulation (Table I), which is 49.5 hours for a three month simulation. Using the spot price as per Fig. 2, the compute cost for 10,000 workunits will be roughly \$4700. We also have the additional cost of storage used by the instances while they are computing the simulations. Given an estimated disk usage of 4GB per workunits for 49.5 hours, we estimate this to be \$275. This results in a minimum cost of \$5000 to run such an ensemble. This costing will depend on the resolution and size of the region simulated (the example here is a 25km resolution regional model over Europe), a 50km resolution model for the same region would take less than half the time to compute and cost less than half the price to run.

Data storage of the simulation results is an additional cost which will depend on how much model output is requested for analysis and how long the output needs to be stored (for example data used in publications commonly need to be kept for at least 5 years). As of June 2016, data storage in AWS Simple Storage Service costs \$30 per month per TB. If data is uploaded to other servers, this cost will vary.

As a comparison, a similar ensemble would cost over twice the amount run on a dedicated resource such as the University of Oxford Advanced Research Computing (ARC) high performance computer arcus-b⁶. The cost to run the above ensemble would be around \$12,800, based on an average benchmark run time of 32:23 and cost per CPU core of 0.02 per hour. This was based on 7 benchmarks on arcus-b where 16 simulations were run in parallel on a 16 CPU core node (comparable to the AWS set up). The storage on ARC however is slightly cheaper than AWS at \$27.40 per month per TB. Prices quoted here assume exchange rates between GBP() and USD(\$) as on 28/07/2016. Lastly we note that running on EC2 on-demand instances rather than spot instances would be

⁶<http://www.arc.ox.ac.uk/content/services>

much more expensive again (e.g. \$54,000 given the m4.large instance as above).

VI. CONCLUSION

We have presented a workflow for including AWS cloud resources within the BOINC distributed computing framework to supplement existing volunteer computing resources for CPDN. This includes optimizing the resources by benchmarking a range of AWS EC2 instance types and using spot fleets to choose the cheapest instances at the time of submission. We assume that for a given instance type, the hardware configuration used by AWS will be the same. We do note that as our benchmarks seem to depend on instance load, and usage patterns of AWS may change over time, recalculating benchmarks at regular intervals is a good idea. In addition to this, the usage patterns will differ between region (and possibly availability zones) so calculating benchmarks for different regions and availability zones may improve our optimization.

When results are needed within a very tight timeframe, and it is not expected that volunteer resources (which are only typically available when participants computers are on and idle) can produce them within that timeframe, dedicated cloud resources may be able to meet this requirement. Cloud resources also have potential to add additional compute capability where the volunteer computing base has decreased, or the project computing requirement has increased. However the cost of cloud resources are significant, so at the present time we consider their use most suited to the urgent computing case that by design cannot be met by volunteer computing. This makes them a very useful resource, but we stress that they cannot in the foreseeable future replace the unique computing framework provided by our generous and dedicated volunteer base.

ACKNOWLEDGMENT

We would like to thank our colleagues at the Oxford eResearch Centre: A. Bowery and S. Sparrow for their technical expertise. We would like to thank the Met Office Hadley Centre PRECIS team for their technical and scientific support for the development and application of weather@home. This work was funded by Climate Central through the World Weather Attribution project. The compute resources were provided under the AWS Cloud Credits for Research Program.

REFERENCES

- [1] M. Allen, "Liability for climate change," *Nature*, vol. 421, pp. 891–892, 2003.
- [2] K. Haustein, F. E. L. Otto, P. Uhe, N. Schaller, M. R. Allen, L. Hermanson, N. Christidis, P. McLean, and H. Cullen, "Real-time extreme weather event attribution with forecast seasonal SSTs," *Environmental Research Letters*, vol. 11, no. 6, p. 064006, 2016. [Online]. Available: <http://stacks.iop.org/1748-9326/11/i=6/a=064006>
- [3] S. H. Leong, A. Frank, and D. Kranzmüller, "2013 International Conference on Computational Science: Leveraging e-Infrastructures for Urgent Computing," *Procedia Computer Science*, vol. 18, pp. 2177 – 2186, 2013. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1877050913005310>
- [4] P. "Beckman, S. Nadella, N. Trebon, e. P. W. Beschastnikh, Ivan", and J. C. T. Pool, "SPRUCE: A System for Supporting Urgent High-Performance Computing". Boston, MA: Springer US, "2007", pp. 295–311. [Online]. Available: "http://dx.doi.org/10.1007/978-0-387-73659-4_16"
- [5] C. Catlett, "The philosophy of teragrid: Building an open, extensible, distributed terascale facility," in *Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on*, May 2002, pp. 8–8.
- [6] V. Alexandrov, M. Lees, V. Krzhizhanovskaya, J. Dongarra, P. M. Sloot, S. V. Kovalchuk, P. A. Smirnov, S. V. Maryin, T. N. Tchurov, and V. A. Karbovskiy, "2013 international conference on computational science deadline-driven resource management within urgent computing cyberinfrastructure," *Procedia Computer Science*, vol. 18, pp. 2203 – 2212, 2013. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1877050913005346>
- [7] Q. Huang, C. Yang, K. Benedict, S. Chen, A. Rezgui, and J. Xie, "Utilize cloud computing to support dust storm forecasting," *International Journal of Digital Earth*, vol. 6, no. 4, pp. 338–355, 2013. [Online]. Available: <http://dx.doi.org/10.1080/17538947.2012.749949>
- [8] D. P. Montes, "climateprediction.net: A Cloudy Approach," Master's thesis, Universidade de Santiago de Compostela, June 2014.
- [9] N. Massey, R. Jones, F. E. L. Otto, T. Aina, S. Wilson, J. M. Murphy, D. Hassell, Y. H. Yamazaki, and M. R. Allen, "weather@home—development and validation of a very large ensemble modelling system for probabilistic event attribution," *Q. J. Roy. Meteor. Soc.*, pp. n/a–n/a, 2015. [Online]. Available: <http://dx.doi.org/10.1002/qj.2455>
- [10] N. Schaller, F. E. L. Otto, G. J. van Oldenborgh, N. R. Massey, S. Sparrow, and M. R. Allen, "The heavy precipitation event of May–June 2013 in the upper Danube and Elbe basins [in "Explaining Extremes of 2013 from a Climate Perspective"]," *Bull. Amer. Meteor. Soc.*, vol. 95, no. 9, pp. S69–S72, 2014.
- [11] C. MacLachlan, A. Arribas, K. A. Peterson, A. Maidens, D. Fereday, A. A. Scaife, M. Gordon, M. Vellinga, A. Williams, R. E. Comer, J. Camp, P. Xavier, and G. Madec, "Global seasonal forecast system version 5 (glosea5): a high-resolution seasonal forecast system," *Quarterly Journal of the Royal Meteorological Society*, vol. 141, no. 689, pp. 1072–1084, 2015. [Online]. Available: <http://dx.doi.org/10.1002/qj.2396>

Automatic Fire Perimeter Determination Using MODIS Hotspots Information

N. Chiaraviglio*, T. Artés†, R. Bocca†, J. López†, A. Gentile†, J. San Miguel Ayanz†, A. Cortés*, T. Margalef*

* Departamento de Arquitectura de Computadores y Sistemas Operativos, Universitat Autònoma de Barcelona, Spain

Email: {nicolas.chiaraviglio, tomas.margalef, ana.cortes}@ub.cat

† Institute for the Environment and Sustainability, EU Joint Research Center, Ispra, Italy

Email: jorge.lopez-perez@ext.jrc.ec.europa.eu, jesus.san-miguel@jrc.ec.europa.eu

Abstract—Every year wildfires are responsible of the loss of thousands of hectares of European forest, millions of dollars in damage and, in the worst case, for loss of human lives. Accurate observation of wildland fire evolution is a crucial issue to estimate the burned area, to use the observed data in propagation prediction and to calibrate input parameters of propagation models. In this direction the European Forest Fire Information System (EFFIS) supports the services in charge of the protection of European forests. Satellite images provide useful information, but, in many cases, the clouds or the fire smoke itself do not allow to have a good estimation of the fire front position. So, a methodology based on thermal anomalies information has been applied to estimate fire perimeters even on adverse conditions. This methodology has been recently added to EFFIS to determine the perimeter evolution of a wildfire using thermal anomalies information coming from NASA's satellites. In this paper we describe the algorithm to perform this task and how it was implemented and integrated into the EFFIS services.

Index Terms—Fire perimeter, Wildfire, MODIS, EFFIS, hotspots

I. INTRODUCTION

Wildfires are environmental disasters that produce the loss of thousand hectares of forest, millions of dollars in material damage and, in the worst case, loss of human lives every year. At a local scale the physical properties [1] and chemical composition [2] of the burned area are changed and CO₂ and water vapor emissions affect the local ecosystem [3]. At global scale wildfires also affect the vegetation dynamics and the carbon cycle [4]. They are an important source of greenhouse effect gasses [5] and have a significant impact on air pollution [6]. Accurate estimation of the burned area and the position of the fire front are crucial issues to predict the potential damage and to fight against fire propagation in the best possible way. Particularly, the fire perimeter is needed for several tasks:

- **Survey of burned areas:** several projects at different scale use satellite information to monitor burned area to estimate forest degradation [7] and forest inventory [8]. Burned areas can be directly used to update this kind of information. The general goal is to maintain an accurate and updated information about the situation of forests at European and, even, global scale.
- **Smoke dispersion analysis produced by wildfires:** an accurate estimation of the burned area is needed to

determine the amount of smoke produced due to the burning of the fuel [6]. For this kind of analysis more than one perimeter is needed to infer the rate of smoke generation and the evolution at different times.

- **Online fire propagation prediction systems:** several projects are willing to create operational frameworks to estimate and predict fire propagation [9], [10]. In these projects it is critical to have the fire perimeter to launch an automated fire propagation simulation to predict fire evolution.
- **Fire perimeter assimilation for parameter calibration:** Usually, there is a certain degree of uncertainty in forest fire model propagation input parameters. To reduce this uncertainty, a common approach is to carry out a two stage propagation prediction [11], [12]. In a first stage, the input parameters are calibrated by reproducing the observed behaviour of the fire perimeter. The fire perimeters at two different times t_0 and t_1 are used to estimate the fire evolution between these time instants. The fire perimeter at time t_0 is used as the starting fire front and some optimization technique is used to determine the set of values of the input parameters that best reproduces the propagation from t_0 to t_1 . The obtained values for the input parameters in this calibration stage are used in the prediction stage to predict the propagation between t_1 and t_2 . When new information about the evolution of the fire is available the calibration process is repeated. A framework implementing this method is presented in [9].

Many studies have relied on satellite information to obtain fire perimeters: Mitri et al. [13] used images from LANDSAT and segmentation techniques to map burned areas while Gitas and Mitri [14] developed an object-based image classification using NOAA imagery. Other authors, such as Fraser [15], use images from the non-visible spectrum information in conjunction with common satellite images. However, good weather conditions in the moment the visible spectrum images are taken, is a prerequisite for them to be useful. If the obtained image presents cloud cover or smoke generated by the fire itself, to obtain the fire perimeter of the ongoing wildland fire becomes a very difficult task and, in many times, it results impossible. However, since determining the

area affected by the fire is mandatory to be able to manage and analyze different post-fire effects, alternative strategies have been developed. Most of these perimeter determination approaches are post-fire analysis of the visible images obtained by the satellites. These techniques require human intervention and imply a delay (sometime days) in delivering the final burnt area. When dealing with, for example, smoke analysis dispersion due to a wildfire or real-time forest fire spread prediction, the lack of a real-time fire perimeter is a hard constraint. A good alternative to solve this problem is to use the current available non-visible spectrum information to automatically generate real-time forest fire perimeters.

Terra and Aqua NASA satellites carry an instrument called Moderate Resolution Image Spectroradiometer (MODIS) which produces different fire products [16]. One of these products is the hotspot information, which represents temperature anomalies registered by the middle and thermal infrared sensors. Hantson et al. [17] found out that the information retrieved from MODIS hotspots data is closely associated with true burned areas with a low number of errors. This correspondence is increased in areas with medium to intensive fire activity. However, these thermal anomalies can only be detected while the fire is active and do not remain in time.

The European Forest Fire Information System (EFFIS) is a comprehensive system covering the full cycle of forest fire management including prevention and damage analysis. The main components of EFFIS are an ORACLE relational database manager containing information from MODIS Terra and Aqua rendered at a pan-European level to 250 m resolution and web-based tools with services for maps, features and coverage. A full description of this system can be found in [18]. One of the products stored in the EFFIS database is the hotspots information provided by the MODIS instrument aboard the Terra and Aqua satellites. As we have previously mentioned, this data can be extremely useful to determine forest fire perimeters but, currently, there is no automatic tool that process such product with this purpose. In this paper, we propose a methodology to automatically generate fire perimeters using MODIS hotspots information by retrieving temperature anomalies from EFFIS georeferenced database. The proposed methodology can be used either to obtain the real time fire evolution for an ongoing wildland fire or to obtain the intermediate fire perimeters for a past fire whose associated hotspot data have been stored in EFFIS without being processed.

This paper is structured as follows: The used methodology to extract fire perimeters from hotspots is described in section II. The implementation of the fire perimeter determination in EFFIS is described in section III. This methodology has been applied to two wildfires occurred in Sweden in 2014 and Spain in 2012. The results obtained are presented in section IV.

II. EXTRACTING FIRE PERIMETER FROM HOTSPOTS USING α -SHAPE ALGORITHM

In this section, the methodology applied to extract fire perimeters from hotspots is described. This methodology is

based on the α -shape algorithm. This algorithm is a generalization of the convex hull algorithm of a finite set of points in the plane proposed by Edelsbrunner et al. [19]. This generalization produces a family of graphs formed by straight lines called α -shapes. These shapes capture different levels of detail of the resulting shape depending on the value of α and they are closely related to the Delaunay Triangulation [20] and the alpha complex of the set of points. The algorithm relies on a series of mathematical results that are beyond the scope of this work, but some of them are needed to describe the proposed algorithm:

Definition 1. Let S be a set of points and $T \subset S$ with $|T| = k + 1 < d + 1$, the polytope $\Delta_T = \text{conv}(T)$ has dimension k . Then Δ_T is a k -simplex.

where $\text{conv}(T)$ is the convex hull of the set of points T . For each k -simplex we will say that is α exposed if:

Definition 2. There exists an empty circumference of radius α with $T = \partial b \cap S$, where ∂b is the boundary of the circumference.

Figure 1 shows the difference between an α -exposed and a non α -exposed simplex. In Figure 1a, it can be observed that at least one of the circles of radius α that pass through the two selected points does not include any other point in the set of points. In Figure 1b, both circles of radius α passing through the two selected points include other points of the set.

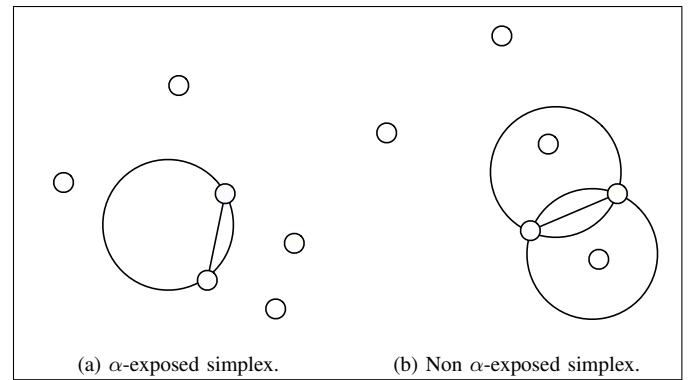


Figure 1: Difference between α -exposed simplex and not exposed simplex.

Lema 1. Let $DT(S)$ be the Delaunay Triangulation of a set of points in the plane. The α -shape S_α of S is a subgraph of $DT(S)$.

Definition 3. The α -complex $C_\alpha(S)$ is a subset of $DT(S)$ formed by simplices. A simplex $\Delta_T \in DT(S)$ is in $C_\alpha(S)$ if:

- 1) its circumference is empty and has radius smaller than α .
- 2) Δ_T is a face of another simplex in $C_\alpha(S)$.

Observation 1. The boundary $\partial S_\alpha(S)$ of S_α consists on all the k -simplices which are α exposed for $0 < k \leq d$: $\partial S_\alpha = \{\Delta_T \text{ with } |T| \leq d \text{ and } \Delta_T \text{ } \alpha\text{-exposed}\}$

Observation 2. $\partial S_\alpha(S) = \partial C_\alpha(S)$.

Different implementations of the α -shape are available and described in [21]. Here, we rely on the above results and use the algorithm that found the α -exposed simplices in the Delaunay Triangulation. This implementation is only valid for points $S \in \mathbb{R}^2$. Then, the α -shape can be computed using the pseudocode shown in the algorithm 1. In Figure 2a the Delaunay triangulation is shown. Figure 2b represents the elimination process of the α -exposed simplices. The circumference of the 2-simplex has a radius smaller than α and the simplex is part of $C_\alpha(S)$ while the circle of radius α that passes through the points of the 1-simplex is not empty, then the edge e is not part of ∂S_α and is eliminated. In Figure 2c the result of the algorithm is shown.

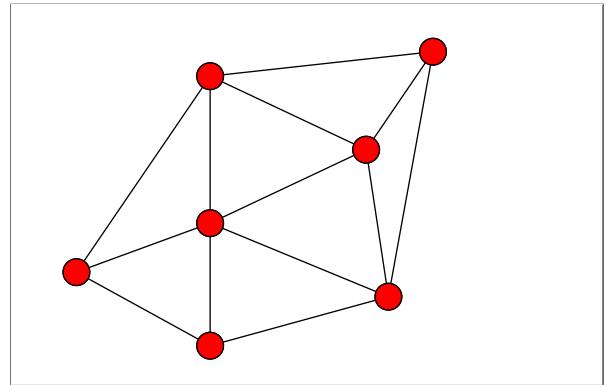
Algorithm 1 α -shape computation

```

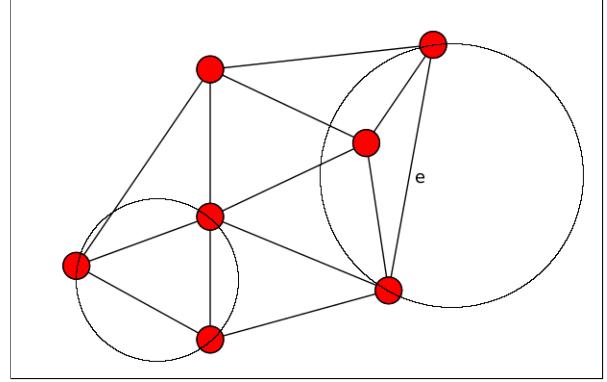
1: procedure :
2:    $\alpha \leftarrow$  value
3:    $\alpha\text{-Set} \leftarrow$  Empty set
4:    $C_\alpha(S) \leftarrow$  Empty set
5:    $DT(S) \leftarrow$  compute Delaunay triangulation of  $S$ 
6:   Compute  $C_\alpha(S)$ :
7:   for  $2 - \text{simplex} \in DT(S)$  do
8:      $R \leftarrow$  circumcircle of simplex
9:     if  $R < \alpha$  then
10:       Add simplex to  $C_\alpha(S)$ 
11:     Find  $\partial C_\alpha(S)$ :
12:     for  $1 - \text{simplex} \in C_\alpha(S)$  do
13:       Calculate the  $\alpha$ -ball which contains the vertices of
          the 1-simplex.
14:       if one  $\alpha$ -ball is empty then
15:         Add edge to  $\alpha - \text{Set}$ 
```

III. IMPLEMENTATION IN EFFIS

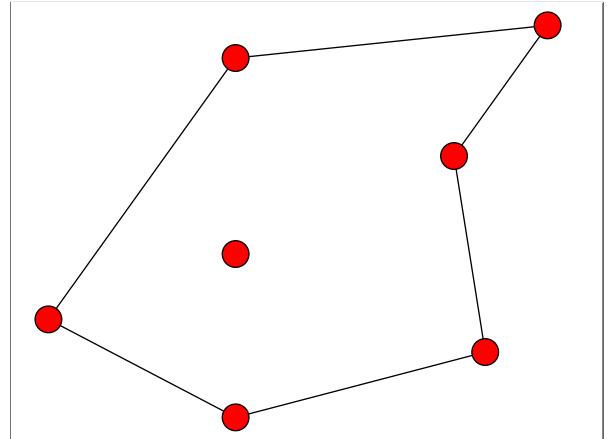
The EFFIS (European Forest Fire Information System) system supports the services in charge of the protection of forest fire in the EU countries and provides the European Commission services with updated and reliable information on wildland fires in Europe. In particular, the EFFIS database stores information from MODIS Terra and Aqua NASA's satellites rendered at a pan-European level to 250m resolution. Part of this data are the hotspots, which are collected and stored without any automatic tool available to process them. The algorithm described in the previous section conforms the basis of an automatic tool to retrieve fire perimeters from the available hotspots information. The proposed methodology can be used in two different ways: on the one hand, it can provide at near real-time the active front of an ongoing forest fire, and, on the other hand, it can be used to enhance the stored information of past fires by finding out intermediate fire perimeters in those cases where the visibility was null at the moment of the event, and no image processing was performed.



(a) Delaunay triangulation of a set of points.



(b) Elimination of the α -exposed simplices. The circle with smaller radius is the circumcircle of the 2-simplex formed by three edges; since the radius is smaller than α (represented by the bigger circle) the simplex is part of $C_\alpha(S)$. On the other hand the edge e is eliminated from ∂S_α since the α -ball passing throw the extreme points of e is not empty.



(c) Result of the α -shape algorithm.

Figure 2: Fire perimeter extraction using α -shape algorithm

A. MODIS Hotspot

The Moderate Resolution Imaging Spectroradiometer (MODIS) sensors are aboard Terra and Aqua NASA's satellites and collect information of the whole surface of the earth. The information is collected in 36 spectral bands ranging

from 0.4 to 14.4 μm with resolutions from 250 to 1000 m. Hotspots are determined evaluating the temperature of each pixel (based on the wavelength of the reflected light) and thermal anomalies are considered with a threshold of 310 K ($4\mu\text{m}$) during daytime and 305 K at nighttime [22].

These pixels, considered as thermal anomalies, have also a confidence value assigned and can be rated as:

- High confidence (if the confidence of the pixel is above 80%).
- Medium confidence (above 30 % and bellow 80 %).
- Low confidence (bellow 30%).

This confidence interval depends greatly on the observing conditions. Since the thermal anomalies provided by MODIS represent the center of a 1 km^2 pixel, ideally, the size of the fire that can be detected ranges from 50 m^2 in pristine conditions to 1000 m^2 in regular conditions. However, using the current operational service, we assume that fires smaller than 1000 m^2 cannot be detected.

In this work, the MODIS hotspots used are the ones available in EFFIS database, which are downloaded from the German AeroSpace Center (DLR) receiving station. The confidence level of the hotspots is not taken into account since spurious anomalies are going to be automatically discarded by the implemented algorithm (widely explained later on in this section). A crucial point for being able to provide near real time fire perimeters using hotspots, is the capacity of having access to the MODIS thermal anomalies at near real time too. This so called NRT processing of the MODIS hotspots is possible because the MODIS instruments on board of the Terra and Aqua satellites have the technical feature of Direct Broadcast capability together with an on board data storage and a transmission capability. This mechanism allows the system to immediately transmit to the ground all raw data collected, which is rapidly received by two DLR ground stations. Once the MODIS data is received, it is processed to obtain the thermal anomalies. This thermal information is collected twice daily by each sensor (Aqua and Terra) providing up to four thermal observation daily.

