The Future of Scientific Workflows

Report of the DOE NGNS/CS Scientific Workflows Workshop
April 20–21, 2015

Sponsored by the Office of Advanced Scientific Computing Research
U.S. Department of Energy
Office of Science

The Future of Scientific Workflows

Report of the DOE NGNS/CS Scientific Workflows Workshop
Rockville, Maryland
April 20-21, 2015

Meeting Organizers

Ewa Deelman (USC) (co-organizer)
Tom Peterka (ANL) (co-organizer)
Ilkay Altintas (SDSC)
Christopher Carothers (RPI)
Kerstin Kleese van Dam (PNNL)
Kenneth Moreland (SNL)
Manish Parashar (RU)
Lavanya Ramakrishnan (LBNL)
Michela Taufer (UD)
Jeffrey Vetter (ORNL)

Meeting Participants

Greg Abram (UT)
Gagan Agrawal (OSU)
Jim Ahrens (LANL)
Pavan Balaji (ANL)
Ilya Baldin (RENCI)
Jim Belak (LLNL)
Amber Boehnlein (SLAC)
Shreyas Cholia (NERSC)
Alok Choudhary (NU)
Constantinos Evangelinos (IBM)
Ian Foster (ANL)
Geoffrey Fox (IU)
Foss Friedman-Hill (SNL)
Jonathan Gallmeier (AMD)
Al Geist (ORNL)
Berk Geveci (Kitware)
Gary Grider (LANL)
Mary Hall (UU)
Ming Jiang (LLNL)
Elizabeth Jurrus (UU)
Gideon Juve (USC)
Larry Kaplan (Cray)
Dimitri Katramatos (BNL)

Dan Katz (UC)
Darren Kerbyson (PNNL)
Scott Klasky (ORNL)
Jim Kowalkowski (FNAL)
Dan Laney (LLNL)
Miron Livny (UW)
Allen Malony (UO)
Anirban Mandal (RENCI)
Folker Meyer (ANL)
Dave Montoya (LANL)
Peter Nugent (LBNL)
Valerio Pascucci (UU)
Wilfred Pinfold (Intel)
Thomas Proffen (ORNL)
Rob Ross (ANL)
Erich Strohmaier (LBNL)
Christine Sweeney (LANL)
Douglas Thain (UND)
Craig Tull (LBNL)
Tom Uram (ANL)
Dean Williams (LLNL)
Matt Wolf (GT)
John Wright (MIT)
John Wu (LBNL)
Frank Wuerthwein (UCSD)
Dantong Yu (BNL)

DOE Program Managers
Richard Carlson
Lucy Nowell

Table of Contents

1. Executive Summary

Today's computational, experimental, and observational sciences involve growing science collaborations that require interactive, collaborative access to data analysis and visualization services. Experimental instruments generate increasing amounts of data that necessitate on-the-fly processing. Applications involve computationally-intensive codes that also generate large amounts of data and adopt in situ analysis methodologies. Often the computations are conducted as a workflow of many related tasks. The success of the U.S. Department of Energy (DOE) scientific mission hinges on the computer automation of these workflows. In April 2015, a diverse group of domain and computer scientists from the Office of Science, the National Nuclear Security Administration, industry, and academia assembled in Rockville, Maryland, to review the workflow requirements of DOE's science and national security missions, to assess the current state of the art in science workflows, to understand the impact of emerging extreme-scale computing systems on those workflows, and to develop requirements for automated workflow management on those systems.

The mission of this workshop was to develop requirements for workflow methods and tools in a combined high-performance computing (HPC) and distributed-area computing (DAC) work environment, in order to enable science applications to better manage their end-to-end data flow. HPC and DAC workflows will be defined more carefully in the next section; but for the time being, a *workflow* in this context is the composition of several computing tasks. An *HPC* workflow is a one whose tasks are coupled by exchanging information over the memory/storage hierarchy and network of current leadership-class DOE supercomputing architectures and future extreme-scale machines. A *DAC* workflow is one whose tasks are more loosely coupled through files and that execute on geographically distributed clusters, clouds, and grids.

The workshop had the following high-level objectives:

- Identifying the workflows of representative science use cases in HPC and DAC
- Understanding the state of the art in existing workflow technologies, including creation, execution, provenance, (re)usability, and reproducibility
- Addressing emerging hardware and software trends, in both centralized and distributed environments, as they relate to workflows
- Bridging the gap between in situ HPC and DAC workflows.