As a result of the described process, one obtains a set of points (hotspots or thermal anomalies), which are stamped with the day and hour that have been gathered. In this work, we describe a methodology to process the obtained cloud of hotspots (set of point georeferenced) in order to extract the corresponding fire perimeter and the evolution in time of a forest fire. In the next section, the algorithm used as a core of the proposed methodology is described.

B. Fire perimeter extraction in EFFIS

Before computing the α -shape algorithm in EFFIS, some previous steps must be performed in order to find the set of points required for the algorithm. In our case, as it has been stated above, this set of points corresponds to the MODIS hotspots associated to a certain wildland fire. Hotspots give useful information if the fire is active at the moment the satellite acquires the image. Moreover, big fires evolve during more than one day and due to the fuel exhaustion the active

fire is not necessarily the whole burned area. Therefore, to produce a reliable perimeter we must integrate the information acquired during the time window in which the fire was active.

To narrow the spatial window in which hotspots are retrieved, we make a query on the EFFIS fire database and get the approximate area of the active fire. Since the resolution of MODIS hostspots are of 1 km^2 , the minimum fire we can detect with this method is 500 ha. (at least 5 hotspots in the same cluster as it will be later explained). If the burned area is bigger than this value, we use it to calculate the size of a square window with equation 1.

$$d = \sqrt{FireArea} \quad (1)$$

This distance is transformed to geographical coordinates using the naive approximation shown in equation 2. A factor f to increase the size of the window is used (typically $f = 4$) as a safety factor, since the shape of the fire can significantly differ from a square.

$$d_{geo} = fd180^\circ / (64000[hm]\pi) \quad (2)$$

The spatial window is centered in the centroid of the fire and the extreme coordinates of it are given by equations 3.

$$\begin{aligned} lat_{min} &= lat_{centroid} - d_{geo}/2 \\ long_{min} &= long_{centroid} - d_{geo}/2 \\ lat_{max} &= lat_{centroid} + d_{geo}/2 \\ long_{max} &= long_{centroid} + d_{geo}/2 \end{aligned} \quad (3)$$

Finally, it must be considered that multiple fires could exist in the same area. Then, given a set of points, we have to define a maximum distance to consider two points as neighbors of the same set (i.e. belonging to the same fire). With this distance we perform a clustering analysis as shown in figures 3a and 3b. In this step, spurious anomalies are eliminated.

The last step is to apply the alpha shape algorithm to each founded cluster. If the number of elements in the cluster is less than three no perimeter is generated. If the cluster has exactly three elements the α -shape algorithm is reduced to the convex hull and the later one is applied. For the rest of the clusters a Delanuay triangulation is performed (figure 3c) and the α -shape algorithm is applied iteratively with increasing α 's until an α value is found for which all the elements of the cluster remain connected. Finally, the result of applying this algorithm is a shape file of the obtained perimeter. The result of this algorithm is shown in figure 3d. The pseudocode of the implemented methodology is shown in algorithm 2.

In the subsequent section, the proposed methodology is applied to two different cases.

IV. EXPERIMENTAL RESULTS

In order to test the proposed methodology, we have selected two different past fires from the EFFIS database. The first case studied (Vastmandlands Ian fire - Sweden) exemplifies the situation of delivering fire perimeters at a near real-time.

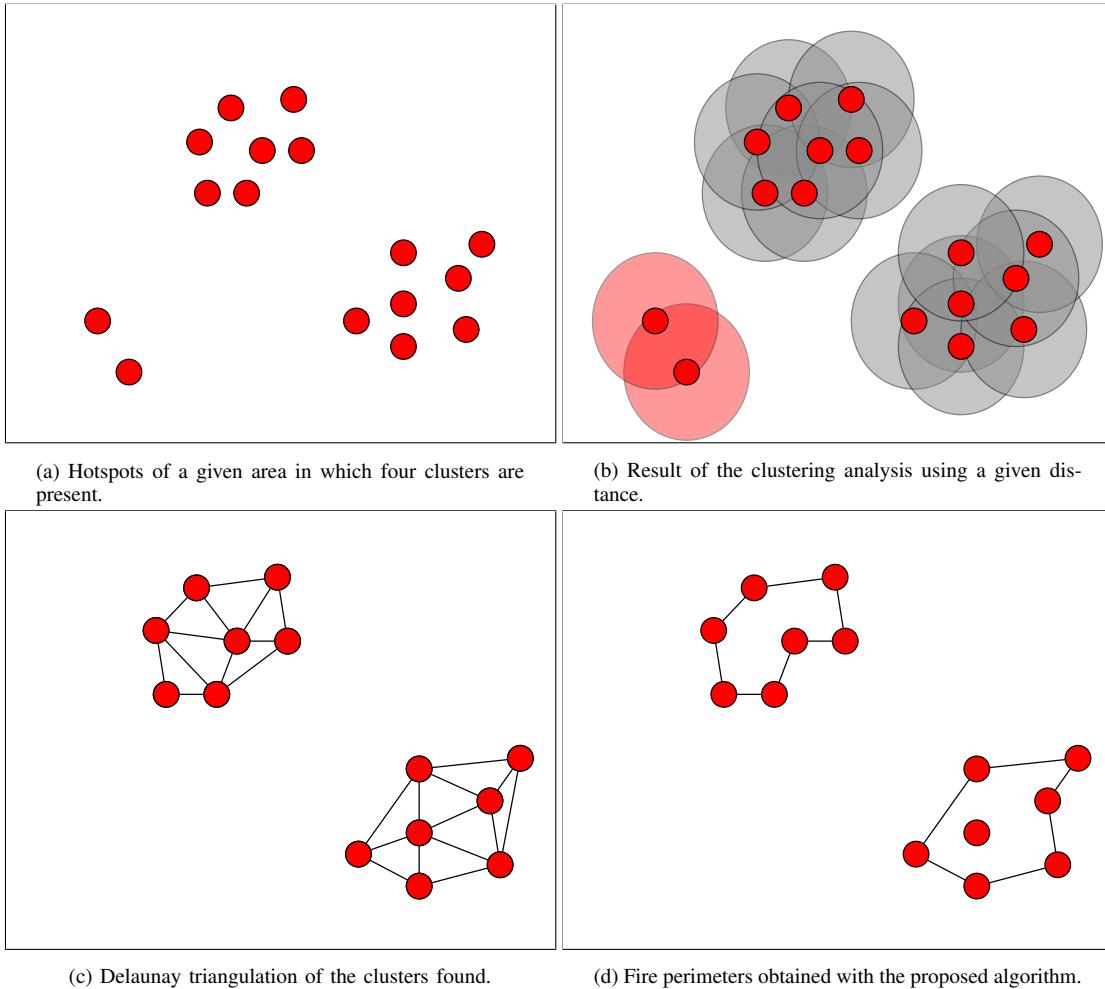


Figure 3: Steps performed in EFFIS to generate the fire perimeters.

Algorithm 2 Implementation in EFFIS

```

1: procedure :
2:    $F \leftarrow$  Empty set for active fires.
3:    $F \leftarrow$  Find in EFFIS DB all active fires.
4:   for each fire in  $F$  do
5:     if  $\text{Area}(\text{fire}) > \text{threshold}$  then
6:       Calculate temporal window
7:        $d \leftarrow \sqrt{\text{Area}(\text{fire})}$  Calculate spatial windows
8:        $H \leftarrow$  Retrieve hotspots for the spatial and
       temporal window
9:        $C \leftarrow$  Perform a clustering analysis
10:      for each cluster in  $C$  do
11:         $\alpha \leftarrow$  Initial Value
12:        while cluster disconnected do
13:          Apply  $\alpha$ -shape.
14:          Increase  $\alpha$ .
15:      Write output perimeter.

```

The other case corresponds to a wildland fire that took place in Valencia (Spain) in 2012. In this case, there were no intermediate fire perimeters stored in EFFIS, instead there were only the final burnt area. In both cases, the proposed hotspots methodology to obtain forest fire perimeters is applied and, when possible, the obtained fire propagations have been compared to the burn area stored in EFFIS.

A. Västmanlands Ian fire

This wildfire occurred in the province of Västmanlands Ian, Sweden. It started the 31 of July of 2014, ending the 5 of August and it burned 12484 ha of forest according to EFFIS database. The available information in this database is shown in table I. The corresponding fire perimeters are shown in figure 4. As it can be observed, in this case, the data retrieved from EFFIS is only from 3 days, whereas the fire lasted 6 days, therefore, there are some missing fire perimeters.

In order to apply the proposed strategy, the hotspots acquired by MODIS in the period the fire took place were retrieved from EFFIS database. This information is shown in figures 5a, 5b and 5c and it has been grouped every two days to simplify the visualization. After retrieving the hotspots, the

Table I: Evolution of the fire produced in Västmanlands Ian, Sweden.

Date	Burned Area[ha]
2014-08-03	1438
2014-08-04	3402
2014-08-05	12484

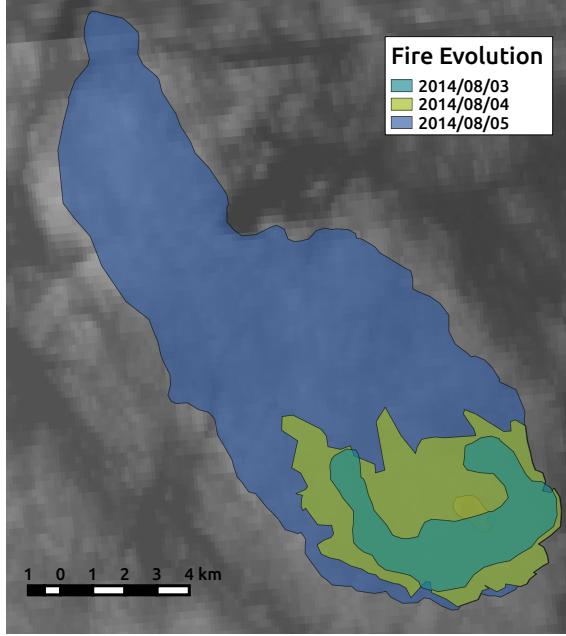


Figure 4: Evolution of the Västmanlands Ian fire obtained from EFFIS database.

perimeter detection algorithm has been applied to the whole set of hotspot points. The obtained fire fronts are shown in figures 6a, 6b and 6c. Applying this methodology two perimeters that were not available in EFFIS database could be retrieved enhancing the information associated to that forest fire giving the possibility of improving the post-fire analysis in this case. In order to verify the correctness of the proposed methodology, the forest fire perimeters obtained using hotspots have been compared to the burn area stored in EFFIS (only the perimeters that were available). The difference between the perimeters obtained from hotspots and the perimeters available in EFFIS database are compared in figures 7a, 7b and 7c. This figures reveal the agreement between the perimeter obtained by both methods. Finally, the areas calculated from the hotspots perimeters are shown in table II. It can be seen, that the total burned area (05/08) differs in 1% with the reported by EFFIS, but for the intermediate perimeters there is less agreement with differences of 10 and 36 %. This can be explained due to the lack of detail produced by the low resolution of the hotspots. The bigger the fire the less notorious this differences are expected to be. Nevertheless, this difference in area can be neglected since the hotspot methodology gives information almost instantly after the information is registered while the perimeters available in EFFIS are processed manually and have a delay of a few days.

Table II: Burned areas obtained from hotspots and its difference with the area in EFFIS database for the Västmanlands Ian fire.

Date	Burned Area[ha]	Difference with EFFIS [%]
2014-08-03	1292	-10
2014-08-04	4642	36
2014-08-05	12334	-1

B. Valencia fire

This fire occurred in Valencia, Spain. It started on June 28 of 2012 and ended 4 days later. The total burned area was 32424 ha according to EFFIS database. In this case no information about the evolution of the fire was available due to the presence of clouds and smoke as can be seen in figure 8. The burned area was obtained days after the fire was extinguished and is shown in figure 9.

The hotspots acquired by MODIS in the period the fire took place are shown in figures 10a and 10b (grouped every two days to make visualization easier).

The active fire for each day can be found using the proposed algorithm. To obtain this, the temporal integration step is not performed and only the available hotspots for each day were used. In figure 11 the perimeters representing the active fire are shown. This information would not be available without using the MODIS hotspots.

The evolution of the fire perimeter obtained using the α -shape algorithm is shown in figures 12a and 12b. With this procedure the complete evolution of the fire during the four days was reconstructed.

In this case, only the final burned area can be compared since no fire evolution is available in EFFIS. In figure 13, a comparison between the perimeter obtained for the Valencia fire on 1/07/2012 using the α -shape algorithm and the available perimeter on EFFIS is shown. Again, it can be seen that a very good agreement exists between both perimeters.

In table III, the evolution of the burned hectares founded with the proposed algorithm is shown. When EFFIS information is available the area is compared with the one obtained from the hotspots. The difference in this case for the total area is around 7%.

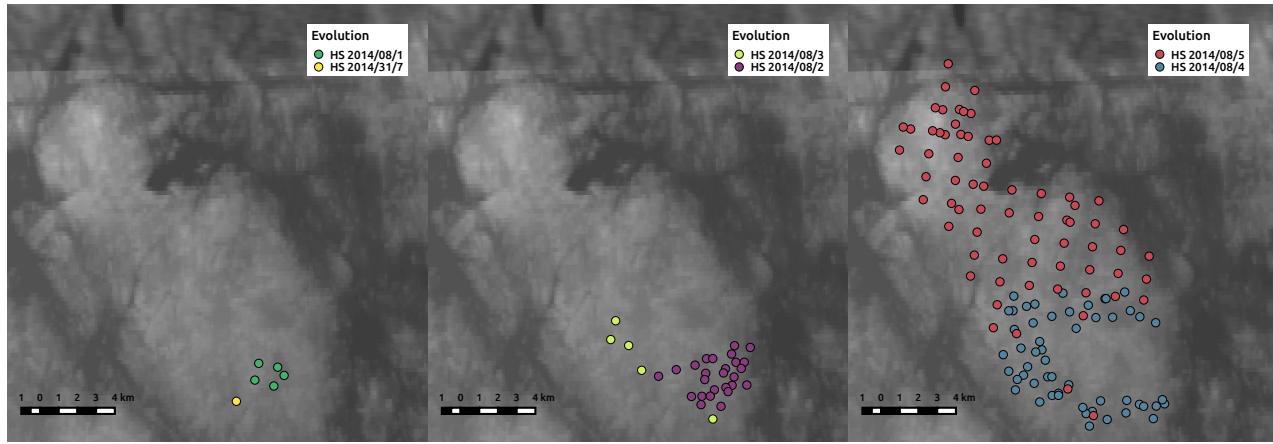
Table III: Burned areas obtained from hotspots for the Valencia fire and its difference with the area in EFFIS database.

Date	Burned Area[ha]	Difference with EFFIS [%]
2012-06-28	7629	-
2012-06-29	23948	-
2012-06-30	33539	-
2012-07-01	34661	6.8%

V. CONCLUSION

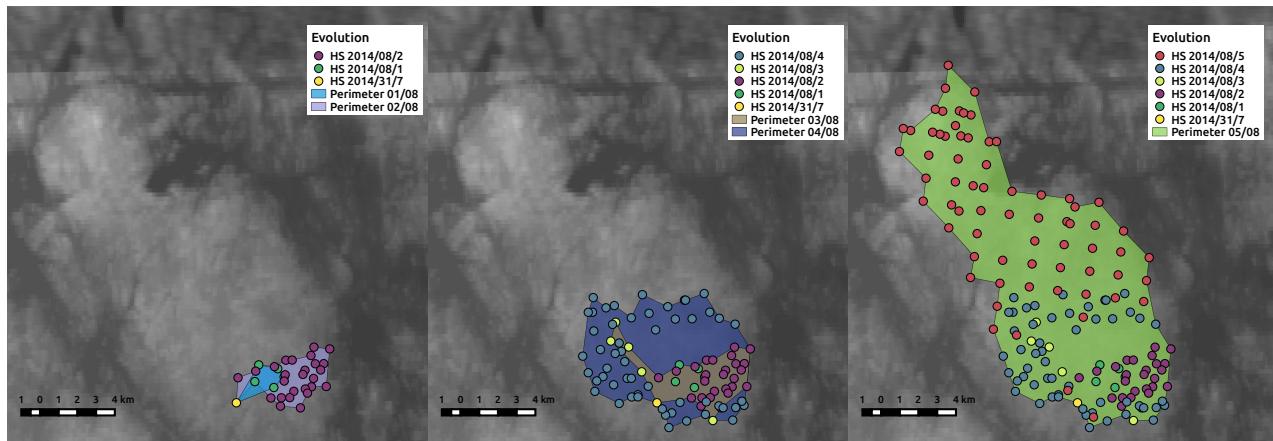
A methodology to find fire perimeters from MODIS hotspots information that relies on the α -shape algorithm has been described in this work. This methodology was implemented in EFFIS.

This algorithm was used to analyze a fire occurred in Sweden in 2014 and allowed us to find fire perimeters during



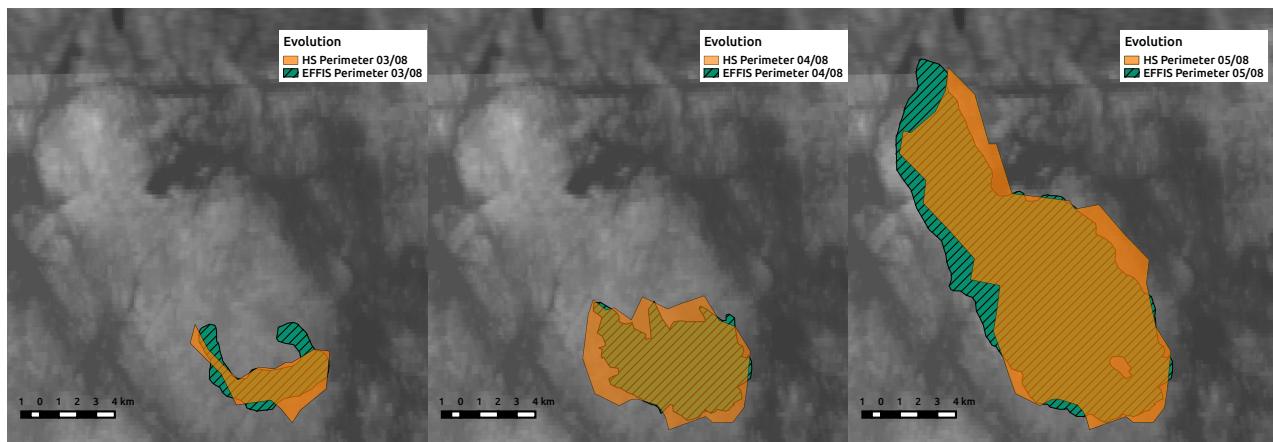
(a) Hotspots acquired during 31/07 to 1/08. (b) Hotspots acquired during 2/08 to 3/08. (c) Hotspots acquired during 4/08 to 5/08.

Figure 5: Hotspots acquired from MODIS during the Vastmandlands Ian fire.



(a) Fire perimeter on 1/08 and 2/08. (b) Fire perimeter on 3/08 and 4/08. (c) Fire perimeter on 5/08.

Figure 6: Fire perimeters obtained using the described α -shape algorithm using hotspots retrieved from MODIS for the Vastmandlands Ian fire.

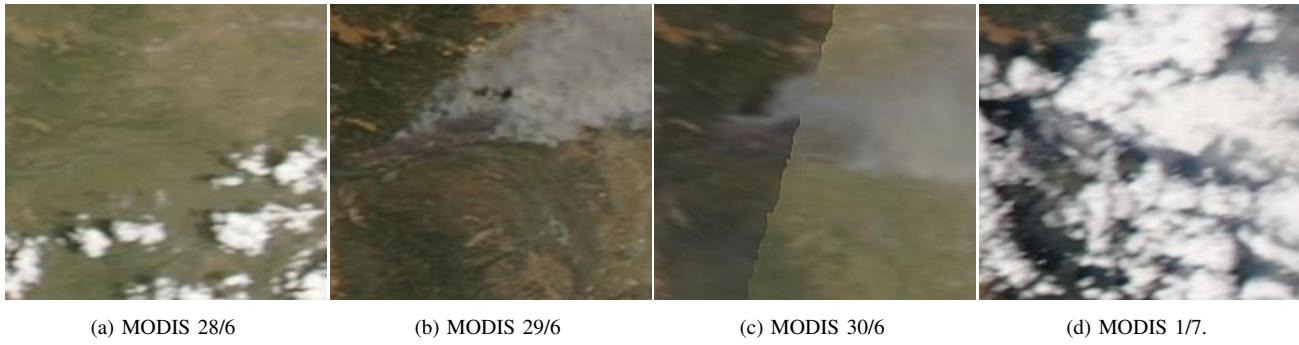


(a) Perimeter from 03/08.

(b) Perimeter from 04/08.

(c) Perimeter from 05/08.

Figure 7: Difference between perimeters available in EFFIS database and the obtained with MODIS hotspots for the Vastmandlands Ian fire.



(a) MODIS 28/6 (b) MODIS 29/6 (c) MODIS 30/6 (d) MODIS 1/7.

Figure 8: Corrected reflectance MODIS images obtained during the Valencia wildfire.

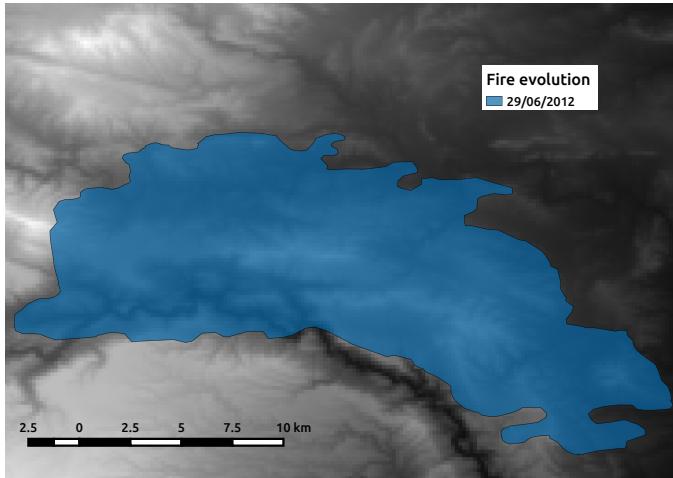


Figure 9: Valencia fire burned area obtained from EFFIS database.

the first three days of the fire, when no information at all was available in EFFIS database. This provides a very important tool to track the evolution of the fire.

Furthermore, the resultant shapes were compared with those available in EFFIS database with good agreement among both methods. We found that the perimeters obtained from image processing (available in EFFIS) are more detailed since the resolution of the images used to create the perimeters is higher than the hotspots resolution. Nevertheless the hotspot methodology does not require to process any image manually and functions in an automated manner.

The final total burned area differs in 1% but for the intermediate perimeters the differences are as high as 36%. This is consistent with the resolution of the hotspot pixel.

The same methodology was applied to a fire occurred in 2012 in Spain. In this case, the only available information was the final total burned area, obtained 3 days after the fire ended. Using the methodology proposed a daily evolution of the burned area could be obtained.

Also, we found the active fire area for each day using hotspots information without considering the temporal integration. In this case, the total burned area differed on 7% with

the total fire area available in EFFIS database.

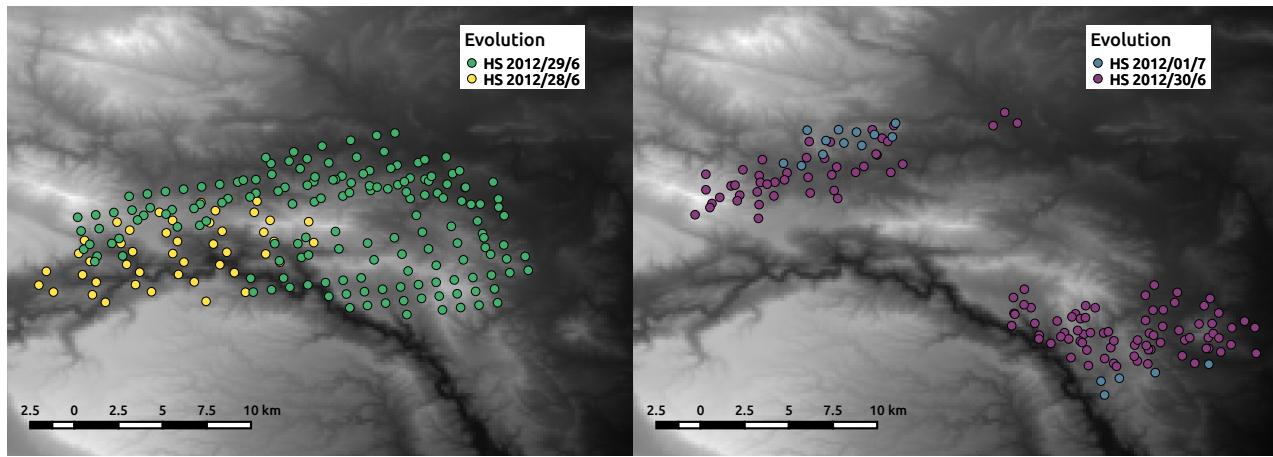
We consider this methodology to be a valuable asset to the EFFIS system, allowing to automatically track fire evolution without the need of manual processing. Also, the perimeters can be used as ignition perimeters for prediction systems incorporated in EFFIS. On a further note, the same algorithm could be applied to the set of VIIRS hotspots to increase the resolution of the perimeter and to lower the fire detection threshold to 60 ha.

ACKNOWLEDGMENT

This work was founded by the Ministerio de Economía y Competitividad of Spain under contract TIN2014-53234-C2-1-R and developed under the collaboration between the Institute for Environment and Sustainability of the Joint Research Centre of the European Commission and the High Performance Computing Applications for Science and Engineering research group of the Universitat Autònoma de Barcelona.

REFERENCES

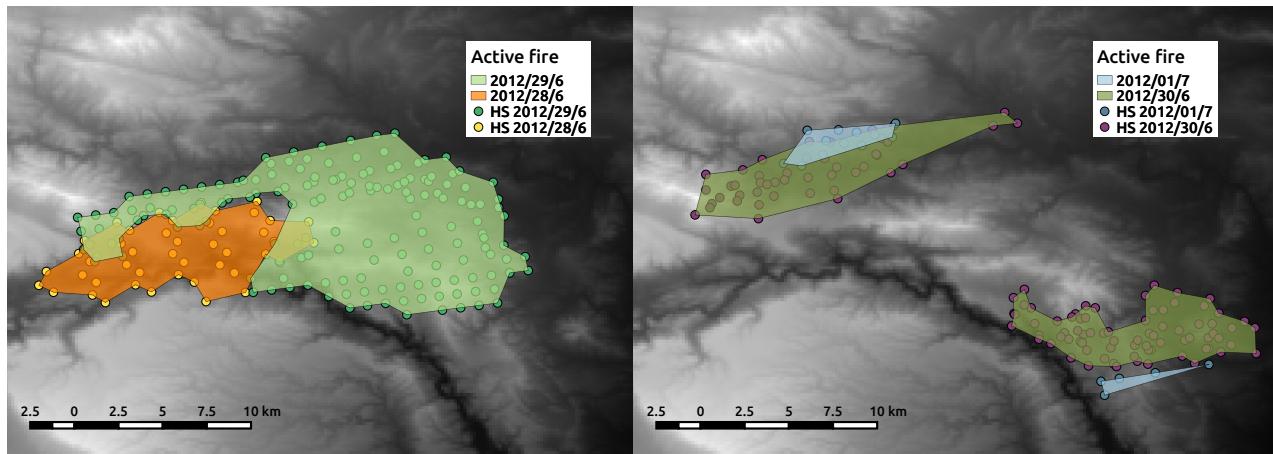
- [1] D. Scott and D. Van Wyk, "The effects of wildfire on soil wettability and hydrological behaviour of an afforested catchment," *Journal of Hydrology*, vol. 121, no. 1-4, pp. 239–256, dec 1990.
- [2] E. Thiffault, K. D. Hannam, S. A. Quideau, D. Paré, N. Bélanger, S.-W. Oh, and A. D. Munson, "Chemical composition of forest floor and consequences for nutrient availability after wildfire and harvesting in the boreal forest," *Plant and Soil*, vol. 308, no. 1-2, pp. 37–53, apr 2008.
- [3] D. Obrist, E. H. Delucia, and J. A. Arnone, "Consequences of wildfire on ecosystem CO₂ and water vapour fluxes in the Great Basin," *Global Change Biology*, vol. 9, no. 4, pp. 563–574, apr 2003.
- [4] B. J. S. Eric S. Kasischke, N. L. Christensen, "Fire, global warming, and the carbon balance of boreal forests," *Ecological Applications*, vol. 5, no. 2, pp. 437–451, 1995.
- [5] T. M. Bonnicksen, "Greenhouse gas emission from four California Wildfires : Opportunities to prevent and reverse environmental and climate impacts," vol. 95603, no. 530, 2008.
- [6] B. Langmann, B. Duncan, C. Textor, J. Trentmann, and G. R. van der Werf, "Vegetation fire emissions and their impact on air pollution and climate," *Atmospheric Environment*, vol. 43, no. 1, pp. 107–116, 2009.
- [7] A. Langner, J. Miettinen, and F. Siegert, "Land cover change 2002–2005 in Borneo and the role of fire derived from MODIS imagery," *Global Change Biology*, vol. 13, no. 11, pp. 2329–2340, nov 2007.
- [8] F. Maselli, M. Chiesi, M. Mura, M. Marchetti, P. Corona, and G. Chirici, "Combination of optical and LiDAR satellite imagery with forest inventory data to improve wall-to-wall assessment of growing stock in Italy," *International Journal of Applied Earth Observation and Geoinformation*, vol. 26, no. 1, pp. 377–386, 2014.



(a) MODIS hotspots 28-29/06/2012.

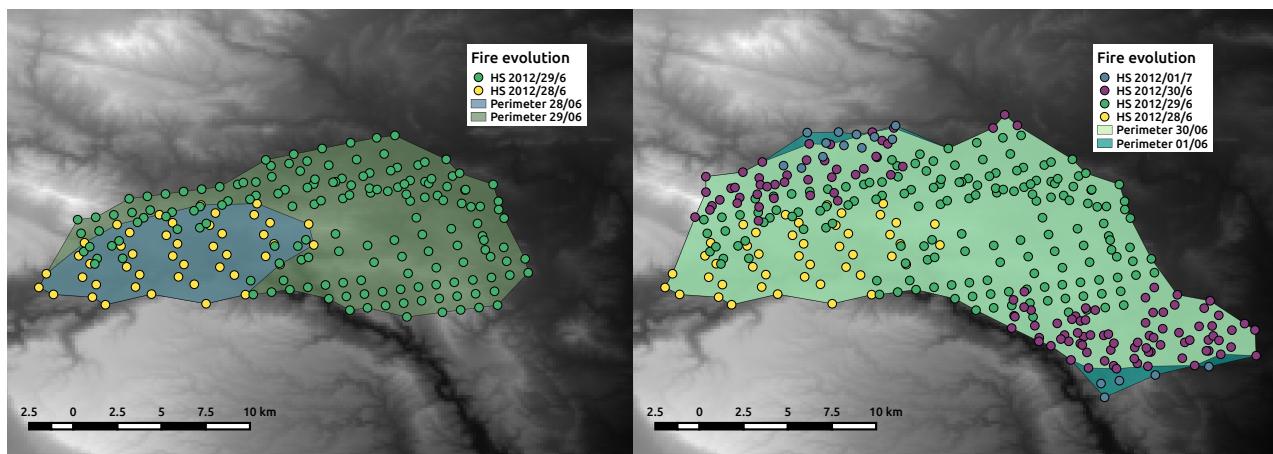
(b) MODIS hotspots 30/06-01/07/2012.

Figure 10: MODIS hotspots for the Valencia fire.



(a) Fire perimeter on 28 and 29 of June.

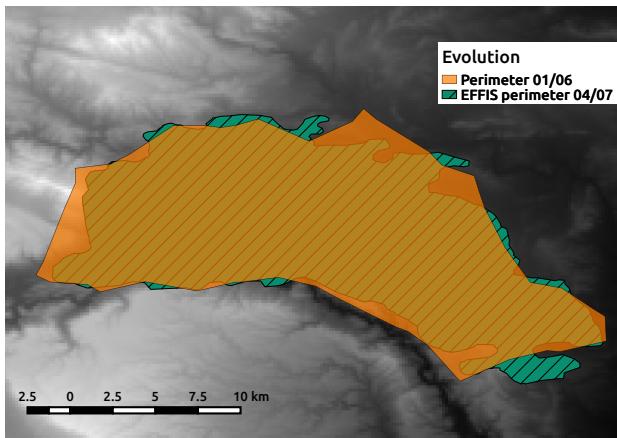
(b) Fire perimeter on 30 of June and 1 of July.

Figure 11: Active fire area obtained with the α -shape algorithm using MODIS hotspots for the Valencia fire.

(a) Fire perimeter on 28 and 29 of June.

(b) Fire perimeter on 30 of June and 1 of July.

Figure 12: Perimeters obtained with the α -shape algorithm using MODIS hotspots for the Valencia fire.



active fire detection algorithm and fire products," *Remote Sensing of Environment*, vol. 178, pp. 31–41, 2016.

Figure 13: Difference between perimeters available in EFFIS database and the obtained with MODIS hotspots for Valencia fire.

- [9] T. Artés, A. Cencerrado, A. Cortés, T. Margalef, D. Rodríguez-Aseretto, T. Petroliaikis, and J. San-Miguel-Ayanz, "Towards a dynamic data driven wildfire behavior prediction system at European level," *Procedia Computer Science*, vol. 29, pp. 1216–1226, 2014.
- [10] I. Altintas, J. Block, R. De Callafon, D. Crawl, C. Cowart, A. Gupta, M. Nguyen, H. W. Braun, J. Schulze, M. Gollner, A. Trouve, and L. Smarr, "Towards an integrated cyberinfrastructure for scalable data-driven monitoring, dynamic prediction and resilience of wildfires," *Procedia Computer Science*, vol. 51, no. 1, pp. 1633–1642, 2015.
- [11] B. Abdalhaq, A. Cortés, T. Margalef, and E. Luque, "Enhancing wildland fire prediction on cluster systems applying evolutionary optimization techniques," *Future Generation Computer Systems*, vol. 21, no. 1, pp. 61–67, 2005.
- [12] A. Cencerrado, A. Cortés, and T. Margalef, "Genetic algorithm characterization for the quality assessment of forest fire spread prediction," *Procedia Computer Science*, vol. 9, pp. 312–320, 2012.
- [13] G. H. Mitri, "The development of an object-oriented classification model for operational burned area mapping on the Mediterranean island of Thasos using LANDSAT TM images," *Fire Research*, no. Chuvieco 1997, pp. 1–12, 2002.
- [14] I. Z. Gitas, G. H. Mitri, and G. Ventura, "Object-based image classification for burned area mapping of Creus Cape, Spain, using NOAA-AVHRR imagery," *Remote Sensing of Environment*, vol. 92, no. 3, pp. 409–413, aug 2004.
- [15] R. Fraser, "Hotspot and NDVI Differencing Synergy (HANDS) A New Technique for Burned Area Mapping over Boreal Forest," *Remote Sensing of Environment*, vol. 74, no. 3, pp. 362–376, dec 2000.
- [16] L. Giglio, "MODIS Collection 6 Active Fire Product User's Guide Revision A," no. March, 2015.
- [17] S. Hantson, M. Padilla, D. Corti, and E. Chuvieco, "Strengths and weaknesses of MODIS hotspots to characterize global fire occurrence," *Remote Sensing of Environment*, vol. 131, pp. 152–159, apr 2013.
- [18] J. San-Miguel-Ayanz, E. Schulte, G. Schmuck, A. Camia, P. Strobl, G. Liberta, C. Giovando, R. Boca, F. Sedano, P. Kempeneers, D. McInerney, C. Withmore, S. S. de Oliveira, M. Rodrigues, T. H. Durrant, P. Corti, F. Oehler, L. Vilar, and G. Amatulli, "Comprehensive monitoring of wildfires in Europe: the European Forest Fire Information System (EFFIS)," 2012.
- [19] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane," *Information Theory, IEEE Transactions on*, vol. 29, no. 4, pp. 551–559, 1983.
- [20] B. Delaunay, "Sur la sphère vide," *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, vol. 7, no. 793-800, pp. 1–2, 1934.
- [21] H. Edelsbrunner, "Alpha Shapes - a Survey," *Tessellations in the Sciences*, pp. 1–25, 2010.
- [22] L. Giglio, W. Schroeder, and C. O. Justice, "The collection 6 MODIS

A columnar architecture for modern risk management systems

Romulo Goncalves¹, Sisi Zlatanova², Kostis Kyzirakos³, Pirouz Nourian², Foteini Alvanaki³, Willem van Hage¹

¹Netherlands eScience Center, The Netherlands

{r.goncalves,w.vanhage}@esciencecenter.nl

²TU Delft, The Netherlands

{s.zlatanova, p.nourian}@tudelft.nl

³CWI, The Netherlands

{kostis.kyzirakos,f.alvanaki}@cwi.nl

Abstract—3D digital city models form the basis for flow simulations (e.g. wind flow and water runoff), urban planning, under- and over- ground formation analysis, and they are very important for automated anomaly detection on man made structures. They consist of large collections of semantically rich objects which have many properties such as material and color. Such user's data structure perception is leading to complex storage schemas. The number of table relations to manage and the large data storage footprint drawbacks are then extended with the fact that not all the systems have a "real" 3D data type.

In this work we would like to show our efforts to develop a new kind of Spatial Data Management System (SDBMS) where topological and geometric functionality for 3D raster manipulation will become part of the relational kernel and not an add-on. With it spatial analysis tailored to different use case scenarios is done on-demand and fast enough to support real-time interaction in modern risk management systems.

I. INTRODUCTION

Digital 3D city models play a crucial role in research of urban phenomena; they form the basis for flow simulations (e.g. wind streams and water runoff), analysis of underground formations and man made structures which provide crucial information for effective risk management systems.

An urban scene, represented as a 3D city model, consists of large collections of semantically rich objects which have a number of properties such as use, function, and year. They are commonly reconstructed by segmenting and triangulating a point cloud thereby creating a surface representation. Representing urban objects (e.g. buildings, roads, trees, etc.) as surfaces has drawbacks while calculating intersections and volumes, and creating cross-sections is complex. Furthermore, modeling volumetric objects, such as walls, water, and underground, requires the deployment of complex shapes [18].

Such users data structure perception is leading to complex storage schemes. The storage scheme designed for systems like Oracle Spatial, Grass, and PostGIS has limitations such as the management of many tables when the selection predicate is on the 3D city model semantics. The number of table relations to manage and the large data storage footprint drawbacks are then extended with the fact not all the systems have a "real" 3D data type. PostGIS, highly adopted in eScience projects, is a clear example.

We have tackled all these issues by re-designing the conceptual model and the storage model. For conceptual model we have adopted a voxel-based city model, a path considered novel and promising [18]. Voxels are the volumetric representation of pixels. Alongside a length and a width, voxels also have height thereby forming a cube in 3D space. Voxel storage offers a number of interesting simplifications, use cases, but also challenges. One of the major challenges is its storage and efficient handling by Spatial Database Management Systems (SDBMSs). With different semantic level of detail (e.g., LOD in CityGML [16]) models and coverage of in- and out- side empty spaces, the voxelization of an entire city will generate a massive 3D grid of voxels at different resolutions with a large number of semantic attributes attached [18].

It is clear a dense flat relational table is not ideal to store such massive 3D grid. The holy grail is an architecture which allows effective compression to reduce storage footprint, and efficient data retrieval to access only the attributes of interest at a specific resolution. Such key features is what distinguishes a column-oriented architecture from a record-oriented architecture and the reason for their efficiency on analytic workloads [5].

Despite column-oriented architectures emerge as the right candidate and the efforts to extend them for spatiotemporal analysis over large data sets [8], [6], [12], [13], their flat storage model is not yet suitable to store a large 3D city model. To do so, we extended a column-store to also support a nested column-oriented storage for 3D city models. The chosen format is Parquet [1]. It is an effective storage model for sparse data sets with a nested structure (the different LODs). Its flat columnar format fits well the column-oriented programming model.

With our contribution, spatial analysis tailored to different use case scenarios is done on demand and fast enough to be used by modern risk management systems. The adopted storage model, Parquet [1], opens doors to also exploit state-of-the-art processing technologies, such as Spark, to scale out to country size. Furthermore, the simplicity of the conceptual model gives the opportunity to use interactive front-ends borrowed from gaming for real-time interaction with the surroundings.

The remainder of the paper is as follows. Section II describes the storage strategies and their challenges. Section III presents the general architecture. Section IV shows the steps already taken to put the vision in action. The article ends with future plans in Section V and a summary in Section VI.

II. BACKGROUND

In this section we do a top-down description of our solution, i.e., from the conceptual model to the storage model. For the conceptual model we first identify its advantages followed by the challenges in supporting it on current SDBMSs. For the storage model we give a description on the challenges in mapping a voxel-based conceptual model into a flat and nested column-oriented storage.

A. Voxels

Our world can be represented in voxels by gridding the 3D space and specifying what each cell represents by semantically "attaching" every cell/voxel to a real world object. Storing volumetric spaces such as air, water and underground is possible.

Every object is defined by set of voxels, with set's length depending on the level of detail (LOD). The storage unit base is a 3D voxel of certain size and each voxel's characteristics e.g. type (wall, glass, roof, door, etc.), color, density, etc. is then stored as a semantic property. Such data type atomicity avoids the use of a set of multiple geometries, approach currently used in other spatial RDBMSs to store 3D city models [18].

Representing real world objects by a single geometry type (3D cube) instead of collection of polygons/polyhedron greatly simplifies a range of geometric operations: volumes and areas are calculated by simply counting the number of voxels that form an object; 3D bisections become simple selection operations; dynamic Levels-of-Detail (LOD) as objects can be resampled with larger cubes [18].

B. Storage challenges

The storage and indexing of 3D voxels linked with properties, such as voxels created to simplify a point cloud, two approaches can be considered, a homogeneous voxel grid versus a heterogeneous voxel collection. The former allows for factorization of invariant properties from the data structures, while the latter is better suited to sparse models such as a 3D city model with different LODs.

A homogeneous voxel grid is easy to define using a flat relational schema, i.e., real-world objects are formed by semantically grouping voxels together via foreign key relations and relational views. The schema normalization is used to reduce the storage footprint at the cost of expensive spatial joins. The schema normalization storage footprint is proportional to the size of each voxel. Hence, efficient data access becomes dependent on efficient column compression techniques and effective storage of geometric empty spaces. The latter is very important because it strongly affects the data set size. If empty spaces were also materialized in the storage

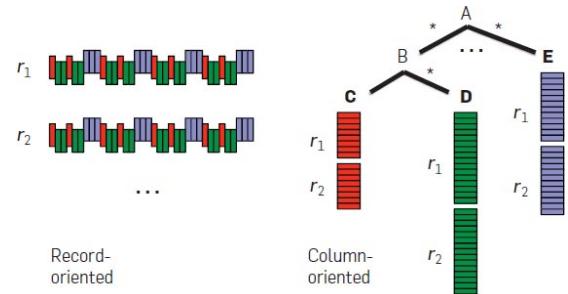


Fig. 1: "Record-wise versus columnar representation of nested data" [14] scheme, the storage of the whole of The Netherlands as e.g. 10 cm blocks would result in many petabytes of data.

A heterogeneous voxel grid poses extra challenges compared to a homogeneous voxel grid due to the preservation of the geometry semantics when converting vector to raster data. The object's semantics depends on the semantic level of detail (LOD) [18]. For example, the buildings LOD1 are *buildings*, LOD2 semantic is extended with *ground surface*, *wall surface*, and *roof surface*; LOD 3 has in addition to LOD 2 *window* and *door*; LOD4 *room*, *ceiling surface*, *interior wall surface*, *floor surface*, *closure surface*, *door*, *window*, *building furniture* and *building installation*. Hence, depending on the LOD, a voxel can have different semantic tags, e.g. (building, roof), (building, wall), (building, wall, window), etc. The LOD has a clear nested data organization and a sparse structure because of the in- and out- empty spaces. Furthermore, not all the levels in the nested structure are defined due to incompleteness or absence of vector information for a specific LOD.

C. Nested column-oriented storage

For efficient storage and data retrieval at different resolutions we embraced a column-oriented format for voxel-based 3D city models. Columnar formats have several advantages. Organization by column allows better compression, as data is more homogeneous. For large data sets the I/O is improved since it is possible to efficiently scan a subset of the columns while reading the data. Of course, better compression also reduces the bandwidth to read input data [4]. By storing together values of the same primitive type, a columnar format provides more efficient encoding and decoding.

Hence, to store nested data structures in flat columnar format, the schema is mapped to a list of columns in such a way that records are written and read back to its original nested data structure in an efficient way. Figure 1 illustrates the record-wise versus columnar representation of nested data. In the columnar representation all the values of a nested field are stored contiguously. For example, *A.B.C* can be retrieved without reading *A.E*, *A.B.D*, etc [14].

D. Parquet

In our work we use the well known Hadoop format called Parquet [1]. It stores nested data structures in a flat columnar format using a technique outlined in the Dremel paper from Google [14]. Parquet file layout is represented in Figure 2. Its internal structure is designed for efficient data skipping during

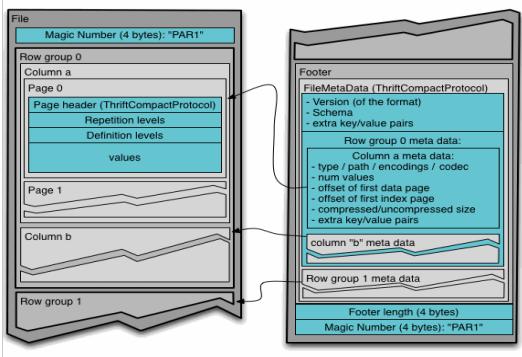


Fig. 2: Parquet file layout

query processing. The metadata stored in the file header and column-page header allows a kernel during predicate evaluation to skip data blocks and have lazy predicate evaluation over compressed data. At the same time, this metadata is used in our in-situ data access strategy, which is explained in Section III-B, to reduce the amount of data imported during query execution.

Parquet captures the record structure of each value through two integers called *repetition level* and *definition level*. During query processing they are used to fully reconstruct the nested structure¹.

Definition level. It is used to store the level of which the field is *NULL*. From 0 at the root of the schema up to the maximum level for the column. When a field is defined then all its parents are also defined. The definition level records at which level it started being null.

Repetition level. It is used to define when a new list starts in a column of values. It marks the level at which we have to create a new list for the current value.

Storing definition levels and repetition levels efficiently. For each primitive type it is necessary to store three sub columns. Due to the columnar representation the storage overhead is low. The depth of the schema defines the number of levels. For instance with 3 bits it is possible to store 7 levels of nesting. Required fields do not need definition level, and fields that are not repeated do not need repetition level.

Figure 3 represents the nested structure for a voxel-based city model. The LOD is used for the definition level. It is assumed that if a object has LOD2 semantics, it will also has LOD1 semantics, i.e., all the voxels inherit the semantics from the parent. The repetition level is the number of sub-divisions a parent voxel has. As an example, an object is semantically identified as a building in LOD1 while in LOD2 it might be composed by a set of sub-voxels to define walls, floor surface and etc.