Two categories of extreme-scale drivers were investigated:

- *Application requirements of science workflows.* workflows used by computational sciences, observations from sensors and other instruments, experiments at user facilities, and the collaborations that such teams need to conduct those activities were studied.
- *Extreme-scale computing systems*. the hardware and software subsystems in current, next-generation, and extreme-scale HPC and DAC systems, to the extent that they

interact with workflow systems. Also identified were gaps and opportunities to influence those systems as they are being designed.

As a result of the trends listed above, the workshop identified five main research areas:

- *System design and execution*. The most important factors in designing the workflow management system (WMS) are scalable and robust control and data flow, data management and triage, workflow management and monitoring, provenance capture, and fault tolerance and recovery.
- *Programming and usability*. Ease of use remains the single biggest barrier to adoption of workflow technologies. Programming models, design patterns, the user interface, task communication, and portability of individual task modules are potential areas for improvement.
- *Provenance capture.* Provenance is the key to validation and reproducibility of any scientific process, and the WMS is the natural place to capture provenance data. Relevant topics include the content and format of provenance data (including the ability to customize to scientists' needs), capture mechanisms, communication of metadata across system software levels, short-term storage and long-term archival, and data-mining algorithms to distill raw provenance data to its essentials.
- *Validation.* The validation of a workflow execution involves (a) comparing performance of the entire execution and of each component against predictions based on models, (b) comparing the science results with expectations (models, auxiliary methods, invariants), and (c) being able to reproduce the workflow on the same or another computing environment.
- *Workflow science*. The formalism of theories, models, and experiments in workflows may be embodied in a new field, similar to data science, called workflow science. Training will enable the next generation of workflow scientists to conduct research in workflows.

An investigation into the drivers and workflow research areas above resulted in the following high-level findings. Sections 3 and 4 of this report present a detailed explanation of these findings, including state of the art, research challenges, and specific recommendations of research and development activities.

- The computer science community needs to better understand science processes in simulations, instruments (experiments and observations), and collaborations in order to be able to design workflow management systems (WMSs) that facilitate those processes.
- An understanding of extreme-scale architectures is needed in order to design WMSs for that hardware and to influence the design of those architectures to support workflow execution.
- The intersection of workflow systems and system software is not well understood. Research is needed in the delineation of the roles of the WMS and the operating system/ runtime (OS/R) in the areas of resource management and scheduling.
- The design of control and data flows, data models, and programming interfaces needs further research in the general area of WMS design.

- During and after the workflow execution, the capture of provenance information and its use to validate performance and correctness and to support data reuse and repurposing are areas where much research is needed.
- Benchmarks and community data sets are needed to drive workflow research.

Many of these efforts can be systematically described in a new body of research called workflow science, which studies the theory, modeling, and benchmarking of workflows.

2. Introduction

Although a scientist's workflow can include all the steps in the science discovery process, from forming hypotheses to disseminating results, we limit the scope of the term *workflow* to mean a subset of those tasks involving the collection, generation, and processing of numerical data that can be automated with computer systems. The term workflow, therefore, refers to sequencing and orchestrating operations, along with moving data among those operations. Systems that aid in the automation of these processes, freeing the scientist from the details of the process, are called *workflow management systems* (WMSs).

In the context of scientific computing, a workflow is the orchestration of multiple tasks in the course of a science campaign. Examples are computational simulations and data analysis and visualization software. To use a programming analogy, workflows can be considered "programming in the large": workflows are to programs what programs are to functions or subroutines. In other words, the workflow is the outer structure that ties together the individual programs. A large-scale science campaign is likely to consist of several interrelated workflows.

Workflows perform two basic functions. They manage the execution of constituent programs and the information exchanged between them. Therefore, an instantiation of a workflow must represent both the operations and the data products associated with a particular scientific domain. It should be assumed that individual operations and data products were developed independently, potentially by different scientists or communities. Workflows must be usable by the target audience (e.g., computational scientists) on target platforms (i.e., computing environments and networks), while being represented by abstractions that can be reused across sciences and computing environments and whose performance and correctness could be modeled and verified.

Workflows may have high-performance and distributed-area computing characteristics. In this report, we will refer to these modes as HPC and DAC, respectively. HPC workflows are executed on one or more clusters or supercomputers in the same physical facility. DAC workflows are executed across systems that are geographically more widely distributed, including grid and cloud settings. One of the goals of the workshop is to share information between the HPC and DAC workflow communities. In the past, research in these areas was funded, conducted, and disseminated separately. Realizing that a scientist's complete workflow may include both modes, another goal of the workshop is to identify the needed research to develop an interface between HPC and DAC subworkflows when they are parts of a larger workflow that includes both execution environments.

In situ workflows. Extreme-scale workflows face particular challenges, especially workflows designed for current- and next-generation leadership-scale HPC environments. These challenges include power, performance, resilience, and productivity: heterogeneous computing cores, new memory systems, and small or no growth in bandwidth to external storage systems are some of the main hurdles for HPC workflows at scale. The challenges motivate an important category of HPC workflows: in situ workflows. The term *in situ* has different meanings to different people; but in this report we simply mean multiple tasks running on the same supercomputer within a fixed time interval (e.g., a job submission). A concrete example is an analysis running on the same supercomputer where the simulation is currently running. We do not differentiate between various "flavors" of this definition such as whether the tasks execute in the same or separate resources (nodes, cores, etc.) within the supercomputer. By definition, an in situ workflow does not involve moving data off the supercomputer (to a storage system) for post hoc analysis.

In situ workflows contain a coupling of very large-scale tasks, containing hundreds of millions of processes, operating in a specialized environment that is comparatively compute-rich but data-movement-poor, despite specialized interconnection fabrics and parallel storage systems. The usage model, at least today and for the foreseeable future, is batch-scheduled. This is especially important as we contemplate human-in-the-loop interactive analysis in the context of in situ workflows. The biggest driver for creating a WMS for in situ HPC workflows is the need to capture provenance for interactive data analysis and visualization, as well as any computational steering that results. We have to support publication of results, validation of results, and sharing data; and interactive analysis is the element that makes a WMS essential for these tasks, as well as for the potential reuse and/or repurposing of the tiny stream of data that will be saved for post hoc use. The in situ use case is so critical to the success of HPC workflows that, in the remainder of this report, the two terms—in situ workflow and HPC workflow—can be used interchangeably unless otherwise noted. When we do not qualify the type of HPC workflow (e.g., post hoc), we mean in situ.

Many other challenges obstruct the deployment of production workflows at extreme scale. For example, there are multiple intermediate representations of data. Significant improvements in data abstractions and their representations are required in order to support scientists' specialized data uses. The data also must be augmented with provenance information, which. is important for validation of results. Workflows currently capture a great deal of relevant provenance information, but it is unclear how this information from workflows can best be captured and managed at the exascale. Since an exascale machine may have billion-way concurrency, determining the provenance of one billion interrelated parallel tasks will be exceptionally complex, if not infeasible. Even if this information exists in system log files, a key issue is whether scientists want that level of detail for their use. Workflows spanning multiple systems also must capture provenance information about all the steps executed on all systems. The WMS presumably has a record of what was done in an individual system, but there is no formal way to construct, capture, and manage heterogeneous provenance data in an interoperable manner.

In the HPC world today, scientists use Python or shell scripts to specify workflows, or they integrate the workflow directly into their simulation code. In these and other workflow programming models, we need to understand the best methods of interfacing WMSs and their tasks. For example, it is unclear how best to present "locality" in the context of a deep memory hierarchy so that decisions can be made regarding whether to move computation to data or data to computation or whether to further partition computation or data in order to facilitate a mix of approaches. In such cases, the WMS may need to communicate requirements to the operating and runtime (OS/R) systems, which would execute on behalf of the WMS. More research is needed to understand the interface between WMS and the OS/R systems. The overlap between "big data" programming models and scientific data programming models is another challenge. MapReduce is an important big data model, and initial activities have explored its utility for scientific data; but deeper study of programing models for big data workflows is required for use in scientific workflows.

Together with other parts of the computing ecosystem, the following features must be addressed in future HPC and DAC systems as part of, or in support of, extreme-scale workflows.

- *Programming models for workflows and their components*. Managing various and possibly heterogeneous software stacks, expressing tasks and their relationships productively and portably, and defining data models and their semantics.
- *Resource selection and provisioning*. Allocation of various types of resources for the generation, movement, analysis, and retention of data in a workflow, with particular attention to heterogeneity (nodes vs. cores, virtualization, memory hierarchy), power and time costs of moving data, and centralized and distributed systems for storage and staging data.
- *Scheduling*. Coordination of task launching and data transfers over all of the above resources during the staging and execution of a workflow
- *Fault tolerance and performance monitoring at runtime*. Monitoring the infrastructure and applications, understanding workflow behavior (modeling, anomaly detection, and diagnosis), detecting, isolating, and recovering from hard and soft errors, and maintaining security.
- *Provenance tracking and validation.* Validating accuracy of results, validating performance logs, and ensuring reproducibility.