III. A 3D RASTER SDBMS

A voxel-based 3D city model is best managed in a spatial DBMS as each voxel has a semantic relation to a real world object and various attributes (e.g. color, material, porosity, reflection properties, etc). Furthermore, a single spatial DBMS

¹For a detailed explanation with examples we recommend the read of <https://blog.twitter.com/2013/dremel-made-simple-with-parquet>

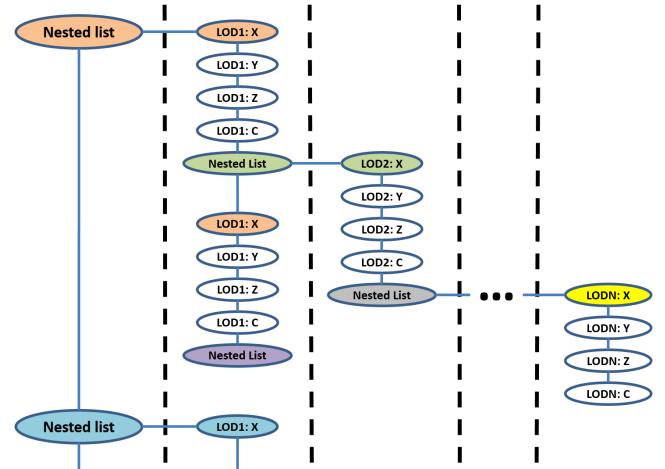


Fig. 3: LOD in Parquet

offers all functionality in one place, avoids the need for multiple software tools with associated high volume data transfer and format transformations.

During the last decade many DBMSs have been successfully extended with support for spatial and geo-spatial applications. For instance the OGC implementation specification, defining basic geometry types like points and polygons, is followed in PostGIS, Oracle, MySQL, Microsoft SQL Server, and MonetDB. To implement it they use their user-defined functions (UDF) functionality augmented in some cases with spatial search accelerators. However, contemporary DBMSs still lack advanced functionality and efficient implementations needed for analysis of voxel-based models.

We might argue that Oracle Spatial, Graphs 12c, and PostgreSQL 9.2 are developing extensions to support 3D geometries, even in GIS packages, only GRASS has support for voxels, but it still stores them as flat files. The systems are still in their infancy and they offer limited functionality. Due to the complexity of their software stack, deep integration with the database engine is even further away.

A. Column-oriented architecture

For our work we have extended a modern column-store, MonetDB [9], which steps away from traditional SDBMS which are all record-oriented architectures. Through vertical partitioning of relational tables column-store significantly reduce data access. In our case, vertical partitioning is exploited to reduce the number of columns to be imported as explained in Section III-B. Such data organization improves data compression, simplifies data skipping strategies and it suits well vector processing [4].

Currently through the works [8], [6], [12], [13], MonetDB spatial features have been matured to provide core technology components for geo-spatial big data analytics. Atomic spatial types and their operations are becoming part of the relational kernel and not an add-on. All the operations are available for spatial applications through integrated environments, such as R and Python, and a SQL front-end.

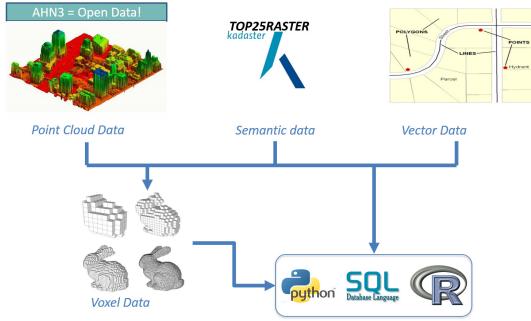


Fig. 4: MonetDB’s spatially enabled architecture

Currently the system is equipped with SQL primitives for building complex spatial analysis pipelines over 3D point clouds, more concretely: geometry-based selections (rectangular query window, 3D bounding box), attribute-based selections (point intensity, RGB, multi-spectral properties), conversion to Triangulated Irregular Network (TIN) and triangulation using constrained Delaunay. It also provides the option to export the results into a pre-defined format, such as X3D, GeoJSON and LAS/LAZ format, to be loaded into visualization tools.

In the context of 3D city models, MonetDB is currently being extended with SQL operations to manipulate voxel attributes: 3D selections (contains, within, intersects); regridding of homogeneous voxel grids; semantic categorization; volume based aggregation; and also rendering for interactive visualization tools such as Cubiquity [2], more details in Section V-B.

Our architecture, represented in Figure 4, is an attempt to couple under the same storage descriptive spatial data, such as point clouds, vector data and 3D rasters semantically enriched. It creates the grounds to have direct and on the fly conversion to a data type tailored to the type of user interaction.

B. Dynamic data access

The need of large area coverage and up to date information to support near real-time decisions was the reason for us to explore *in-situ* data access, i.e., data is kept in its original format while scalable and distributed processing functionality is offered through a DBMS.

Our work adopts the same strategy defined in [8] where the authors presented a solution for *in-situ* data access to large NetCDF data repositories. The work stands on the shoulders of previous work called data-vaults [10]. In this article we have extended it to support Shapefiles, Parquet and LAS/LAZ file format.

The *in-situ* data access is possible due to the large amounts of metadata (data of data) existent on file formats such as Parquet and LAS/LAZ formats. Such metadata is used for effective data skipping, but also to collect data insights, e.g. summaries and samples, without having to process the entire data set.

The dynamic data loading comprises of three phases: the attachment of a file, the import of the file’s content and the collection of statistics to boost query optimization. During the

attachment, the file’s metadata is loaded into a special DBMS catalog. At query time, such a catalog is inspected to decide whether the file has information relevant to the query. In such a case the file’s content is imported into the database, otherwise, it is not.

The data import happens in two ways, if the file format has each attribute sequentially stored then the import memory maps each attribute as a column, otherwise, the data is converted and loaded into the database as temporary data. In the latter case, cache policies, such as Least Recently Used (LRU), are used for data eviction.

IV. VISION IN ACTION

Our 3D raster SDBMS emerges from efforts in providing a scalable and generic solution for eScience projects with spatio-temporal data analysis. It is a continuous work standing on the shoulders of [8], [6], [12], [13]. In this section we summarize the steps taken towards a fully functional solution. The complete system evaluation is out of the scope of this article. An extended version with such evaluation will be submitted to a referee journal.

A. Voxel data Management

A 3D raster is commonly obtained from a existing 3D vector model or 3D discrete measurements such as point clouds. The vector model contains structured data and it is represented according to the rules of either GIS or BIM models. GIS models are used for modeling natural phenomena and man-made objects while newly constructed man-made objects such as buildings, bridges, etc, typically available in BIM models.

Our work supports 3D vector-raster conversion of vector models stored in CityGML² and it uses the voxelization principles defined in [15]. The voxelization of surfaces and curves are a customization of the Topological Voxelization approach presented in [11] and they ensure correct representation of geometries, topology, and semantics [15].

One object at the time is voxelized and the results saved into a Parquet file. Instead of voxelizing sequentially an entire city, the voxelization is done in parallel by tiling it. For each tile a Parquet file is created. For efficient data access, a Parquet file size is kept above 1GB to maximize the length of the stored column. Such optimization, and the fact objects distribution is not uniform, the tiling is not uniform.

B. Point cloud voxelization

The voxelization library [15] additionally provides an extension for voxelization of point clouds. The methods provide an easy management of connectivity levels in the resulting voxels, they are not dependent on any external library except for primitive types and constructs, therefore, easy to integrate them into a DBMS.

The *in-situ* data access combined with efficient spatial selections allows voxelization of point cloud on demand and near real-time. It allows us to extract a series of point-clouds of certain region to determine volume differences of nature

²<http://www.citygml.org>

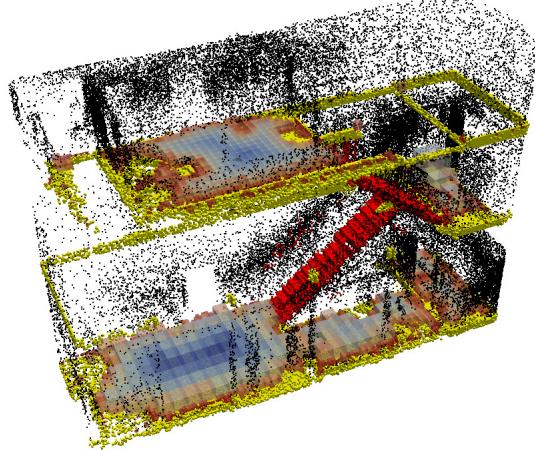


Fig. 5: Semantically enriched voxels from a point-cloud [7]

objects, or visualize quickly the impact of introducing or removing a man-made structure [15], [18].

For risk management such flexibility and efficiency allows rescue teams to study a building in a question of seconds. As an example, Figure 5 illustrates a large building of the Technical University of Delft (TUDelft). It maps the building into a series of points (red - stairs, yellow - floor and black-walls) while the voxels mapping empty spaces above the floor between objects using a color gradient, orange means objects are close by while blue means they are far away. The compact representation of the each voxel allows analysis of the possible routes and the available space to define escape trajectory routes.

C. Efficient spatial selections

For a performance profiling we have used the benchmark defined in [17]. The results are compared with the most efficient solution in [17], LAStools from Rapidlasso³.

1) **Setup:** The experiments are conducted in a server with double capacity as the one used in [17]: instead of 16 Intel Xeon CPUs, it has 32 CPUs; instead of 128GB of main memory, it has 256GB; and instead of 2 x 41TB SATA in RAID 5 configuration, it has SAS (Thunderbolt) with 24 x 2TB disks. Despite the difference, input and intermediate data fits in memory. MonetDB was set to only use 16 cores. Regarding the software, we used the JUN2016 branch of MonetDB (*M*) and the latest version of LAStools (*L*).

LAStools is assisted by a DBMS (*LD*) to store each file bounding box in order to avoid the inspection of each file header. It also required us to run *lassort* and *lasindex* to boost query performance. Such pre-query stage had the same cost, around 18 hours, as a complete data import for MonetDB.

³<http://rapidlasso.com>

2) **Data:** Massive 3D discrete measurements have been obtained through airborne LiDAR (Light Detection and Ranging) or terrestrial scanning campaigns. As an example, the height map of the Netherlands, the *Actueel Hoogtebestand Nederland 2* (AHN2)⁴ which is stored and distributed in more than 60,000 LAZ files, contains 640 billion points. The data is only composed by X, Y and Z coordinates, i.e., no extra benefit for column-oriented architectures compared to record-oriented architectures.

3) **Queries:** The queries are a series of small (*c*), large (*l*) space selections using simple (*S*) or complex (*C*) polygons. Table I's first line has which type of query and all other lines has the hot-run execution times for LAStools (*L*), LAStools combined with a DBMS (*LD*) and MonetDB (*M*). Due to space constraints, we have omitted all the queries for which it was not possible to obtain result in less than 1000 seconds or if it was 10 times worse.

4) **Results:** The 18 hours reported for pre-query data loading and preparation are automatically reduced thanks to our *in-situ* data access approach, i.e., only data relevant for the queries is imported using the file's metadata loaded during the attachment phase. For query performance, only for very small selections or simple polygons, LAStools combined with a DBMS, performs better than our solution. For all other queries with exception of query 24 and 27, our solution outperforms LAStools. It is important to notice that selections or aggregations on other attributes would be hard to express for the file-based solution and would required the inspection of all files' header. Furthermore, the numbers obtained by our open-source solution are comparable with the best commercial solution with customized hardware [17]. Overall our solution stands as the best DBMS solution.

D. Query execution

The used column-store, MonetDB, has operator-at-the-time paradigm with output of each operator being materialized before being passed as an argument to the next operator. The feature simplifies the integration of a nested column-oriented format, such as Parquet, since it allows us to un-nest a column and materialize it when needed.

One of the advantages of columnar processing is late materialization, i.e., tuples are re-constructed as late as possible in the query plan. It allows column-oriented architecture to have a low memory footprint during query execution, especially during the filtering phase. The extended operators, independently if they are processing nested or unnested data, continue to support late materialization.

⁴<http://www.ahn.nl>

TABLE I: EFFICIENT SPATIAL SELECTIONS OVER A MASSIVE POINT CLOUD DATA SET

Typ	sS	sS	sS	sS	sS	sC	sS	sC	sS	sS	sS	IC	IC	sS	sS	ss	IS	IS	IS	IS	IS	IC	IC		
Q	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	21	22	23	24	26	27	28	29
LD	.07	.16	.07	.16	.7	1.52	0.55	3.72	2.34	.08	.05	.26	412	102	.49	.04	.032	2.28	142.0	313	.234	282	.13	x	x
L	.9	.87	.78	.92	33.2	32.8	32.29	36.2	34.8	.88	.84	1.03	829	424	1.39	.75	1.18	20.74	x	828	x	x	923	x	x
M	.2	.41	.22	.5	.36	.66	.25	.69	.54	.24	.24	.44	24.9	23.6	.41	.08	.73	.15	99.9	35.9	363*	.17.2	.16	224	108

The scan- and aggregation operators were modified to be aware of the repetition level. The definition level is only used by projection operators to un-nest the data and do tuple reconstruction. The tuple re-construction happens in the presence of a blocking operator, or a result constructor, or when it needs to combine nested data with flat data.

V. FUTURE PLANS

Once fully operational, we will study thoroughly the robustness of the proposed conceptual model and the efficiency our storage model using in-house projects. At the same time, we will design support for horizontal scalability and for interactive visualization of voxel-based 3D city models.

A. Horizontal scalability

With voxels stored in Parquet, our current work aligns with on going advances for large scale spatial processing in the cloud. As future work we intend to explore the possibility of 3D data manipulation of large scale voxel-based 3D city models using GeoSpark [3]. GeoSpark was built to efficiently exploit the internals of Spark⁵. It extends Resilient Distributed Datasets (RDDs) to form Spatial RDDs (SRDDs). For efficient data parallelism it efficiently partitions SRDD data elements across machines and introduces novel parallelized spatial geometric operations.

GeoSpark is still in an early development stage and it only supports few geometries (point, rectangle, and polygon), two spatial indexes (R-Tree and Quad-Tree). On top of that, it also supports spatial queries, e.g., range queries, K nearest neighbor (KNN) queries, and join queries on large-scale spatial datasets. Its major advantage is the fact it is built on top of Spark. By using Spark infra-structure, Hadoop friendly file formats such as Parquet can be directly ingested and used for large spatial analysis. In addition to our single-server mode using MonetDB, we will also provide cluster-mode using Spark (both providing integrated R and Python environments).

B. Interactive visualization

For interactive visualization we plan to explore Cubiquity [2] as an extension of Unreal Engine 4⁶. Cubiquity is a voxel engine written in C++ and released under the terms of the MIT license. It allows the creation of volumetric (voxel-based) environments which can be dynamically modified, i.e., it enables dynamic digging, building, and destruction.

Cubiquity is a flexible and powerful voxel engine, e. g., create terrains with caves or defined environments built from millions of colored cubes. It supports both smooth terrain and colored cubes type environments, multiple volumes which can exist in transform hierarchies and direct voxel access for implementing procedural generation.

⁵<https://spark.apache.org/>

⁶<https://github.com/volumesoffun/cubiquity-for-unreal-engine>

VI. SUMMARY

In this work we have presented an architecture for a 3D column-oriented raster DBMS and so far our efforts on its implementation. The uniqueness of our solution stands on the combination of a novel concept model for 3D city models, a voxel-based one, with a efficient nested column-oriented format to explore the 3D city model at different levels of detail.

It is designed to iteratively load data from different sources and where topological and geometric functionality for 3D raster manipulation is part of the relational kernel and not an add-on. It is the first DBMS based solution with in-situ access to spatial data repositories and on demand voxelization. With it, spatial analysis tailored to different use case scenarios is done on demand and fast enough to be used by modern risk management systems where real-time interaction is a key feature.

REFERENCES

- [1] <https://parquet.apache.org/>.
- [2] <https://bitbucket.org/volumesoffun/cubiquity>.
- [3] <http://geospark.datasyslab.org/>.
- [4] D. J. Abadi, S. Madden, and M. Ferreira. Integrating compression and execution in column-oriented database systems. In *Proceedings of the ACM SIGMOD*, 2006.
- [5] D. J. Abadi, S. Madden, and N. Hachem. Column-stores vs. row-stores: how different are they really? In *Proceedings of the ACM SIGMOD*, 2008.
- [6] F. Alvanaki, R. Goncalves, M. Ivanova, and et al. GIS navigation boosted by column stores. *PVLDB*, 2015.
- [7] F. Fichtner. Semantic enrichment of as point cloud based on a octree for multi-storey path finding. *MSc Thesis, TU Delft*, 2016.
- [8] R. Goncalves, M. Ivanova, F. Alvanaki, J. Maassen, K. Kyzirakos, O. Martinez-Rubi, and H. Muhleisen. A round table for multidisciplinary research on geospatial and climate data. *IEEE e-Science*, 2015.
- [9] S. Idreos, F. Groffen, N. Nes, S. Manegold, and et al. Monetdb: Two decades of research in column-oriented database architectures. *IEEE Data Engineering Bulletin*, 2012.
- [10] M. Ivanova, Y. Kargin, and et al. Data Vaults: A Database Welcome to Scientific File Repositories. *SSDBM*, 2013.
- [11] S. Laine. A topological approach to voxelization. *Eurographics Symposium on Rendering*, 2013.
- [12] O. Martinez-Rubi and et al. Benchmarking and improving point cloud data management in monetdb. *SIGSPATIAL Special*, 2015.
- [13] O. Martinez-Rubi, M. L. Kersten, R. Goncalves, and M. Ivanova. A column-store meets the point cloud. *FOSS4G-Europe*, 2014.
- [14] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton, and T. Vassilakis. Dremel: Interactive analysis of web-scale datasets. *VLDL'10*, pages 330–339, 2010.
- [15] P. Nourian, R. Gonçalves, and et al. Voxelization algorithms for geospatial applications: Computational methods for voxelating spatial datasets of 3d city models containing 3d surface, curve and point data models. *MethodsX*, pages 69 – 86, 2016.
- [16] A. Stadler and T. H. Kolbe. SPATIO-SEMANTIC COHERENCE IN THE INTEGRATION OF 3D CITY MODELS. *ISPRS Archives – Volume XXXVI-2/C43*, 2007.
- [17] P. van Oosterom, O. Martinez-Rubi, and et al. Massive point cloud data management: design, implementation and execution of a point cloud benchmark. *Computer Graphics*, 2015.
- [18] S. Zlatanova, P. Nourian, R. Gonçalves, and A. V. Vo. Towards 3d raster gis: on developing a raster engine for spatial dbms. *ISPRS WG IV/2 Workshop: Global Geospatial Information and High Resolution Global Land Cover/Land Use Mapping*, 2016.

The eWaterCycle Project

Niels Drost*, Rolf Hut†, Maarten van Meersbergen*, Edwin Sutanudjaja‡, Marc Bierkens‡§ and Nick van de Giesen†

*Netherlands eScience Center

†Chair of Water Resources Engineering, Faculty of Civil Engineering and Geosciences

Delft University of Technology, Delft, The Netherlands

‡Department of Physical Geography

Utrecht University, Utrecht, The Netherlands

§Deltares, Utrecht, The Netherlands

Water related catastrophes such as floods are putting more and more people at risk. Moreover this has a large economic impact as well. For example, in 2011 a flood in Bangkok wiped out a large number of harddrive manufacturing plants, leading to a global shortage and increase in price for a two year period. We have a decent grasp on forecasting the weather, especially on the short to medium term (a few days to a week). We have no such grasp for flood forecasting, especially not on the global scale.

The eWaterCycle [1] project tries to make progress towards remedying this by creating a global Hydrological forecast system. We combine:

- A global high resolution Hydrological model.
- Data from a state of the art global weather forecast, GFS [2].
- Data assimilation of global satellite data, in this case EUMETSAT Soil Moisture observations [3].

In our system, we use open source software, and standard interfaces and file formats as much as we can, as this allows us to more easily swap components as better become available, and allows others to reuse parts of our work.

We use PCR-GLOBWB [4] as our global model. It is implemented in Python on top of PCRaster, a C++ library. Using Python allows rapid development and testing of new concepts. As the underlying implementation is C++, the efficiency of PCR-GLOBWB is not hindered by performance limitations of the Python interpreter.

We use OpenDA [5] for Data Assimilation. OpenDA allows scientists to use Data Assimilation schemes without having to re-implement these from scratch.

We use the Basic Model Interface (BMI) [6] to couple our model to OpenDA. BMI is a simple, easy to implement model interface. This allows us to easily replace the model used, and makes the model itself easier to re-use in other systems and frameworks.

We have also created a visualization based on Cesium and D3, and ncWMS as a backend server. This allows anyone to view any point of the data at full resolution, including the uncertainty of the forecast over time.

The final result: a fully open source global Hydrological forecasting system, and a starting point for further research in this direction. The live eWaterCycle forecast can be viewed at



<http://forecast.ewatercycle.org>, and all code can be found on GitHub at <http://github.com/eWaterCycle>.

REFERENCES

- [1] R. Hut, N. Drost, M. van Meersbergen, E. Sutanudjaja, M. Bierkens, and N. van de Giesen, “eWaterCycle: a hyper-resolution global hydrological model for river discharge forecasts made from open source pre-existing components.” *Geoscientific Model Development*, 2016, DOI: 10.5194/gmd-2016-225.
- [2] S. Moorthi, H.-L. Pan, and P. Caplan, *Changes to the 2001 NCEP operational MRF/AVN global analysis/forecast system*. US Department of Commerce, National Oceanic and Atmospheric Administration, National Weather Service, Office of Meteorology, Program and Plans Division, 2001. [Online]. Available: <http://www.emc.ncep.noaa.gov/gmb/moorthi/gam.html>
- [3] P. De Rosnay, M. Drusch, G. Balsamo, L. Isaksen, and C. Albergel, “Extended Kalman Filter soil moisture analysis in the IFS,” *ECMWF Spring Newsletter*, vol. 127, pp. 12–16, 2011.
- [4] L. P. H. van Beek, Y. Wada, and M. F. P. Bierkens, “Global monthly water stress: 1. Water balance and water availability.” *Water Resources Research*, vol. 47, no. 7, p. W07517, Jul. 2011. [Online]. Available: <http://onlinelibrary.wiley.com/doi/10.1029/2010WR009791/abstract>
- [5] N. v. Velzen, M. U. Altaf, and M. Verlaan, “OpenDA-NEMO framework for ocean data assimilation,” *Ocean Dynamics*, vol. 66, no. 5, pp. 691–702, Mar. 2016. [Online]. Available: <http://link.springer.com/article/10.1007/s10236-016-0945-z>
- [6] S. D. Peckham, E. W. H. Hutton, and B. Norris, “A component-based approach to integrated modeling in the geosciences: The design of CSDMS,” *Computers & Geosciences*, vol. 53, pp. 3–12, Apr. 2013. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0098300412001252>

The IQmulus high volume fusion and analysis platform for Geospatial Point Clouds, featuring a marine bathymetry use case

Quillon Harpham

Hydrodynamics and Metocean Group
HR Wallingford

Wallingford, United Kingdom
q.harpham@hrwallingford.com

Abstract—New data acquisition techniques are emerging and are providing a fast and efficient means for multidimensional spatial data collection. Single and multi-beam echo-sounders, airborne LIDAR, SAR satellites and mobile mapping systems are increasingly used for the digital reconstruction of the environment. All these systems provide point clouds, often enriched with other sensor data providing extremely high volumes of raw data. With these acquisition approaches, a great deal of data is collected, but it often requires harmonisation and integration before reaching its maximum use potential. Use cases include supporting numerical modelling on land such as simulations of flooding and drought, and for use in modelling waves and flow in seas and oceans.