In order to better understand the scope of challenges such as those above, the workshop focused on the following five major topics. For each topic, some of the driving questions that were used to seed discussions are listed below. Prior to the workshop, we also invited participants to submit white papers on these same issues. Those white papers are included in the workshop website.[1]

Science applications and use cases of workflows. What are the needs of science communities in terms of managing large-scale workflows? What are the current practices? How are the technological trends (better instruments, bigger data, different computing capabilities) expected to affect future work?

[1] http://extremescaleresearch.labworks.org/events/workshop-future-scientific-workflows

State of the art in DAC and HPC workflow management systems. What is the state of the art in DAC and in HPC? What are the strengths and weaknesses of the DAC and HPC workflows? What are the common functions in DAC and HPC workflows?

Impact of emerging architectures on workflow systems. How do new extreme-scale architectures affect the DAC and HPC workflows? Possible considerations include power, concurrency, heterogeneity, bottlenecks, scheduling and use policies including potentially supporting interactive exploration of data, and the severe data triage needed for HPC workflows.

Future needs for extreme-scale DAC and HPC workflows. What are the challenges for DAC and HPC workflows going forward? These may include data management (including how to select the small percentage of data that can be saved to persistent storage), computation scheduling, usability, verification, provenance, fault tolerance, and performance.

Interface between DAC and HPC workflow systems. In an overall science campaign, what is the interface between HPC and DAC workflows? How can the gap be bridged? How is information (data, computation, and metadata products) transferred between HPC and DAC systems? Considerations include the interface between different types of WMSs, different software and hardware environments, different communities of users, and different objectives of the workflows.

3. Extreme-Scale Drivers

Summary

Two categories of extreme-scale drivers motivate the future of science workflows. Science applications (Section 3.1) include computational science, experimental and observational data, and collaborations in support of those activities. Extreme-scale computing systems (Section 3.2) include the hardware and software subsystems in current, next-generation, and extreme-scale HPC and DAC systems, to the extent that they interact with workflow systems. Also identified are gaps and opportunities to influence these systems as they are being designed.

One must understand the present state of scientific workflows before studying their future. Hence, existing workflows for both computational and experimental sciences are described. In computational science, workflows are used to link multiphysics codes together, to perform parameter sweep studies that take into account different initial conditions, and to link simulations with analysis or visualization tasks. Observational and experimental science face exponentially growing data volume and velocity, increasing the need for workflows to coordinate data collection with processing. Such workflows may be widely distributed because the scientific instruments and computing resources are seldom collocated. Simulations and experiments may also be combined; for example, synchrotron light source imaging experiments have recently been coupled with molecular dynamics simulations.[2] Simulations may be used to design experiments or observations, to fill in gaps in data, and to test hypotheses. Experimental and observational data may be used to seed a simulation with starting conditions. The relationships are complicated and becoming more so.

Some workflows also support collaboration. Collaborations range anywhere from a small team of scientists using a custom set of software tools, to public contributions, to web-based citizen-science projects. Sharing workflows, migrating workflows between different computing environments, accommodating different user roles (e.g., biologist, computer scientist, data analyst), and combining different languages and software tools used by various users are the challenges. How to address those challenges are open questions today.

Changes in hardware for extreme-scale supercomputing systems pose major challenges for workflows in terms of power, performance, resilience, and productivity. These changes require significant planning and investment in system software, programming environments, applications, and WMSs. Workshop participants identified heterogeneous computing, new memory systems including nonvolatile memory, and small or no growth in bandwidth to external storage systems or networks as the main hurdles that must be cleared in order to deploy workflows at scale. Arguably, the biggest challenge for science is the worsening I/O bottleneck. I/O rates are expected to be flat for the next decade. Some hardware designs can still be

[2] http://www.anl.gov/imaging/project/maui-modeling-analysis-and-ultrafast-imaging

influenced by research: vendors are eager to include proxy applications representative of workflows in their design activities and machine acceptance tests.

Some perceived challenges, such as NVRAM to extend the traditional system memory, may actually be opportunities in disguise, provided the right interfaces exist to use them. Other hardware trends, such as decreased storage bandwidth relative to the compute rate, require scientists to change their usage patterns: for example, by selecting in situ workflows as opposed to post hoc ones. One of the surprising outcomes from the discussion about managing resources is that human productivity is arguably still the most expensive resource, trumping power, performance, and other factors. While the focus of the workshop was not scientific productivity per se, the usability of new hardware and software technologies and the gap between their research and mainstream use merits continued research. Programmability and usability of workflows are addressed in the following section.