The IQmulus high volume fusion and analysis platform offers an architecture for processing such geospatial point clouds, through a set of pre-defined workflows, on a cloud infrastructure. Workflow elements include deconfliction of spatially overlapping data, spline interpolation to create high precision surfaces and the latest visualisation techniques for these datasets.

Featured in the presentation is a workflow designed to process collections of surveys of water depth. Individual surveys vary both spatially and temporally and can overlap with many other similar surveys. Where measurements of water depth differ

greatly between surveys a strategy needs to be employed to determine how to create an optimal bathymetric surface using all of the relevant, available data. As part of its SeaZone suite of data products, HR Wallingford employs the latest deconfliction techniques to produce such a ‘best’ surface. The workflow begins with a methodology for prioritising individual surveys, followed by spline interpolation of adjacent or overlapping datasets with a potentially parallel implementation which includes tiling and stitching to create the final completed surface. An example of how these datasets can support the immersive visualisation of civil engineering applications is shown through HR Wallingford’s advanced Ship Simulation Centre.

Keywords—Environmental numerical modelling, bathymetry, deconfliction, point cloud, big data, immersive visualisation

ACKNOWLEDGMENT

The author acknowledges the IQmulus project team, for the contribution of each in creating these project exploitable outcomes and the funding provided by Framework Programme 7, Grant Number 318787.

Manufacturing of Weather Forecasting Simulations on High Performance Infrastructures

V. Šipková, L. Hluchý, M. Dobrucky

Institute of Informatics

Slovak Academy of Sciences

Dúbravská cesta 9

845 07 Bratislava, Slovakia

{sipkova, hluchy, dobrucky}.ui@savba.sk

J. Bartok

Microstep-MIS

Monitoring and Information Systems

Čavojského 1

841 04 Bratislava, Slovakia

jurob@microstep-mis.sk

B. M. Nguyen

School of Information and

Communication Technologies

Hanoi University of

Science and Technology

minhnb@soict.hust.edu.vn

Abstract—The Weather Research and Forecasting (WRF) model represents a mesoscale numerical weather prediction system designed for both the atmospheric research and operational forecasting use. WRF offers several computationally efficient platforms while providing serial and parallel models running with or without multi-threading. The main goal of the work described in this paper was to develop management software tools facilitating the execution of WRF simulations on High Performance Computing (HPC) infrastructures. The management tools enable easily to carry out the complex simulation process, representing a workflow of central jobs, i.e. preprocessing, modeling and post-processing, which are running primarily on our local HPC cluster with the possibility to exploit also the computational power of the European Grid infrastructure. The second objective of this work was the investigation of parallel MPI and hybrid parallel MPI+OpenMP WRF models using the created tools to run simulations. The aim was to find out the most suitable combination of hardware and software configuration setting for the realization of the WRF simulation process with the given input scenario. Performance results of real simulation processes running on various hardware configurations of the cluster are presented.

I. INTRODUCTION

In recent years, along with the rapid development of information technologies and extensive amounts of data observed also the field of mathematical modeling and computer simulation has enjoyed intense advances in both directions of data and computational intensive strategies. Computer simulations are of great importance mainly in such situations when it is impractical, financially expensive or too risky for the real-world system to be directly subjected to experimentation, or in case of evaluating a system before it is actually built [1], [2]. Applying simulations, the behavior of the event can be predicted based on a set of parameters and initial conditions, and solutions can be found out before time, money, and materials are invested. In most cases simulations with a real scenario represent a long-time, computational intensive and memory consuming job. Moreover, both the model calibration and also the research objective itself calls for performing a great number of simulation runs using diverse input scenarios with variable input parameters and boundary conditions. For these facts it is unavoidable for the researcher to have a HPC infrastructure in hand which is capable to accomplish

its task taking time as short as possible. The second no minor important issue is understanding of the middleware necessary to execute applications. For non-informatics scientists it is often too complicated to be used correctly. A promising approach has proven to employ various application supporting instruments that make easier the process of the submission and execution of simulations. Hiding much of the middleware complexity they can insulate the researcher from the computing platform and give him the impression that all of the used resources are available in a coherent virtual computer center.

The main goal of this work was to develop flexible and easily manageable software tools which enable the user the simple and efficient execution of WRF [3] simulations on the local HPC cluster [4] and on the European Grid Infrastructure [5]. We focus primarily on the investigation of parallel MPI and hybrid parallel MPI+OpenMP WRF models. Using the created tools we aimed to find out the most correspondent cluster configurations (i.e. the number of compute nodes, cores, MPI processes and OpenMP threads) for the realization of the WRF simulation with the given input scenario. Domain experts can then apply these configurations for their real, bigger data and/or for a longer simulation time repeatedly and continuously. The work presents performance results of real simulation processes running on various hardware configurations of the IISAS HPC cluster.

There are several related works in this research direction, which tried to help the WRF user community to realize simulations on high performance computing environments, such as scheduling WRF jobs in Grid [6] or Cloud [7], the execution management using Python scripting language [8]. These works show many advantages and disadvantages of the variety of hardware infrastructures, difficulties of the environment setup and flexible handling. In addition, at the moment, few Grid sites and Virtual Organizations are supporting the execution of parallel MPI and OpenMP applications. In Cloud computing, despite of its number of advantages, MPI computing is still not supported and the data transfer here is slow as well. Although WRF simulations using MPI and hybrid MPI+OpenMP models can

theoretically run effective on local HPC clusters, the efficient exploitation of the computational power is still not easy and straightforward for common users from the technological background viewpoint, nor for computing resources providers from the power consumption viewpoint. Therefore the search for (sub)optimum configurations for manufacturing large-scale simulations is unavoidable and beneficial for both sides.

The next content of the paper is organized in the following sections. Section II gives the overview of the WRF modeling system. Section III presents in brief the main capabilities of the middleware installed on target computing infrastructures. Section IV describes the realization of WRF simulations. Section V summarizes the WRF performance results for some model configurations. The conclusion of the work is listed in Section VI.

II. WEATHER RESEARCH AND FORECASTING

The domain of weather forecasting simulations described in this paper is a part of the IMS Model Suite, which is a product of the worldwide operating company MicroStep-MIS [9]. It is a complex software system designed to address the needs of the accurate forecast of the weather and hazardous weather phenomena, environmental pollution assessment and prediction of consequences of an accident. Weather simulations were realized with the big input scenario/data for sandstorm evolution prediction, transport and diffusion of sand and dust in the atmosphere, short term fog prediction or relative humidity.

A. WRF framework

In these weather forecasting simulations the Weather Research and Forecasting (WRF) model [3], [10] is employed which represents a numerical weather prediction and atmospheric simulation software designed for both research and operational purposes. WRF is a public domain software and is freely available for community use. It reflects flexible, state-of-the-art, portable code that is efficient in various computing environments ranging from laptops, compute clusters, to massively-parallel supercomputers. It is effective for a broad span of applications across scales ranging from meters to thousands of kilometers.

The WRF software framework provides the infrastructure that accommodates dynamics solvers, physics packages interfacing with the solvers, programs for initialization, variational data assimilation package, and air chemistry package. There are two dynamics solvers included

- ARW (Advanced Research WRF), originally referred to as the Eulerian mass or “em” solver developed at NCAR (National Center for Atmospheric Research),
- NMM (Nonhydrostatic Mesoscale Model) solver developed at NCEP (National Center for Environmental Prediction). In the WRF package there is included also the WPS (WRF Preprocessing System) system, WRF-DA (WRF Data Assimilation) system, and several post-processing and visualization tools.

B. WRF version 3 capabilities

The WRF model is a fully compressible and nonhydrostatic model (with a run-time hydrostatic option). Its vertical coordinate is a terrain-following hydrostatic pressure coordinate. The grid staggering is the Arakawa C-grid. The model applies the Runge-Kutta 2nd and 3rd order time integration schemes, and 2nd to 6th order advection schemes in both the horizontal and vertical. It uses a time-split small step for acoustic and gravity-wave modes.

The WRF version 3 supports a variety of capabilities such as real- and idealized-data simulations, various lateral boundary condition options, full physics and various filter options, positive-definite advection scheme, non-hydrostatic and hydrostatic (runtime option), one-way and two-way nesting, and moving nest, three-dimensional analysis nudging, observation nudging, regional and global applications, digital filter initialization.

The WRF code includes

- initialization program for real-data (`real`),
- initialization program for idealized data (`iideal`),
- numerical integration program (`wrf`),
- program to do one-way nesting (`ndown`) and
- program to do tropical storm bogussing (`tcc`).

The idealized simulations typically manufacture an initial condition file for the WRF model from an existing 1-D or 2-D sounding and assume a simplified analytic orography. The real-data cases usually require preprocessing through the WPS package, which provides each atmospheric and static field with fidelity appropriate to the chosen grid resolution for the model.

C. WRF key components

The key components of the WRF software package 3.7.1 installed on our cluster are the following:

1) *WPS (WRF Pre-processing System)*: WPS contains three programs whose collective role is to prepare the input to the program *real* for real-data simulations. Each of programs performs one stage of the preparation:

- a) *geogrid* defines model domains and interpolates static geographical data to the grids,
- b) *ungrib* extracts meteorological fields from GRIB-formatted files, and
- c) *metgrid* horizontally interpolates the meteorological fields extracted by *ungrib* to the model grids defined by *geogrid*. The work of vertical interpolating meteorological fields has been moved to the program *real* under WRF.

2) *ARW (Advanced Research WRF)*: is the dynamics solver which represents the core component of the WRF modeling framework. ARW may be run with user-defined initial conditions for an idealized simulation, or it may be run real-data cases using interpolated data from either an external analysis or forecast. ARW is based on the Eulerian solver for the fully compressible nonhydrostatic equations, cast in flux (conservative) form, using a mass (hydrostatic pressure) vertical coordinate. To list all ARW features, the scientific

and algorithmic approaches (the solver, physics options, initialization capabilities, boundary conditions, and grid-nesting techniques) lies beyond the scope of this paper; the detailed description is included in [11].

3) *Post-processing Utilities and Visualization Tools*: their function is to post-process and display the WRF simulation output. The standard output from the WPS and real WRF model is represented in the NetCDF (Network Common Data Form) [12] format. In our simulation process the NCEP UPP (Unified Post Processor) [13] is employed. This software package is capable to post-process output from a variety of numerical weather prediction models, including WRF-ARW, and WRF-NMM. In addition to the option to output fields on the model's native vertical levels, it interpolates output from the model's native grids to NWS (National Weather Service) standard levels (pressure, height, etc.) and standard output grids (AWIPS, Lambert Conformal, polar-stereographic, etc.) in GRIB format. UPP is written to process a single forecast hour, however, it can be run across multiple forecast hours in a single output file.

D. WRF models

The WRF model has been designed to run either serially or in parallel, with or without multi-threading. The portable and highly modular WRF code supports two-level domain decomposition: for distributed memory, shared memory, and combined distributed+shared modes of execution. At first each model domain may be decomposed into patches for distributed memory, and then, within each patch the multi-threading is applied over tiles for shared memory. Each domain in a simulation may be decomposed and tiled independently.

III. HPC ENVIRONMENTS FOR WRF SIMULATIONS

Most modern HPC clusters are characterized by the multi-level hierarchical architectures. They are composed of sophisticated nodes which may consist of multi-core processors, floating-point accelerators, graphics processing units, and complex memory hierarchies. Computing nodes are linked together by an high-power interconnection network enabling interactions with each other. Each node has

- its own address space (distributed memory), and
- all cores within a node have access to a global address space (shared memory).

For applications to be executed on compute clusters representing systems with distributed and shared memory, standard programming models are

- MPI (Message Passing Interface) [14],
- OpenMP (Open Multi-Processing Interface) [15] with the multi-threading concept and
- the combination of MPI+OpenMP.

For extremely data-intensive applications, where similar calculations are performed on vast quantities of data, the employment of GPUs (Graphics Processing Units) [16] with corresponding programming technologies has proved to be highly beneficial. The most mature programming model for GPUs are

- CUDA (Compute Unified Device Architecture) [17], [18]
- OpenCL (Open Computing Language) [19] which provides the open royalty-free standard for cross-platform parallel programming of modern devices.

A. HPC clusters

The execution of a job on a local cluster is realized by default through the Portable Batch System (PBS) [20]. It represents a workload management system that provides a unified batch queuing and job management interface to a set of computing resources. PBS has three primary roles:

- 1) *queuing* – jobs submitted to the resource management are queued up until the system is ready to run them,
- 2) *scheduling* – selecting which jobs to run, when and where, according to a predetermined policy,
- 3) *monitoring* – tracking system resources, enforcing usage policy, and monitoring, tracking and manipulating jobs.

PBS provides a set of commands that the user can use to submit, monitor, alter, and delete jobs. For submitting a job to PBS the command `qsub` is used to which the *batch job script* written in scripting language (Shell, Python, Perl, etc.) is passed as an argument. After job submitting, PBS returns a job identifier which is asked as input parameter in any actions involving the job, such as checking job status, modifying the job, tracking the job, or deleting the job. The initialization of a parallel MPI job within a PBS batch script is performed using the command `mpixexec`, which handles the creation of the node list file, provides for spawning copies of the executable on nodes using the task manager library of PBS, and redirects standard input/output/error streams to the Shell from which it was invoked. It also enables to specify the mapping strategy of MPI processes to physical processors.

B. Grid systems

Grid systems [21], [22], [23], [24] can involve thousands of computing resources, which may be geographically distributed. To achieve the best performance on a given computing infrastructure the programming model should be capable to exploit *concurrently* as many of underlying hardware components as possible. Grid computing requires from applications to apply hybrid programming models combining technologies for cluster computing and technologies of Web and Grid services.

The execution of applications on the European Grid Infrastructure [5] is realized exclusively through the Grid middleware EMI [25], [26], [27]. It is placed between the infrastructure and user applications and enables sharing the heterogeneous grid resources including commodity or HPC clusters, disk storage, various instruments, data archives or digital libraries, and software packages. EMI comprises of services and corresponding client commands involved in the processing and management of user requests concerning the execution of a computational task, storage management, data access and data transfer, and retrieval of information. Job management services are concerned with the acceptance, scheduling, monitoring, and management of remote computations. A grid job consists

of a computation, and optionally, the file transfer and management operations related to the computation. For distributing and managing jobs across computing and storage resources the Workload Management System (WMS) takes care. The access to the WMS is provided by the service WMProxy exposing a web services interface that the user can interact with by means of the Command Line Interface (CLI).

Jobs to be submitted to the Grid system, using the command `glite-wms-job-submit`, must be described in the Job Description Language (JDL) [28]. JDL is a flexible, high-level language which enables to describe one job and aggregates of jobs with arbitrary dependency relations, and to express any requirements and constraints on the computing element, storage element, installed software, and others. In general, JDL attributes hold request-specific information which designate in some way actions that have to be performed. Besides simple jobs WMS can handle also compound job structures:

- a) *Directed Acyclic Graph (DAG)* - a set of jobs where the input/output/execution of one of more jobs may depend on one or more other jobs.
- b) *Parametric job* - a job having one or more parametric attributes. The submission of a parametric job results in the submission of a set of jobs having the same descriptions apart from the values of the parametric attributes.
- c) *Collection* - is a set of independent jobs that for some reasons have to be submitted, monitored and controlled as a single request.
- d) *Parallel MPI and multi-threaded OpenMP* applications - they need to be initialized from within a sequential executable script using the software tool MPI-Start [29]. The MPI-Start provides an abstraction layer that offers a unique interface to the grid middleware to start MPI programs with various execution environment implementations.

IV. MANUFACTURING OF WEATHER PREDICTION SIMULATIONS

The whole WRF simulation process consists of the accomplishment of the sequence of many programs which are of different type and complexity – serial and parallel, taking a different number of processors for execution. The set of all programs can be divided into three groups that shape the jobs of the WRF workflow. Formally, the *WRF workflow* is defined as a simple DAG – the sequence of three consecutive jobs (*WPS preprocessing*, *WRF modeling*, *UPP post-processing*), where the input and execution of the second/third job is dependent on the output of the first/second job (see Fig. 1).

Each of the workflow jobs is performed through the invocation of its own *job-run* script (`run_wps.sh`, `run_wrf.sh`, `run_upp.sh`) implemented as a command line in the *Shell* language. Scripts embody all necessary operations, including parameters and various conditions checking, environment variables setting, the input / output data management, and calls to diverse transforming scripts and executables. A logging information about all operations is stored and is available after the job has terminated. All three scripts require

input parameters specifying the predicted time period (*startDate*, *endDate*, *predictedHours*), the number of MPI processes, and optionally, the number of OpenMP threads. The post-processing script includes additionally input parameters specifying the section of the predicted time period (*firstHour*, *lastHour*) to be processed. The first action within all scripts is the checking and validation of the given input parameters, and according to their values the MPI and OpenMP setup is defined (parameters and options for the MPI command `mpiexec`).

The WRF workflow jobs are responsible for the carrying out the following tasks:

- 1) **WPS preprocessing** (script `run_wps.sh`) – the job provides for the conversion of input files in GRIB-format to the files in netCDF-format with the names `met_em.d<domain>.<date>.nc`, where `<domain>` represents the domain ID, and `<date>` represents a date string with the format “`yyyy-mm-dd hh:mm:ss`”. Transformations are carried out by the successive invocation of three programs: `geogrid.exe` (serial/MPI), `ungrib.exe` (serial) and `metgrid.exe` (serial/MPI). Each of the programs reads parameters from the common namelist file `namelist.wps` and table files `GEOGRID.TBL`, `METGRID.TBL`, `VTable`.
- 2) **WRF modeling** (script `run_wrf.sh`) – the job provides for the execution of two parallel MPI/MPI+OpenMP programs: `real.exe` and `wrf.exe`. The initializing program `real.exe` inputs data files produced in the `run_wps.sh` and generates files `wrfinput_d<domain>` and `wrfbdy_d01` which are directly used by the integration program `wrf.exe`. A successful run should create several output files with the names `wrfout_d<domain>_<date>`. Most of options to run the simulation are handled through the variables defined in the file `namelist.input`.
- 3) **UPP post-processing** (script `run_upp.sh`) – the job provides for the conversion of files in netCDF-format produced by `wrf.exe` into the GRIB-format using the program `unipost.exe` (serial/MPI) which is called in a nested cycle over the given predicted hours and a domain list. The `unipost` works according to the variables defined in the control file `wrf_cntrl.parm` which is composed of a header and a body. The header specifies the output file information and the body allows the user to select which fields and levels to process. There is no dependency between processing data of the predicted time period hours, so, the computation cycle may be split and the job can be realized as a *parametric study*, where each sub-job handles a section of the time period.

All these scripts and executables, together with several configuration and table files, constitute a fixed suite of programs working within a common directory space.

In case of computing on the grid infrastructure EGI, the WRF workflow is designed as one grid job encapsulating

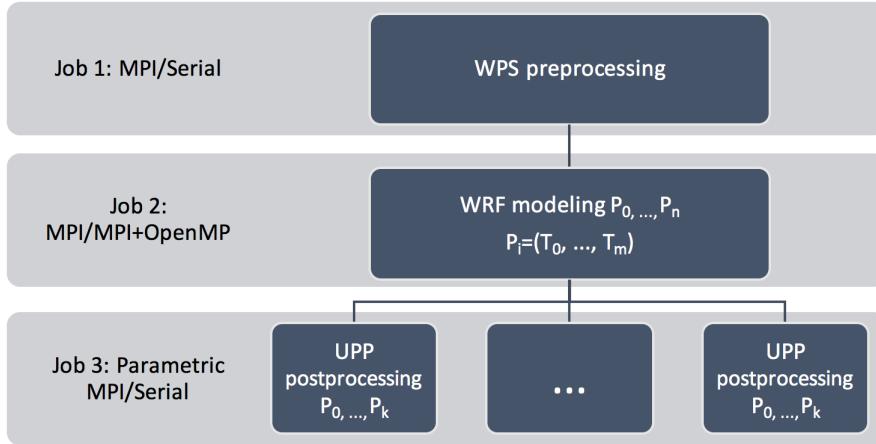


Fig. 1. Workflow of WRF simulation process (P_i - MPI process, T_j - OpenMP thread)

all tasks: WPS + WRF + UPP. Accordingly, it is performed through the invocation of one *job-run* script which provides for a consecutive execution of all program modules and also data file transfers between the Computing Element (CE) and Storage Element (SE). For the submission of each MPI program the software tool MPI-Start [29] is used.

In order to facilitate the WRF workflow execution and to automate and speed up the process of multiple workflow runs, we have developed a couple of supporting tools: *WRFworkflow-manager* scripts applicable for simulating on the local cluster [4], and on the grid infrastructure EGI [5]. Scripts are implemented as a command line in the *Shell* language. Every *WRFworkflow-manager* script performs the next actions:

- In the first step, it accepts and checks input parameters specifying the time period to be predicted (mandatory), and the number of sub-jobs the post-processing should be divided into (optional) – this determines the number of jobs in the parametric study and the size of the predicted time period sections assigned to. The values of parameters are passed to the subsequent commands.
- Following the values of input parameters, for each workflow job a *job-submission* script is produced.

In case of computing on the local cluster, it represents a file in Shell containing PBS directives, comments, and executable commands (including a call of the *job-run* script). The *job-submission* script serves as an input to the PBS submission command `qsub` [20].

In case of computing on the grid infrastructure EGI, one grid *job-submission* script is created which represents a file in JDL [28] language, and serves as an input to the submission command `glite-wms-job-submit` of the middleware EMI [26], [27].

- Finally, it provides for the realization of the WRF workflow employing the previously generated *job-submission* scripts to start all workflow jobs.

The WRF simulation process is started through the invocation of the *WRFworkflow-manager* script with needed input parameters. The environment which the WRF simulation process is running in, has the following structure:

geog	– directory for geographical data and several geo-tables (symbolic link)
cfg	– directory for configuration files specifying the input scenario and options for the simulation
parm	– directory for UPP post-processing parameters
bin	– directory for run-scripts and executables
input_arch	– directory for input data files
output_arch	– directory for output files
wps_run	– directory for WPS preprocessing
model_run	– directory for WRF modeling
postpr_run	– directory for UPP post-processing

The execution of WRF parallel MPI programs on the cluster requires the shared file system to be available. So, the WRF running environment together with all files included is hosted on the network file system (NFS) in the address space visible and accessible by each worker node of the cluster.

V. PERFORMANCE RESULTS

In order to find out the most suitable cluster configuration, that is “*the number of compute nodes, cores, MPI processes and OpenMP threads*” for a given input scenario, it was necessary to run many experiments. All experiments were realized using the developed *WRFworkflow-manager* scripts with only varying the input parameters. We concentrated especially on the WRF modeling phase, as this represents the dominant, most time-consuming part of the simulation process. Within the grid-aware simulation our interest was directed to the evaluation of the grid overhead and time delay brought

TABLE I
SERIAL MODEL: EXECUTION TIMES OF WRF WORKFLOW

	Number of		Time hh:mm:ss
	nodes	cores / node	
WPS	1	1	00:39:54
WRF	1	1	15:57:53
UPP (2 jobs)	1	1	00:03:48
Prediction time: 3 hours			

TABLE II
MODEL MPI: EXECUTION TIMES OF WPS PREPROCESSING AND UPP POST-PROCESSING OF WRF WORKFLOW ON IISAS HPC CLUSTER

	Number of			Time hh:mm:ss
	nodes	cores / node	MPI procs	
WPS	1	10	10	00:04:22
UPP (1 job)	1	3	3	00:03:19
UPP (2 jobs)	1	3	3	00:01:44
Prediction time: 3 hours				

about by data transfers between the User Interface machine (UI), CE and SE.

A. IISAS HPC Infrastructure

The IISAS production cluster [4], which is a part of SIVVP and Grid site, has the following hardware configuration: 52x IBM dx360 M3 (2x Intel E5645 @2.4GHz, 48 GB RAM, 2x 500 GB scratch disk), 2x IBM dx360 M3 (2x Intel E5645 @2.4GHz, 48 GB RAM, 2x 500 GB scratch disk, NVIDIA Tesla M2070: 6 GB RAM + 448 CUDA cores), 2x x3650 M3 managing servers (2x Intel E5645 @2.4GHz, 48 GB RAM, 6x 500 GB disks), 4x x3650 M3 data-managing servers (2x Intel E5645 @2.4GHz, 48 GB RAM, 2x 500 GB disks, 2x 8 Gbps FC), 1x x3550 M4 server (1x Intel E5-2640 @2.5GHz, 8 GB RAM, 2x 500 GB disks), InfiniBand 2x 40 Gbps (in 52+2+2+4 nodes), 2x DS3512 with 72TB disks. The cluster includes also 8x IBM dx360 M4 (2x Intel E5-2670 @2.6GHz, 64 GB RAM, 2x 500 GB scratch disk, 2x NVIDIA Tesla K20: 5 GB RAM + 2496 CUDA cores) with 10Gbps Ethernet interconnection. The HPC cluster is a part of the grid infrastructure SlovakGrid [30] binding all Slovak computing centres integrated in the European Grid Infrastructure EGI.

B. Manufacturing on IISAS HPC cluster

Simulation experiments were performed using the WRF package version 3.7.1 and UPP post-processor version 3.0. The code was built by GNU 4.4.7 compilers (gcc, gfortran, OpenMP) with the Open MPI version 1.10.0. It has been installed on the IISAS HPC cluster [4] located at the Institute of Informatics, Slovak Academy of Sciences, Bratislava.

Two parallel models of WRF have been investigated: the pure MPI [14] and the combined MPI+OpenMP [14], [15]. For WPS preprocessing, the WRF real-data preprocessor and UPP post-processing the fixed number of MPI processes was set: 10, 12 and 3, respectively, these proved to be the most appropriate (Table II). The prediction time period of

TABLE III
MODEL MPI: EXECUTION TIMES OF WRF MODELING

	Number of		Time hh:mm:ss
	nodes	MPI procs	
WRF	1	8	02:36:33
WRF	2	16	01:27:01
WRF	4	32	00:49:03
WRF	8	64	00:30:13
WRF	12	96	00:22:21
WRF	16	128	00:20:47
WRF	20	160	00:16:31
WRF	24	192	00:15:56
WRF	28	224	00:15:23
WRF	32	256	00:13:57
Number of cores per node: 8			
Prediction time: 3 hours			
Data transfer in Grid version (1)	\approx 00:30:00		
Data transfer in Grid version (2)	\approx 00:02:04		

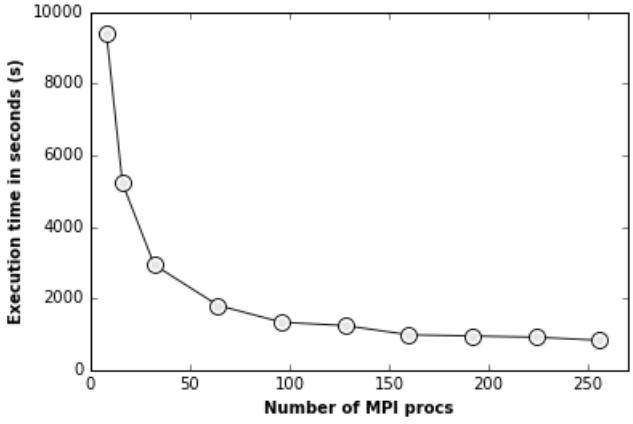


Fig. 2. Model MPI: Execution times of WRF modeling

3 hours was chosen only for purposes of testing the models performance, in real practice, simulations with the prediction time period of 48-72 hours are enforced.

As an input served the initial and boundary conditions (meteorological situation) from the global model GFS (Global Forecasting System of US National Weather Service). The uppermost domain used the same horizontal resolution as input 50 x 50 km, while the second domain provided smooth transition to the detailed 5 x 5 km resolution in the third domain. The setting enabled to model the whole Arabian Peninsula weather in resolution that allowed to take into account the mountain ranges and variability of the coastline. The final domain with the resolution 1.8 km was centered to the populated north-eastern coast of the peninsula with agglomerations around Dubai and Abu Dhabi with high demand on the quality of weather services.

Tables III and IV summarize the times taken by WRF workflow jobs for various cluster configurations (the number of

TABLE IV
MODEL MPI+OPENMP: EXECUTION TIMES OF WRF MODELING

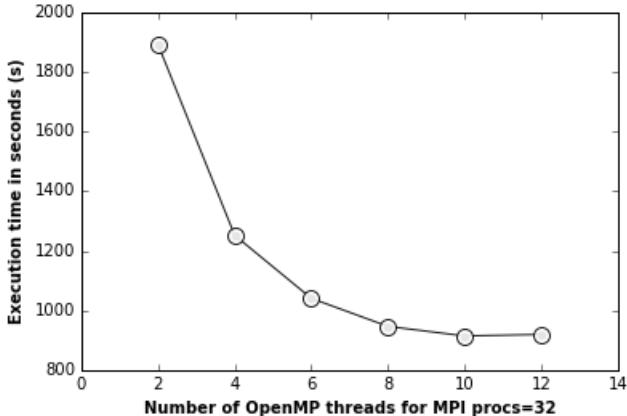


Fig. 3. Model MPI+OpenMP: Execution times of WRF modeling

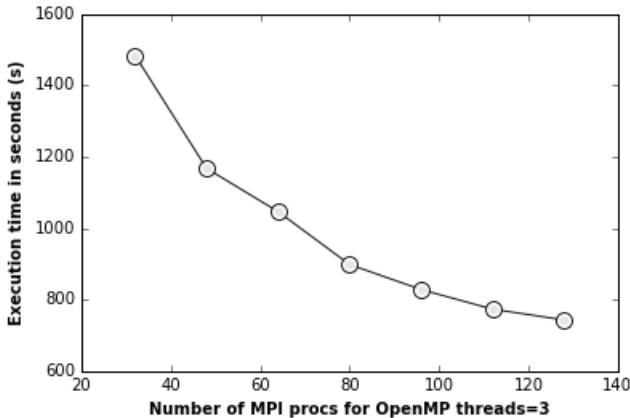


Fig. 4. Model MPI+OpenMP: Execution times of WRF modeling

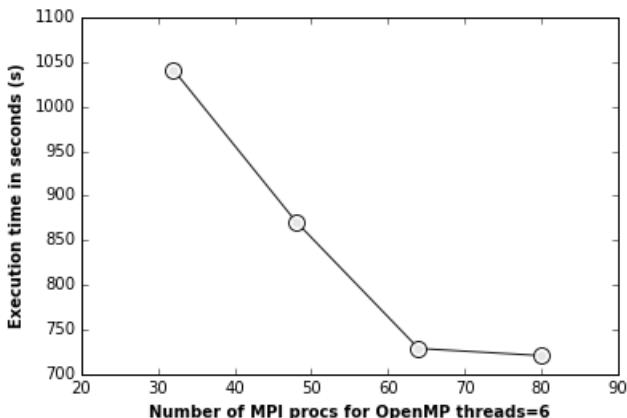


Fig. 5. Model MPI+OpenMP: Execution times of WRF modeling

	Number of			Time hh:mm:ss
	nodes	MPI procs	OpenMP threads	
WRF	8	32	2	00:31:31
WRF	16	32	4	00:20:52
WRF	16	32	6	00:17:21
WRF	32	32	8	00:15:47
WRF	32	32	10	00:15:15
WRF	32	32	12	00:15:20
WRF	8	32	3	00:24:44
WRF	12	48	3	00:19:28
WRF	16	64	3	00:17:27
WRF	20	80	3	00:14:59
WRF	24	96	3	00:13:49
WRF	28	112	3	00:12:54
WRF	32	128	3	00:12:24
WRF	16	32	6	00:17:21
WRF	24	48	6	00:14:31
WRF	32	64	6	00:12:09
WRF	40	80	6	00:12:01

nodes, cores per node, MPI processes and OpenMP threads). From the obtained results is apparent, that for the given input scenario the WRF code shows a good scalability. The hybrid variants can additionally speed up the simulation by an extra factor.

For the sake of comparison in Table I the times of the pure serial WRF workflow jobs are introduced.

C. Manufacturing on European Grid Infrastructure

On the grid infrastructure EGI we tested the simulation process using only the MPI model aimed at finding out the grid overhead. The WRF workflow represents a complex system consisting of many serial and parallel programs performed in successive steps. We have examined two possibilities.

In the first case, which is typical, the execution of a grid application is started on a CE in a scratch address space not hosted on a shared file system. However, each of the WRF MPI programs requires to be executed in a shared address space. Consequently, the transfer of input, intermediate, and output data files (of size of GB) between the scratch and shared address spaces constitutes a significant time increase. The WRF running environment in its initial state, all executives, and the set of input files are available as tar-balls stored in the SE from which they are downloaded. Then, after each simulation step, the intermediate results held in directories of the WRF running environment must be packed and transferred into the address space needed for the next processing. Geographical data are not participating on the transfer, they are placed on the fixed place and accessed by the symbolic link. Finally, the simulation results saved in output files are packed and uploaded to the SE.

The computing element for the simulation was our HPC cluster, so, the grid overhead itself was negligible. But the time

spent by all data transfers including packing and unpacking is substantial (see the last but one row in Table III), it is not acceptable in the real practice.

In the second case, the WRF simulation is started on a CE in a not shared scratch address space indeed, but this address space is immediately changed to the shared address space, and all components of the whole complex job, disregarding the type, are executed here. Finally, all standard input/output/error stream files (OutputSendbox files) are copied back to the initial scratch address space. This approach is equivalent to the execution of the WRF workflow on the local cluster. In order to detect the execution directory in the shared file system we manipulated and used some environment variables provided by the tool MPI-Start [29]. In this case the time of data transfers is reduced to times spent by data transfer between the SE and CE, it is introduced in the last row in Table III.

VI. CONCLUSION

In this work we presented results of performance testing of a real WRF simulation process configured with two parallel programming models: the pure MPI, and the hybrid MPI+OpenMP. Our experiments show that hybrid programming models seem a natural fit for the way most clusters are built today. In case of performing the WRF simulation on EGI infrastructure, the Grid overhead is caused mainly by the transfer of Big Data files between the jobs of the workflow and between the SE and CE.

ACKNOWLEDGMENT

This work is supported by projects VEGA 2/0167/16 and EGI-Engage EU H2020-654142. Simulations and technical realization were achieved on the hardware equipment obtained within the project SIVVP ERDF ITMS 26230120002. We would like to thank to all colleagues and partners for collaborations, scenario preparations and consultations.

REFERENCES

- [1] Y. Che, L. Zhang, C. Xu, Y. Wang, W. Liu, and Z. Wang, "Optimization of a Parallel CFD Code and Its Performance Evaluation on Tianhe-1A," *Computing and Informatics*, vol. 33, no. 6, pp. 1377–1399, 2014.
- [2] S. Benedict and M. Gerndt, "Scalability and Performance Analysis of OpenMP Codes Using the Periscope Toolkit," *Computing and Informatics*, vol. 33, no. 4, pp. 921–942, 2014.
- [3] "WRF - The Weather Research and Forecasting Model," [Online]. Available: <http://wrf-model.org/>, <http://www2.mmm.ucar.edu/wrf/users/>, <http://www.dtccenter.org/wrf-nmm/users/>.
- [4] "SIVVP - Slovak Infrastructure for High Performance Computing," [Online]. Available: <http://hpc.ui.savba.sk/>, <http://www.sivvp.sk/>.
- [5] EGI - European Grid Infrastructure. [Online]. Available: <http://www.egi.eu/>
- [6] F. Schuller, S. Ostermann, R. Prodan, and G. Mayr, "Experiences with distributed computing for meteorological applications: Grid computing and Cloud computing," *Grid computing*, vol. 8, pp. 1171–1199, 2015.
- [7] A. Quarati, E. Danovaro, A. Galizia, A. Clematis, D. D'Agostino, and A. Parodi, "Scheduling strategies for enabling meteorological simulation on hybrid clouds," *Journal of Computational and Applied Mathematics*, vol. 273, pp. 438–451, 2015.
- [8] A. Guerrero-Higueras, E. Garca-Ortega, J. Snchez, J. Lorenzana, and V. Matelln, "Schedule WRF model executions in parallel computing environments using Python," *In Third Symposium on Advances in Modeling and Analysis Using Python*, 2013.
- [9] MicroStep-MIS, Development and manufacturing of monitoring and information systems, processing of acquired data, research and numerical modeling. [Online]. Available: <http://www.microstep-mis.sk/>
- [10] W.C. Skamarock and coauthors, *Description of the Advanced Research WRF Version 3. NCAR Tech. Note NCAR/TN-475+STR, June 2008, 113 pp.* [Online]. Available: <http://dx.doi.org/10.5065/D68S4MVH>
- [11] "Users Guide for the ARW (Advanced Research WRF) Modeling System, Vers. 3.7," [Online]. Available: [http://www2.mmm.ucar.edu/wrf/users/docs/arw_v3.pdf](http://www2.mmm.ucar.edu/wrf/users/docs/user_guide_V3.7/ARWUsersGuideV3.pdf), http://www2.mmm.ucar.edu/wrf/users/docs/arw_v3.pdf.
- [12] NetCDF - Network Common Data Form. [Online]. Available: <http://www.unidata.ucar.edu/software/netcdf/>
- [13] Users Guide for the NCEP Unified Post Processor (UPP) Version 3. [Online]. Available: http://www.dtcenter.org/upp/users/docs/user_guide/V3/upp_users_guide.pdf
- [14] Open MPI - A High Performance Message Passing Library. [Online]. Available: <http://www.open-mpi.org/>
- [15] OpenMP - Open Multi-Processing, API Specification for Parallel Programming. [Online]. Available: <http://openmp.org/wp/>
- [16] J. Was, H. Mroz, and P. Topa, "Scalability and Performance Analysis of OpenMP Codes Using the Periscope Toolkit," *Computing and Informatics*, vol. 34, no. 6, pp. 1418–1434, 2015.
- [17] CUDA - Compute Unified Device Architecture. [Online]. Available: http://www.nvidia.com/object/cuda_home_new.html/
- [18] NVIDIA - The Visual Computing Company. [Online]. Available: <http://www.nvidia.com/content/global/global.php>
- [19] OpenCL - Open Computing Language. [Online]. Available: <http://www.khronos.org/opencl/>
- [20] "PBS - Portable Batch System," [Online]. Available: <http://www.mcs.anl.gov/research/projects/openpbs/>, <http://www.adaptivecomputing.com/>.
- [21] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publisher, 2005.
- [22] I. Foster, C. Kesselman and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *Int. Journal on Supercomputer Applications* 15(3), 2001.
- [23] I. Foster, C. Kesselman, J.M. Nick and S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration," June 22, 2002, Open Grid Service Infrastructure WG, Global Grid Forum.
- [24] WSRF - Web Services Resource Framework. [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf/
- [25] EMI - European Middleware Initiative. [Online]. Available: <http://www.eu-emi.eu/>
- [26] gLite 3.2 User Guide, EDMS Doc. 722398, July 18, 2012. [Online]. Available: <https://edms.cern.ch/file/722398/1.4/gLite-3-UserGuide.pdf>
- [27] EMI User's Guide, April 24, 2013. [Online]. Available: <https://edms.cern.ch/file/674643/1/WMPROXY-guide.pdf>
- [28] F. Pacini, *Job Description Language Attributes Specification*, September 2, 2011. [Online]. Available: <https://edms.cern.ch/document/590869/>
- [29] MPI-Start and MPI-Utils, version 1.5.3, December 11, 2013. [Online]. Available: <https://grid.ifca.es/wiki/Middleware/MpiStart/Releases/1.5.3/>
- [30] SlovakGrid - Slovak Grid Initiative. [Online]. Available: <http://www.slovakgrid.sk/>

The Climate Change for Flood and Debris Mitigation after Typhoon Morakot 2009 in Taiwan

Harold Yih-Chi Tan

Center for Weather Climate and Disaster Research (WCDR)

National Taiwan University

Taipei, Taiwan

yctan@ntu.edu.tw

Abstract—The typhoon Morakot struck Taiwan during August 8-10, 2009, and the government statistics analyzed that typhoon brought maximum 3,004mm in three days, and 21 rain stations is over 2,000mm. It claimed 673 casualties, 26 missing persons and 24,950 evacuees are evacuated by the Disaster Prevention and Response Act. The heavy and long period rainfall also triggered multi-hazards; such as: flooding, debris flow and deep landslides, and the economic loss is 6.2billion USD. This kind damage is extreme weather, and we think that climate change will bring huge impacts to nations all over the world. Those impacts including the followings: change in biosphere, long-duration drought, large floods trigger by extreme torrential rain, spatial change in homelands, and food scarcity.

The extreme weather induced by climate change is the most direct factor influencing the floods, e.g. the extreme rainfall increases discharge and inundation area, sea level and estuary water level raising induce overbank floods, and land-use abuse and land-slides trigger high concentration of sediment discharge and river bed aggradations.

This study aims at the settings of hydrological scenarios due to climate change, evaluation of hydraulic structures (e.g. levees), vulnerability and risk analysis, and adaption strategy and practices. The study area is focused on Kaoping River basin watershed. First, the hydrological scenarios due to climate change are set. Secondly, based on those scenarios, the hydraulic structures are evaluated. Thirdly, the vulnerability and risk analysis are performed. Last, adaption strategy and action plans are proposed by referencing to actions taken by the Netherlands, Japan and USA for improving the capacity of the hydraulic structures of this basin watershed

Keywords— *extreme weather, inundation, landslides, vulnerability and risk analysis*

Performance Evaluation of Environmental Applications using TELEMAC-MASCARET on Virtual Platforms

Minh Thanh Chung¹, Manh-Thin Nguyen¹, Nhu-Y Nguyen-Huynh¹, Nguyen Thong², Nam Thoai¹

¹ HPC Lab - Faculty of Computer Science and Engineering

² Faculty of Civil Engineering

Ho Chi Minh City University of Technology, VNU-HCM

Ho Chi Minh City, Vietnam

Email: ¹{minhchung}@cse.hcmut.edu.vn, ¹{nmthin,nhny,namthoai}@hcmut.edu.vn, ²{nguyenthong}@hcmut.edu.vn

Abstract—Environmental issues can be simulated in High Performance Computing (HPC) systems such as simulating flood inundation. This paper presents the simulation of two problems in Vietnam using TELEMAC-MASCARET, namely flood and salinization of Ho Chi Minh city and Mekong Delta. Evaluating these simulations on both physical machines and virtualization platforms are introduced in detail. To meet the requirements of serving multiple users or different simulation problems, virtualization environment offers feasible solutions and ensures the availability as well as the flexibility. Besides the popularity of virtual machine (VM) in Cloud, recently there is a lightweight virtualization platform called Docker. In this paper, we propose a model for deploying TELEMAC on Docker. Then, we show evaluations through different scenarios when deploying TELEMAC on Docker and VM. Docker has more potential with the lightweight architecture, especially its overhead is nearly negligible. The performance of Docker is better than VM, and the execution time of TELEMAC can be reduced significantly. The execution time of VM is about 1.6 times longer than Docker in terms of running the simulations of environment-related problems in Vietnam.

Keywords-Environmental computing, TELEMAC, Docker, performance

I. INTRODUCTION

Following the rising of environment-related problems, the methods of numerical simulations are playing an important role. The results of these simulations can make supports to estimate and prevent the damage caused by disasters. Besides the annual problem is flood inundation, Vietnam is suffering its worst drought in nearly a century with salinisation in the crucial southern Mekong delta. To tackle with these problems, the use of TELEMAC [1] is becoming popularly. However, the solutions using TELEMAC simulations need the power of parallel computing [2]. The TELEMAC system is one of suitable environmental simulation softwares that has developed rapidly. TELEMAC packages a whole processing chain for the calculation of water solute and sediment motions in the fluvial, coastal, estuarine, lacustrine and groundwater domains [3]. One of the most popular modules of TELEMAC using for environment-related issues in Vietnam is TELEMAC2D. This

module provides the hydrodynamics that consist of horizontal depth-averaged velocities and water depth.

In terms of simulating the large scale environmental issues, the execution time and accuracy are two major criteria. Taking advantages of the developing in HPC system, parallelization is the best way to improve the efficiency of simulations. However, to meet the demand of multiple purposes, the simulation of each problem can be performed many times along with different simulation modules. Simultaneously, the implementation of these modules on different platforms or different operating systems (OS) is possible. To extend the idea providing computing infrastructure on Cloud, we survey the performance as well as the efficiency of two different virtualization platforms, namely, hypervisor and containerization.

The hypervisor-based virtualization known as virtual machines (VMs) has been used for a long time with many service providers, e.g., Amazon EC2, VMWare. Differ from the hypervisor technique, containerization is a lightweight virtualization platform that creates virtual instances called containers [4]. Docker [5] is a containerization platform which is being widely used and developed. From our previous research [6], Docker has more advantages than VM in terms of the computational capacity, the network performance and the ability of data access associated with data intensive applications. The difference between Docker and VM is the architecture. Docker has the architecture sharing resources with the host kernel, and this is one of reasons that makes Docker containers reduce the overhead. Docker also obtains compatibility with some advanced technologies such as InfiniBand (IB) network [7], Xeon-Phi co-processor [8]. From those features of Docker, we evaluate the real efficiency of Docker when deploying the TELEMAC simulation modules in practice. With the real problems and applications, the performance of Docker is still questionable. Besides, we propose a model of deploying TELEMAC on Docker for scalability and give some experiences when running TELEMAC in parallel. The improvement of efficiency and accuracy of TELEMAC needs to consider the configuration of parallelization. The survey of speed-up shows that the maximum amount of cores is not the best way

for obtaining the shortest execution time.

The rest of the paper is organized as follows. Section 2 highlights the fundamental background about TELEMAC system and virtual platforms. In section 3 and 4, we present the motivation and previous related to our research. The experiences and model of deploying TELEMAC on Docker are shown in section 5. The next section reveals our performance results as well as evaluations about Docker and VM. Finally, we draw some conclusions and illustrate future work in section 7.

II. BACKGROUND

A. TELEMAC modeling and Environmental issues in Vietnam

TELEMAC model or TELEMAC-MASCARET system was developed by the Electricity of France (EDF) Company, with the participation of many research organizations around the world. The TELEMAC-2D module is used to simulate free surface flows in two dimensions of horizontal-space (average in vertical direction). TELEMAC uses the Saint–Venant equations to simulate the free surface flow. These equations are derived from Navier-Stokes equations based on assumptions on the flow pattern and approximations by the free surface modeling. The Saint-Venant equations are shown as below:

$$\frac{\partial h}{\partial t} + \vec{u} \cdot \Delta(h) + h \operatorname{div}(\vec{u}) = S_h, \quad (1)$$

$$\frac{\partial u}{\partial t} + \vec{u} \cdot \Delta(u) = -\vec{g} \frac{\partial z}{\partial x} + S_x + \frac{1}{h} \operatorname{div}(hv_t \Delta(u)), \quad (2)$$

$$\frac{\partial v}{\partial t} + \vec{u} \cdot \Delta(v) = -\vec{g} \frac{\partial z}{\partial y} + S_y + \frac{1}{h} \operatorname{div}(hv_t \Delta(v)), \quad (3)$$

where h is the depth of water; u and v are the velocity components; x and y are the horizontal coordinates; z is the free surface elevation, i.e., the elevation along with the vertical direction; \vec{g} is the gravity acceleration; v_t is the momentum diffusion coefficient; S_h , S_x and S_y are the sources or sink terms. There is a variety of simulation modules using standard algorithms to simulate or model environmental issues. Before running simulation modules, we need to set up a fine mesh to describe a simulated space. The fine mesh contains the numbers of elements and nodes in the form of unstructured grid.

For environmental issues, we concern two problems that are occurring in Ho Chi Minh city and Mekong Delta, namely flood inundation and salinization. Flood is one of natural disasters having the most impact in Vietnam. There are some kinds of floods. Some floods are induced by nature, but some floods are induced by the artificial structure, for example the dam-break. They can make a long term effect that people have to face a variety of damages within minutes or hours. Besides the flood inundation, that is the salinization in Mekong Delta. The prolonged salinization has damaged sugarcane plantation areas in several provinces of southern Vietnam.

Over 2,300 hectares of sugarcane plantation has sustained damage, accounting for over 30 percent of the total crop.

The utilization of numerical simulations can provide helps for experts to estimate and prevent these phenomena through the predictions. Based on the theories of physics, the flood flow or salinization of the sea water can be treated as a kind of fluid flows with free surface. We can simulate the water deep or the move of flood by the variant of the height of the free surface. TELEMAC is an open source software that is package and based on finite element method (FEM) to implement these large scale simulations as mentioned above. To deal with the large scale of the flow simulations, we need to the power of parallel computing represented by HPC system. In further, HPC leverages the large scale parallel computing for multiple tasks running different simulation modules.

B. Virtual platforms in HPC

For HPC applications on large scale clusters, the scalability as well as management are becoming increasingly important. For example, the prediction of environmental issues needs the large scale parallel computing in simulation [9]. Besides, the power of HPC system is also employed in other simulation domains, e.g., Land Transformation Model [10], Social Simulations [11]. That is the motivation for leveraging the idea of using computer to emulate itself, namely virtualization. There are many virtualization technologies recently. One of them is virtual machine based on hypervisor-based virtualization. Moreover, we have a new platform based on the lightweight virtualization technology called Docker [5]. The key differences between these two platforms are the architecture and the operation mechanism.

1) Virtual machine: VMs have been popular with the architecture relied on hypervisor-based architecture. The original idea of VM is the virtualizing of an entire system. It means that each VM is virtualized from hardware to OS software. Therefore, hypervisor layer plays a vital role in managing VMs [12]. It allocates and controls the resources of physical machine (PM) for guest domains. Furthermore, this layer not only emulates devices to guest domains, but also simplifies the physical architecture to virtualize. There are two types of hypervisor [13], including:

- **Type-1, native or bare-metal hypervisors:** These hypervisors run directly on the hardware of host to manage resources for guests. Some hypervisors are known as the Citrix XenServer, Microsoft Hyper-V, and VMware ESX/ESXi.
- **Type-2, or hosted hypervisors:** Differ from the previous type, these hypervisors run on the operating system from host. VMs deployed by these hypervisors are considered as a running process on the host machine, for examples, VMware Workstation, VMware Player, Virtual Box.

Besides, there are some hypervisors build on the hybrid architecture of type-1 and type-2, e.g., Kernel-based Virtual Machine (KVM) [14], FreeBSDs bhyve. VMs can greatly benefit cluster computing in such systems [15], especially from the following aspects:

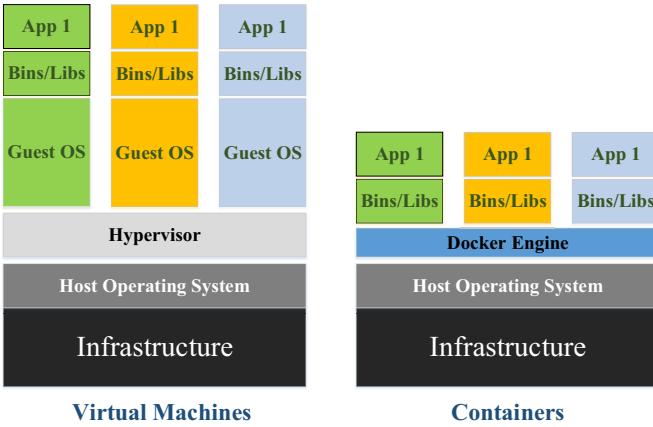


Fig. 1. The architecture of VM and Docker.

- **Availability and Scalability:** Virtual infrastructure can provide a large amount of resources from physical system and make them accessible. Concurrently, a service provider can expand the volume of virtual resources or shrink them.
- **Complete isolation and security:** Each VM is considered as a whole system consisting of hardware and software. Thus, their operation space can be ensure the isolation among VMs and PM. The state of guest can be recovered and controlled by Virtual Machine Monitor (VMM) when malicious codes crash the virtual system.
- **Ease of management:** An administrator may create or delete, start or shutdown a VM easier than a real machine. We can observe VMs along with additional information running inside each instance.
- **Customization:** The customization feature shows that we can modify both the specification of hardware and software before generating a guest, for example, CPU, RAM.

Overall, the deployment of HPC application on VM environment is similar to PM. A parallel application need three requirements in a distributed system: parallelization of that application, communication of compute nodes and parallel libraries such as Message Passing Interface (MPI).

2) **Docker:** In contrast, Docker platform leverages a lightweight architecture named containerization. In principle, Docker emulates their instances called containers at the abstraction level of OS. It is noticeable that Docker shares the same kernel with the host machine. Two containers do not need to concern the limitation of resources before creating them, they can scale up or down the use of all resources on PM. If containers are not running anything, they do not generate extra overhead. They only perform applications in spaces that are isolated from other containers and the host OS [16]. The operation model of Docker is client-server. At the client side, Docker has a set of commands to request the server. Inside Docker, there are three main components [5] Docker images, Docker registries and Docker containers. Concretely, Docker

image is a read-only template that plays a role in creating Docker containers. Then, Docker registry is a place which stores Docker images. There are the public registry and the private registry known as the local registry in the host. The public Docker registry has a huge collection of images for using, it is provided with Docker Hub. The rest of Docker is Docker container that is equivalent to a directory holding everything. A container can wrap things needed for running application. Docker architecture facilitates to run, start, stop, move, migrate virtual instances faster than VM.

This paper does not go into detail about the implementation of Docker. Docker makes use of several kernel features to deploy containers. Firstly, related to the virtualization at the OS level, Docker implements the isolation by using *namespaces* technology. Each container has a set of *namespaces* to define its own workspace such as PID, NET, IPC, MNT, UTS. One more feature is sharing resources with the host kernel. Docker needs a Linux kernel component called *ControlGroups* or *cgroups* to allocate and manage resources for each container. This can reduce the overhead when scale up the amount of generated containers using resources. As Figure 1 shown, if two Docker containers use the same libraries to run their applications, they can share these libraries of related dependencies. When comparing the architecture of Docker to VM, the hypervisor layer of VM provides an entire environment to each instance. The hypervisor creates virtual devices, privileged instructions, virtual memory address to isolate the operation space of VM with the host and others. Therefore, every malicious code crashing VM can recover because the resource integrity of PM is controlled by hypervisor. This is one of strong points about the security feature of VM in comparison with Docker. However, for the lightweight features, Docker is better than VM through reducing the overhead significantly and optimizing the use of resources.

III. MOTIVATION

Environmental issues are harmful effects of human activity that need an integration of solutions in all of related fields. Especially, HPC plays an important role in simulating and predicting the phenomena of environment. Associated with the simulation, the use of TELEMAC in recent years is being popular in Vietnam. For example, Vietnam's Mekong Delta is facing the most severe drought and salinization. Salinization is a worldwide problem affecting to not only land and water, but also the human life [17]. More importantly, the solution of using sequential execution is not efficiency with the large scale problems. That is the reason why the parallel computing is becoming extremely important.

However, along with the expand of demand of using parallel computing resources, physical system cannot support multiple users requiring different operating systems as well as different purposes in simulation. Virtual platforms are feasible solutions. Besides VMs that have been used for a long time, there is a new platform supporting benefits in Cloud environment with the low overhead, namely Docker. In previous research [18], we survey the performance between Docker and VM.

Our evaluation reveals that Docker has more potential than VM in some aspects, e.g., the computational capacity, the ability of data access, high throughput with the compatibility on InfiniBand infrastructure. That is a question whether Docker obtains the efficiency when running HPC applications such as TELEMAC in parallel. We propose a model of deploying TELEMAC tool on Docker environment to meet the demand of scalability.

Our evaluation shows early experiences of using Docker container to support computing infrastructure into environment-related simulations. For example, the use of TELEMAC is to simulate the status of salinization in Vietnam.

IV. RELATED WORKS

The integration of parallelization and simulation can achieve the high efficiency in various domains like civil engineering and transportation engineering. TELEMAC is a powerful open source used to simulate free-surface flows in two dimensions or three dimensions. HPC system is contributing the computing ability to the development of TELEMAC [19]. There is a variety of researches using TELEMAC to tackle the practical problems related to environment such as [20], [21]. The simulation results of TELEMAC facilitate to prevent the natural disasters. To improve the performance of integrating TELEMAC and parallel computing, several projects are initiated. For example, the hybrid programming model using MPI and OpenMP in TELEMAC to improve the parallel performance [9]. We approach the expand of idea providing the computing platforms on Cloud. Two virtualization technologies considered in this paper are VM and Docker container. From the evaluation [6] on our system with the integration of InfiniBand network, the performance of Docker and VM is evaluated through different aspects. Recently, there are some researches that compare the backgrounds of Docker architecture to VM [22] or evaluate criteria related to the operation of Docker and VM such as I/O access, TCP network, computing ability [23]. The previous results of our evaluations show that Docker has more potential in HPC field, especially when running different benchmark scenarios in parallel. In this paper, we measure and analyze the performance of Docker containers based on the running of TELEMAC modules. The expand of idea deploying simulations on Cloud platforms shows many benefits [24]. We have early experiences for running TELEMAC on Docker environment. The performance figures of TELEMAC on Docker highlight that Docker has many advantages in providing a lightweight computing infrastructure for Cloud.

V. MODEL FOR DEPLOYING TELEMAC ON DOCKER CONTAINER

Docker containers like wrappers that can package applications and everything required for execution. For instances, inside a container, we can wrap a complete filesystem containing code, runtime, system tool and libraries. Docker uses a technology called Union file system (UnionFS) to provide

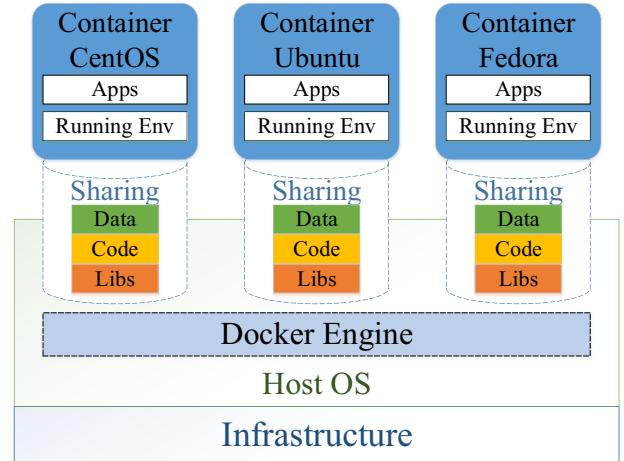


Fig. 2. The model of deploying TELEMAC on Docker.

the building blocks for containers. UnionFS is a filesystem service that allows files and directories of separate systems to be transparently overlaid, forming a single coherent file system [25]. In Docker, we can create various images, and they are read-only templates. Docker images include a series of layers. Thereby, UnionFS is used for merging these layers into a single image. In more detail, instead of replacing a whole image or entirely rebuilding with VM, we can update an application inside Docker container with a new layer gets build. Each Docker image is deployed on a base image, and multiple containers can use the same images without duplicating them. This is one of the reasons that makes Docker lightweight.

As Figure 2 shown, we propose a model for deploying HPC applications on Docker environment. The model use Docker container to merely create a running environment. Data are managed and configured under the host. For managing data in the container, Docker defines a definition named data volume to hold data. A data volume is a specially-designated directory within one or more containers that bypasses the UnionFS [5]. Data volume facilitate to persist data and make them independent of the container's life cycle. We employ these features to perform HPC applications. Data and related dependencies are located in the host machine. We mount a host directory into containers as a data volume. This benefits that data can be made available on any containers supporting the same applications. For HPC system, we consider Docker containers as the passive environment to submit jobs. All of configurations for execution are configured under the host. For example, we set up a computing infrastructure having InfiniBand network and Xeon-Phi co-processor that integrates with Docker container. From the model, we can guarantee data independent of running environment, and achieve the scalability of providing multiple users.

TELEMAC-MASCARET is a set of solvers for use in the field of simulating free-surface flows [1]. TELEMAC is written by Fortran language consisting of mathematics,

TABLE I
OVERVIEW OF PERFORMANCE BETWEEN DOCKER AND VM.

Criteria	Benchmark	Performance Rate (%)	
		Docker	VM
Latency of MPI send-receive on IB network	OSU-Benchmarks	99.75	33.66
Latency of MPI collective on IB network	OSU-Benchmarks	92.12	44.32
Computational capacity	High Performance Linpack	91.88	81.5
Ability of data access	Graph500	83.72	57.81

the physics, the advanced parallelization. To run TELEMAC on parallel system, there are three requirements including, parallel environment, source code of TELEMAC and pre-compiled library for parallel execution named METIS. The parallel environment has to provide the communication among compute nodes and the interface that allows two separate nodes to exchange data, e.g., Message Passing Interface (MPI). According to the model, we can configure and compile the source code of TELEMAC and related libraries under the host. Each container is merely the running environment containing the binaries of TELEMAC which are compiled under the host machine. This helps Docker diminish cumbersome. When Docker containers are used to run simulations, the input as well as data and TELEMAC source code can share among other containers and the host.

VI. EXPERIMENTAL RESULTS AND ANALYSIS

We carry out our evaluation on the performance of Docker when running TELEMAC to simulate environmental issues such as salinization, flood inundation. We study the efficiency of Docker containers when performing real applications. In the previous research [18], we evaluate the performance of Docker and VM on criteria, namely network throughput, computational capacity and the ability of data access. Docker obtains more advantages than VM. For example, Table I shows our statistic about the efficiency of Docker and VM compared to PM. The values from Table I are the performance rate in comparison with PM. They are the ratio of Docker/VM performance results to PM results.

Following that, we use a real application to measure the efficiency between Docker and VM before providing a scalable computing infrastructure in Cloud. Firstly, we propose some scenarios with many sample simulation modules of TELEMAC from small scales to large scales. Secondly, we deploy real problems related to environmental issues that need to be simulated. For instance, the recent problem is the salinization in Vietnam's Mekong-Delta.

A. Testing environment

Our evaluation environment considers the native performance as a standard to compare two virtualization platforms, namely Docker and VM. Objectively, we set up all VMs and containers on the given system called SuperNode.

The benchmarks are deployed on two compute nodes, equipped Intel Xeon CPU E5-2670 @ 2.6 GHz. There are 16 physical cores totally supporting Hyper Threading and 64 GB of RAM. The interconnection is 30 Gbps InfiniBand Connect-X3. The installed OS is CentOS 7 64-bit 3.10.0 on

both of physical and virtual environment. In more detail, VMs and Docker containers are deployed by the latest version including QEMU 2.4.0.1/KVM and Docker 1.8.1 respectively. The problem sizes of our testbeds are equivalent to the capacity of the system under test. We focus on the execution time of Docker and VM when running TELEMAC for different modules.

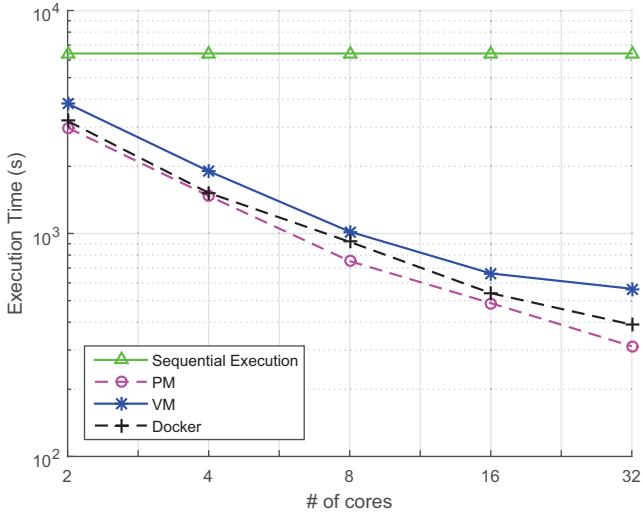
B. Benchmarks

Parallelization and Speed-up: In this scenario, we argue the meaning of parallelization and speed-up among Docker container, VM and PM in using TELEMAC for simulations. We deploy two VMs and Docker containers on each compute node. TELEMAC runs a problem that simulates a general flow with 100,000 elements in parallel mode. The problem is executed respectively on PM, VM and Docker. Time is a standard unit to evaluate the results of three environments.

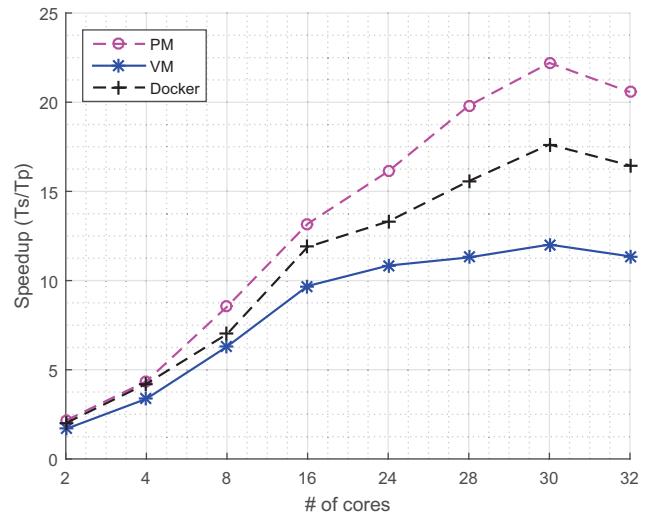
Figure 3.a shows the execution time of simulation on three different platforms. The sequential execution is longer than parallel execution over 6 times as the best case of 32-core. Concretely, Docker obtains the performance that is slightly adjacent to the PM, while VM has a higher level of deviation. It is noticeable that the more cores the execution of HPC applications has, not the more efficiency the system achieves. Figure 3.b reveals that the case of using 30 cores gets the best performance. TELEMAC needs to consider the speedup of running simulations. The purpose of parallelization is to divide a single problem into multiple parts for solving. However, the more parts we divide, the more overhead we have to face. When we divide into too many parts for parallel computing, the number of messages exchanged will increase. That is one of reasons causing the reduction of performance. For the gain of deploying TELEMAC, Docker gets the rate of sequential time over parallel time being better than VM.

Scalability and Availability: We set up two scenarios to benchmark: the first one is the evaluation of PM, VM, Docker along with TELEMAC simulating different problem sizes from the small scale to the large scale. The second one is also the evaluation deployed on Docker and VM, but we run TELEMAC with the different amount of virtual instances. At the same time, each instance of VM or Docker performs an individual simulation. From these scenarios, we concern the changes of performance and overhead as well when running a huge range of different problem sizes, or deploying the execution environment on the scalable amount of virtual instances.

The main requirement of TELEMAC is the running time, and the shortest interval is the best. In TELEMAC system, one of the crucial factors affecting to the simulation is the amount of elements in the fine mesh. In detail, the simulated object is the flow of South China Sea. The fine mesh is built through refining the numbers of elements and nodes to simulate the features of flow. From Figure 4, although the problem sizes increase, the performance of Docker still keep the shorter execution time when comparing to VM. At the view of operation mechanism, the I/O communication of system is one



(a) Parallelism of simulation on PM, VM and Docker.



(b) Speed-up of running TELEMAC on PM, VM and Docker.

Fig. 3. Scenarios 1: TELEMAC running on distributed system.

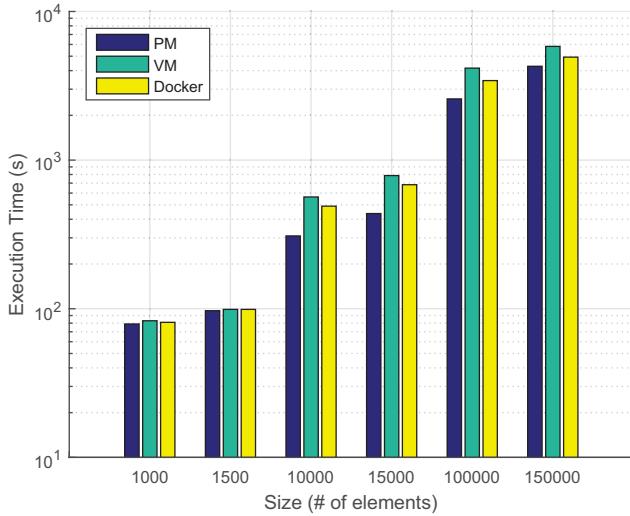


Fig. 4. Performance results of PM, VM and Docker simulating different problem sizes.

of important problems. For VM, the operation of each VM is controlled by hypervisor layer. Therefore, the I/O virtualization of VM can make a large overhead with the increasing number of problem sizes. To perform the I/O functions, VM switches the context between VM-hypervisor and hypervisor-PM. When the simulation module of TELEMAC varies the problem sizes, the overhead of I/O control makes the performance drop. For Docker, a benefit of its architecture is sharing resources with the host kernel. That makes the I/O access faster. This feature is one of the factors which creates the lightweight containers of Docker platform. Compared the results of Docker to PM, Figure 4 shows that the overhead of Docker can be acceptable with the demand of scalability.

Another scenario is the utilization of multiple users with

TABLE II
THE SAMPLE EXAMPLES OF TELEMAC RUNNING ON DOCKER AND VM.

Simulation modules			
TELEMAC2D	TELEMAC3D	SYSPHE	TOMAWAC
tide (Simulating the tide in 2D)	tide (Simulating the tide in 3D)	bump2d (Modeling the bump)	bottom-friction (Modeling the bottom friction)
m2wave	tidal-flats	bosse-t2d	dean
bowl	amr	foulessness	fetch-limited
breach	bump	glissement	opposing-current
bridge	delwaq	bar-formation	shoal
bumpcri	depot	bifurcation	turning-wind
	erosion-flume		whirl-current
	estu-gir-3D		

different simulation modules on our system. In this case, we set up a series of simulation modules based on the sample examples of TELEMAC system. Table II shows the sample examples of simulation modules. The scenario generates the increasing number of VM/Docker instances, and each instance is considered as a user running TELEMAC. An instance performs a specific simulation module different from each other. For the hypervisor-based virtualization, the cost of managing VMs can rise sharply up with the increasing amount. Because, the hypervisor still causes the extra overhead for each workspace of VM without running jobs. In this scenario, the sharing architecture of Docker has more benefits. To provide the computing environment for a huge range of users, not all of instances occupy the resources continuously in a long time. This mechanism of Docker can get the efficiency in these situations because the allocation of resources is implemented at the same host kernel.

Figure 6 shows the means of execution time between Docker and VM with the different amounts of instances. For example, the value being 2 represents two VMs/Docker containers executing TELEMAC. For a variety of users accessing the system, the completion time of each instance is different. Some of VMs/containers can finish their task before others. Therefore, for the architecture of VM, a guest virtualized an entire

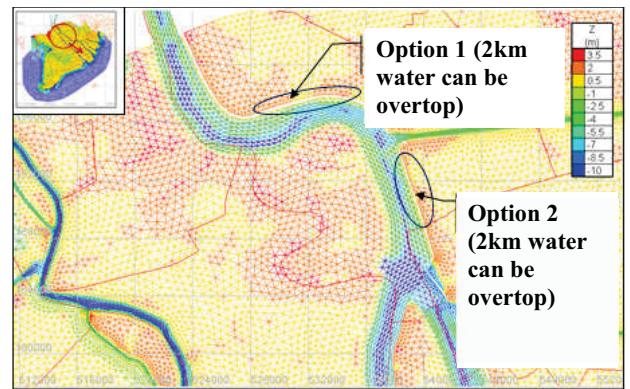
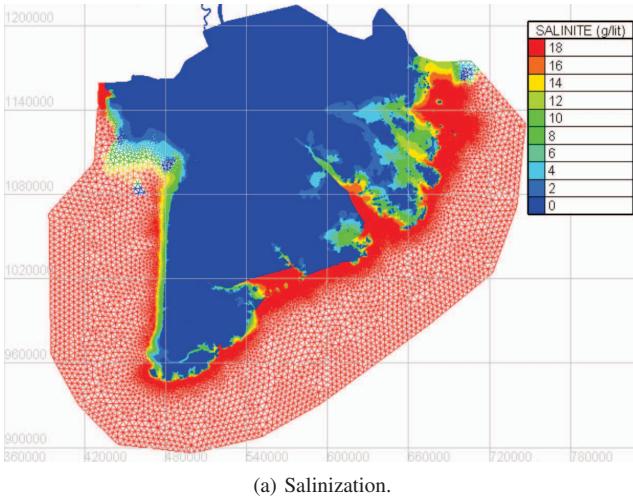


Fig. 5. The results of simulating Salinization and Flood Inundation.

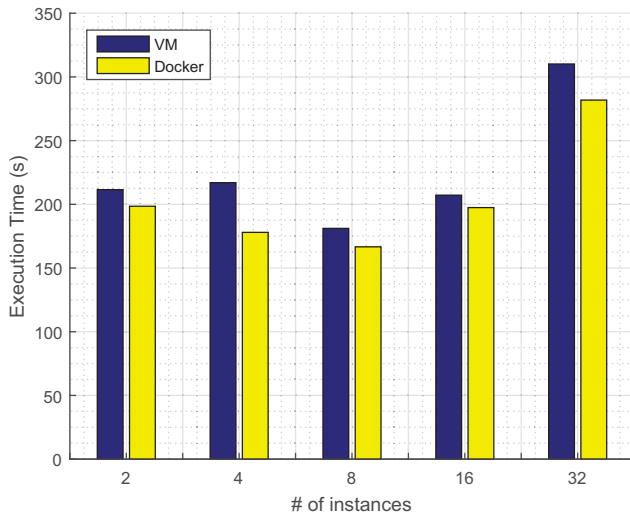


Fig. 6. Execution time of VM and Docker when increasing the number of instances.

system from hardware to software can make the extra load for running. By contrast, Docker containers without running jobs would not generate the extra load. As the results of Figure 6 shown, Docker obtains the better performance figures in comparison to VM. Docker still keeps the efficiency with the increasing number of containers, and their overhead is nearly negligible. In each case of this scenario, one VM/Docker container solves one individual simulation modules shown at Table II. The benchmark scenario represents multiple users using TELEMAC simultaneously.

Simulating the environmental issues in practice: In this scenario, we evaluate the real values of performance and efficiency between Docker and VM. The existing problems are the environmental issues occurring in Vietnam. We use data and input of two issues to compare Docker to VM. The two issues include salinization and inundation in Southern of Vietnam, particularly in Ho Chi Minh city and Mekong Delta.

The first problem is salinization in Mekong Delta. This is the serious condition that has occurred only once in the last 90 years. About 971,200 hectares of farming area in eight provinces of the Mekong Delta has been affected by salt water, according to the Ministry of Agriculture and Rural Development. Especially, Ben Tre is one of the provinces that has severely affected by salinization. The suggested solution is using the numerical model of TELEMAC2D to simulate flows in the definite duration. After that, there are some methods in the field of civil engineering to tackle the problem. In terms of HPC system executing TELEMAC, a fine mesh is built up through refining the numbers of elements and nodes. In this case, the fine mesh contains 722,000 unstructured triangular elements, the largest and smallest edge of the mesh are 75 kilometers and 4 kilometers.

Similarly, the second problem is flood inundation. The state of floods, a vast plain, mainly in the northern part of the Mekong Delta, is affected by annual flooding overflows from Cambodia across the Vietnamese border. The flooded area ranges from 1.2 to 1.4 million hectares with the low and medium flooding, and around 1.9 million hectares with the high flooding annually. The fine mesh of this issue contains 716,000 triangular elements. The size of simulated area is 81,000 square kilometers. This problem is also used the numerical model of TELEMAC2D to simulate the state of floods.

Figure 7 shows the execution time of two issues mentioned above on PM, VM and Docker. The figures of PM are the standard to evaluate the efficiency of Docker and VM. From the Figure 7, it is clear that Docker takes the shorter time to run the simulations in comparison to VM. Docker has many advantages in terms of I/O access, lightweight architecture and resource allocation. The time of running real simulations on VMs takes 1.6 times longer than Docker containers. Figure 5 shows the results of simulating salinization and flood inundation in Vietnam by using TELEMAC2D. From these results, solutions can be proposed to deal with these problems.

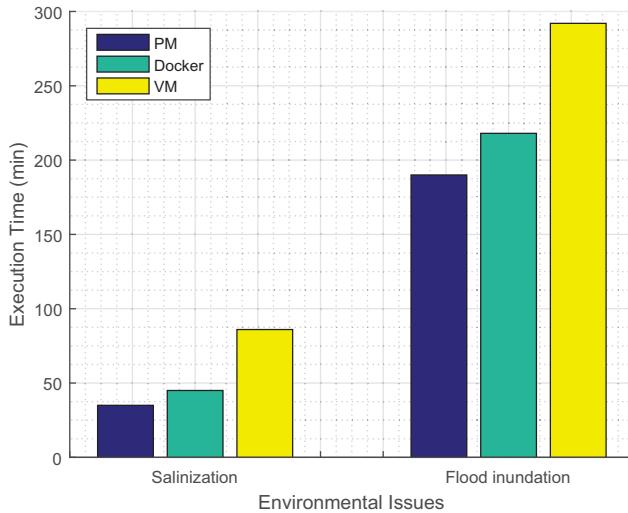


Fig. 7. Evaluation of simulating the real environmental issues in Vietnam by deploying on PM, Docker and VM.

VII. CONCLUSION AND FUTURE WORK

In this paper, we present a set of evaluations which aim to survey the performance as well as the efficiency of deploying TELEMAC on virtual platforms such as VM and Docker. Associated with the demand of parallel computing, this is the ability of providing the computing infrastructure for multiple users with different purposes that need to simulate the environmental issues. To keep up with these requirements, virtual platforms are the feasible solutions. In the previous works, we have done that deploying Docker on the advanced technologies, e.g., IB, Xeon-Phi co-processor. There are many good results about the performance of Docker in comparison to VM. Following that, this paper shows benchmarks and evaluations between Docker and VM when running the practical simulations by using TELEMAC. Based on the lightweight architecture, Docker has benefits of reducing overhead and remaining the execution time that is more efficiency than VM. We presented experiences for deploying TELEMAC on Docker environment. Our evaluation contributes a basic foundation to the trend that expands the computing environment of simulation applications on containers.

As a part of future work, we plan to develop a scheduling algorithm on Docker container. The algorithm aims to the allocation of resources for each container efficiently.

ACKNOWLEDGMENT

This research was conducted within the “Studying and developing a 50-100 TFlops HPC system” funded by Department of Science and Technology - HCMC & HCMUT (under grant number 21/2015/H-SKHCN).

REFERENCES

- [1] (2014) Telemac-mascaret homepage, the mathematically superior suite of solvers. [Online]. Available: <http://www.opentelemac.org/>
- [2] R. Issa, F. Decung, E. Razafindrakoto, E.-S. Lee, C. Moulinec, D. Latino, D. Violeau, and O. Boiteau, “Hpc for hydraulics and industrial environmental flow simulations,” in *Parallel Computational Fluid Dynamics 2008*. Springer, 2010, pp. 377–387.
- [3] J.-M. Hervouet, “Telemac modelling system: an overview,” *Hydrological Processes*, vol. 14, no. 13, pp. 2209–2210, 2000.
- [4] C. Boettiger, “An introduction to docker for reproducible research,” *ACM SIGOPS Operating Systems Review*, vol. 49, no. 1, pp. 71–79, 2015.
- [5] (2014) Docker homepage. [Online]. Available: <https://www.docker.com/>
- [6] M. T. Chung, A. Le, Q.-H. Nguyen, D.-D. Nguyen, and N. Thoai, “Provision of docker and infiniband in high performance computing,” 2016.
- [7] G. F. Pfister, “An introduction to the infiniband architecture,” *High Performance Mass Storage and Parallel I/O*, vol. 42, pp. 617–632, 2001.
- [8] G. Chrysos, “Intel® xeon phi coprocessor-the architecture,” *Intel Whitepaper*, 2014.
- [9] Z. Shang, “High performance computing for flood simulation using telemac based on hybrid mpi/openmp parallel programming,” *International Journal of Modeling, Simulation, and Scientific Computing*, vol. 5, no. 04, p. 1472001, 2014.
- [10] B. C. Pijanowski, A. Tayyebi, J. Doucette, B. K. Pekin, D. Braun, and J. Plourde, “A big data urban growth simulation at a national scale: configuring the gis and neural network based land transformation model to run in a high performance computing (hpc) environment,” *Environmental Modelling & Software*, vol. 51, pp. 250–268, 2014.
- [11] P. Wittek and X. Rubio-Campillo, “Scalable agent-based modelling with cloud hpc resources for social simulations,” in *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*. IEEE, 2012, pp. 355–362.
- [12] E. M. Dow *et al.*, “The xen hypervisor,” *INFORMAT*, dated Apr, vol. 10, 2008.
- [13] G. J. Popek and R. P. Goldberg, “Formal requirements for virtualizable third generation architectures,” *Communications of the ACM*, vol. 17, no. 7, pp. 412–421, 1974.
- [14] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, “kvm: the linux virtual machine monitor,” in *Proceedings of the Linux symposium*, vol. 1, 2007, pp. 225–230.
- [15] Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: state-of-the-art and research challenges,” *Journal of internet services and applications*, vol. 1, no. 1, pp. 7–18, 2010.
- [16] D. Merkel, “Docker: lightweight linux containers for consistent development and deployment,” *Linux Journal*, vol. 2014, no. 239, p. 2, 2014.
- [17] F. Ghassemi, A. J. Jakeman, H. A. Nix *et al.*, *Salinisation of land and water resources: human causes, extent, management and case studies*. Cab International, 1995.
- [18] M. T. Chung, Q.-H. Nguyen, M.-T. Nguyen, and N. Thoai, “Using docker in high performance computing applications,” in *Communications and Electronics (ICCE) (IEEE ICCE 2016), 2016 IEEE Sixth International Conference on*. IEEE, 2016, pp. 52–57.
- [19] C. MOULINEC, “Hpc evolution of the telemac system,” in *E-proceedings of the 36th IAHR World Congress, Hague, Netherlands*, 2015.
- [20] G. Merkuryeva, Y. Merkuryev, B. V. Sokolov, S. Potryasaev, V. A. Zelentsov, and A. Lektauers, “Advanced river flood monitoring, modelling and forecasting,” *Journal of Computational Science*, vol. 10, pp. 77–85, 2015.
- [21] C. Villaret, J.-M. Hervouet, R. Kopmann, U. Merkel, and A. G. Davies, “Morphodynamic modeling using the telemac finite-element system,” *Computers & Geosciences*, vol. 53, pp. 105–113, 2013.
- [22] R. Dua, A. R. Raja, and D. Kakadia, “Virtualization vs containerization to support paas,” in *Cloud Engineering (IC2E), 2014 IEEE International Conference on*. IEEE, 2014, pp. 610–614.
- [23] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, “An updated performance comparison of virtual machines and linux containers,” in *Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium On*. IEEE, 2015, pp. 171–172.
- [24] W. Gentzsch, “Linux containers simplify engineering and scientific simulations in the cloud,” in *Information and Computer Technology (GOCICT), 2014 Annual Global Online Conference on*. IEEE, 2014, pp. 22–26.
- [25] C. P. Wright and E. Zadok, “Kernel korner: unionsfs: bringing filesystems together,” *Linux Journal*, vol. 2004, no. 128, p. 8, 2004.

Environmental computing - applications, tools and data solutions

Matti Heikkurinen
 Munich Network Management Team
 Ludwig-Maximilians-Universität München
 Munich, Germany
 heikku@nm.ifi.lmu.de

Dieter Kranzlmüller
 Leibniz Supercomputing Centre
 Garching, Germany
 Dieter.Kranzlmueller@lrz.de

Abstract—Environmental computing is becoming a recognized specialty focusing on producing actionable knowledge by advanced environmental modelling. At the moment the environmental computing community shares a tacit understanding of what are the initiatives, tools and approaches that belong to the core scope of this emerging discipline. The environmental computing website[1] has been used as an interim community resource for collecting links to resources that are of relevance to this discipline. The information on the site has been collected using the presentations in environmental computing events (listed at the end of this summary) as starting points, complemented with some targeted desk research efforts. This presentation outlines the first attempts to analyse this initial dataset as a way to identify common topics, initiatives and approaches that could serve as basis for clustering of these initiatives.

There are several reasons to attempt to identify commonalities beyond the simple, intuitive categorisation of something being “within the scope of environmental computing”. From the pragmatic perspective, commonalities represent opportunities for identifying synergies, either through the reuse of existing tools or through joining forces to solve common problems. The clustering may also identify areas that could serve as links to other disciplines, potentially indicating a need for a special interest group or another sub-specialty. An example of such a case is data management (especially consistent metadata) that is mentioned as an issue on most of the presentations in the environmental computing events. While the data management issues are very closely aligned with the broader “big data” and “open data” research and policy activities, environmental computing data management has probably a distinct scope involving issues that may be specific to environmental computing (e.g. requirements stemming from the environmental computing applications aiming to provide decision support tools in urgent situations).

In the longer term, repeating this analysis using a more consistent and comprehensive search approach should help consolidation of the shared body of knowledge of the community. The goal of the presentation is to stimulate discussion on the community approaches and tools that could help the practitioners reach this goal. The challenge in building a shared, common “community hub” is that until the system and the active community around it reaches the critical mass, the efforts needed to add information to the repository and curation of the already

stored data will not bring immediate benefits in day-to-day work. Thus a shared vision and roadmap are crucial success factors.

The presentation also highlights some of the semantic challenges, e.g. environmental computing organisations that have departments with distinct profiles necessitating a conceptual model that is capable to deal with “nested” entries. Limitations of the commodity tools in keeping track and cross-linking publications, datasets and tools will also be discussed as a topic for further study.

Keywords—Environmental computing, body of knowledge, survey methods.

MAJOR ENVIRONMENTAL COMPUTING EVENTS SO FAR

The authors consider have used the following environmental computing events as starting points of the analysis:

- “Environmental Supercomputing and disaster risk reduction”[2] – side event during the UN World Conference on Disaster Risk Reduction, March 2015
- eScience 2015[3] Environmental computing focus day, September 2015
- ISGC 2016 Environmental computing workshop[4], March 2016
- ICCS 2016 workshop [5], June 2016
- ISC BoF session[6], June 2016

REFERENCES

- [1] Environmental computing website – www.envcomp.eu (accessed 30th June 2016)
- [2] Side event website – <http://www.wcdrr.org/conference/events/696> (accessed 30th June 2016)
- [3] eScience 2015 website - <http://escience2015.mnm-team.org/> (accessed 30th June 2016)
- [4] Workshop homepage - <http://www.envcomp.eu/ISGC16> (accessed 30th June 2016)
- [5] Workshop homepage - <http://www.envcomp.eu/ICCS16> (accessed 30th June 2016)
- [6] BoF homepage - <http://www.envcomp.eu/ISC16> (accessed 1st July 2016)

AUTHOR INDEX

A

- Abu Jabal, Amani 270
Agarwal, Deb 389
Ahmad, Yanif 130
Alvanaki, Foteini 424
Alzuru, Icaro 41
Amaral, Pedro 111
Ananthakrishnan, Rachana 203
Andes, Alicia 251
Arabnejad, Vahid 137
Artés, T. 414
Asuncion, Hazeline 175

B

- Babuji, Yadu N. 337
Balasubramanian, Vivekanandan 361
Balla, Andre 165
Barnas, Andrew 223
Bartholomew, Amelia 165
Bartok, J. 432
Belhajjame, Khalid 71
Bertino, Elisa 270
Bethel, E. Wes 213
Bethune, Iain 361
Bierkens, Marc 430
Black, Edgar F. 165
Bocca, R. 414
Boverhof, Joshua 389
Bowley, Connor 251
Breitmoser, Elena 361
Brooke, John 282
Bubendorfer, Kris 137
Budavári, Tamás 130

C

- Cała, Jacek 81
Candy, Adam 399
Carey, Nicholas 130
Carvalho, Lucas A.M.C. 71
Cavoto, Patricia 51
Cesar Junior, Roberto Marcondes 243
Chard, Kyle 203
Cheah, You-Wei 389
Chen, Peng 313
Chiaravaggio, N. 414
Chung, Minh Thanh 441
Clementi, Cecilia 361
Clements, Mark 323
Comin, Cesar Henrique 243
Cortés, A. 414
Cunha, Jose C. 120
Czajkowski, Karl 31

D

- da Fontoura Costa, Luciano 243
Davis, Delmar 175
Deelman, Ewa 400
De Roure, David 185
Desell, Travis 223
Dias, Paulo 111
Dijkstra, Henk 399
Dobrucký, M. 432
Donnelly, Peter D. 303
Dorier, Matthieu 371
dos Reis, Julio Cesar 51
Drost, Niels 430
Duede, Eamon 337

E

- Eichinski, Philip 147
Elbashandy, Abdelrahman 389
Ellis-Felege, Susan 223
ElSaid, AbdElRahman 260
Epperly, Thomas 389
Eslick, John 389
Evans, Tom 313

F

- Ferreira da Silva, Rafael 400
Filgueira, Rosa 400
Fortes, José A.B. 41
Foster, Ian 203
Frisby, Michael 313

G

- Garcia, Jordi 276
Garijo, Daniel 331
Gelderloos, Renske 381
Gentile, A. 414
Gerow, Aaron 337
Gil, Yolanda 331
Goncalves, Carlos 120
Goncalves, Romulo 424
Greenwald, Martin 213
Grimshaw, Andrew 1
Grupe, Gisela 233
Gunasinghe, Hasini 287
Gusew, Wladislaw 155

H

- Haine, Thomas W.N. 381
Harpham, Quillon 431
Heikkurinen, Matti 449
Hiden, Hugo 21
Higgins, James 260
Hluchý, L. 432
Hölzl, Stefan 233
Hruska, Eugen 361
Hut, Rolf 430
Hwang, Soonwook 193

I

- Iles, David 223
Ivie, Peter 61
Izquierdo, Eduardo 313

J

- Jha, Shantenu 361
Jiang, Li 101

K

- Karlsson, Andreas 323
Kawashima, Hideyuki 101
Kenyon, Norma 165
Kesselman, Carl 31
Kim, Jik-Soo 193
Kleese van Dam, Kerstin 213
Klyne, Graham 185
Knepper, Richard 1
Kougkas, Anthony 371
Kranzlmüller, Dieter 449
Kröger, Peer 233
Kumar, Geet 11
Kyzirakos, Kostis 424

L

- Latham, Rob 371
Laughton, Charles 361
Laure, Erwin 323
Leek, Jim 389
Lemson, Gerard 381
Li, Tonglin 11
Li, Xiang 355
Lidman, Mattias 203
López, J. 414

M

- Malik, Tanu 355
Margalef, T. 414
Marini, Luigi 165
Marín-Tordera, Eva 276
Marru, Suresh 287
Masip-Bruin, Xavier 276
Matsunaga, Andréa 41
Mattingly III, Marshall 223
Mauder, Markus 233
Mayani, Rajiv 400
Mayer, Benjamin 400
Mayr, Christoph 233
McCollam, Brendan 203
McHenry, Kenton 165
Medeiros, Claudia Bauzer 71
Meng, Haiyan 91
Meng, Zeqian 282
Miller, David 389
Mishra, Saurabh 331
Moreira, José 111

N

- Nakandala, Supun 287
 Nepal, Surya 293
 Newman, Robert 223
 Ng, Bryan 137
 Nguyen, B.M. 432
 Nguyen, Cao 193
 Nguyen, Manh-Thin 441
 Nguyen, Thong 441
 Nguyen-Huynh, Nhu-Y 441
 Nourian, Pirouz 424
 Ntoutsi, Eirini 233

O

- Olofsson, Niten 323
 Otto, Friederike E.L. 407

P

- Page, Kevin R. 185
 Pantoja, Fagner Leal 51
 Parashar, Manish 213
 Patil, Rachana 165
 Pelupessy, Inti 399
 Perez Cervantes, Evelyn 243
 Pfeffer, William 175
 Pierce, Marlon 287
 Plale, Beth 313
 349
 Portegies Zwart, Simon 399
 Prodhan, Md Anindya 1
 Pybus, John 185

Q

- Qasha, Rawaa 81

R

- Raicu, Ioan 11
 Rashid, Md Mamunur 407
 Ratnakar, Varun 331
 Roe, Paul 147
 Rosen, Stephen 203
 Ross, Rob 371
 Rynge, Mats 400

S

- Sadooghi, Iman 11
 Sakellariou, Rizos 282
 San Miguel Ayanz, J. 414
 Santanchè, André 51
 Scheuermann, Björn 155
 Schuler, Robert E. 31
 Sethi, Ricky J. 343
 Shkurti, Ardita 361
 Silva, Joaquim F. 120
 Sinaeepourfard, Amir 276
 Sinnott, Richard 293
 Šípková, V. 432
 Stewart, Craig 1
 Sun, Xian-He 371
 Suriarachchi, Isuru 349
 Sutanudjaja, Edwin 430
 Szalay, Alexander S. 130

W

- Wallom, David C.H. 407
 Wang, Ke 11
 Wang, Nan 303
 Wang, Shuo 293
 Watson, Paul 21
 81
 Weigl, David M. 185
 Wilcoxson, Matthew 185
 Wild, Brandon 260
 Wild, Stefan M. 213
 Wiley, H. Steven 213
 Willcox, Pip 185
 Wolf, Matthias 91
 Woodard, Anna 91
 Woodman, Simon 21

X

- Xu, Xiaoyang 355

Z

- Zhao, Dongfang 11
 Zhao, Yan 165
 Zlatanova, Sisi 424

T

- Tan, Harold Yih-Chi 440
 Tatebe, Osamu 101
 Thain, Douglas 61
 91
 Thoai, Nam 441
 Thomas, Alexander 1
 Toncala, Anita 233
 Tsugawa, Maurício 41
 Tuecke, Steven 203

U

- Uhe, Peter 407

V

- Vahi, Karan 400
 van de Giesen, Nick 430
 van Elteren, Arjen 399
 van Hage, Willem 424
 van Meersbergen, Maarten 430
 van Werkhoven, Ben 399
 Varghese, Blesson 303
 Vasudevan, Subha 175
 Viebahn, Jan 399
 Vyushkov, Alexander 91