The software systems on HPC machines pose a different set of challenges for workflows. First, HPC systems are still intended to run single-program batch jobs. Even though they run many such jobs simultaneously, those jobs are intentionally separated in both hardware (different partitions of nodes) and software (different address spaces and software stacks). Workflows by definition are collections of multiple *coordinated* programs. Interactivity is another feature not supported in HPC systems today. Human interaction with the workflow (for example, to steer a computation based on partial results) is the opposite of batch scheduling. Some of these hurdles require technological innovations, while others are matters of policy (for example, current requirements to maximize core occupancy or impose storage quotas per job). Detecting and responding to faults and other unexpected occurrences in hardware and software layers are active areas of research in HPC systems. Workflows compound the difficulty in detecting and responding to errors because of the collocation of heterogeneous tasks and the addition of extra layers in the software stack.

Schedulers and resource managers must treat storage, I/O, and network capacity as first-class resources to be allocated, managed, and measured to the same degree as computing capacity is today. When the workflow consists of several subworkflows spanning multiple systems, scheduling resources over several systems will require cooperative schedulers that can coordinate with the WMS to manage the individual machines’ schedules.

At the end of the day, the role of the OS/R is to provide well-understood, predictable, and reproducible services to the WMS and, in turn, to the scientist. Predictability can be improved by effective resource allocation or adaptation to conditions. In order to facilitate reproducibility, system components such as batch schedulers, resource allocators, and file systems will need to be sufficiently transparent such that provenance data can be extracted at runtime, whether by the OS/R or the WMS.

Findings

- Application requirements
    - Research is needed in the areas of automation, human-computer interaction, provenance, and validation in order to support in situ workflows.
    - To address the challenges of experimental and observational workflows, the WMS must coordinate the end-to-end workflow life cycle, including real-time scheduling and execution of measurement instruments and analytic platforms, which may include supercomputers.
    - Workflows also need to be more adaptive. Their resource needs may change during the course of a computation or experiment, even to the point of triggering entirely new tasks based on the evolution of certain phenomena.
    - Collaborative workflows are characterized by heterogeneous users and resources, and WMSs must be portable, shareable, and reproducible across a diverse set of environments in order to be usable by all members of a collaboration.
- Hardware systems
    - Metrics are needed for enabling management of performance, power, and productivity. How to measure productivity is an open question. Benchmarks and proxy applications to test workflow workloads are needed to measure such metrics.
    - Mechanisms for passing data between tasks without unnecessary data movement should be investigated.
    - Memory management systems that support scratchpad and NVM are needed for workflows. Increasing the number of computing tasks in the same memory footprint (given that DRAM memory per core is expected to decrease in extreme-scale machines) will increase the likelihood that workflows will need to extend memory to NVM. Moreover, the potential for more complicated analyses that manage more state (for example, several time steps of data) offered by workflows will further require the extended footprint offered by NVM.
- Software systems
    - Efficient low-overhead scheduling of multiple cooperative tasks, various forms of communication (messages, interrupts, publish-subscribe, etc.) between independent tasks, and provisioning of shared resources (e.g., shared storage) among tasks are needed from the OS/R to support the WMS.
    - The WMS needs the OS/R to move from its traditional role of managing single tasks to managing global (i.e., internode, distributed) services. True, a supercomputer usually runs more than one application at a time, and those jobs are managed at some level of global system software. Today, however, user-space jobs are isolated from the others by design, and even OS instances are isolated, with each job booting a new image of the OS (micro)kernel. Research is needed to develop services and expose them to higher levels of the software stack so that the WMS and users can access them. Such global management may be through a hierarchy of resource groupings (enclaves), with

heterogeneous programming models / runtimes managing the resources within a given enclave or task.

- The WMS needs to negotiate with the OS/R through a well-defined interface on behalf of the entire application workflow. The OS/R must provide the WMS with the system calls to coordinate various tasks (such as the simulation and data analysis codes) and capture the provenance that scientists need to support writing papers and validating scientific results, including capturing any changes from the initial workflow that result from human-in-the-loop interactive analysis and steering.

3.1 Application Requirements

Data generated from simulations and instruments are growing rapidly as a result of the increased fidelity and frequency of computations and the increasing sensitivity experimental instruments. In this section, we examine a few key application characteristics from the DOE Office of Science in order to understand the state of the art, challenges, and R&D needed for simulations, experiments, observations, and collaborations.

3.1.1 Simulations

State of the Art:
Today, HPC workflows are usually constructed from Python scripts or from hard-coded functions embedded directly in simulation code. The use of workflow tools for DAC varies widely across communities. Some communities have used existing workflow tools to compose and execute their workflows. For example, the Kepler WMS [1] is used by the Center for Edge Physics Simulation for monitoring and guiding a particular simulation. The Pegasus WMS [21] is used to execute climate modeling workflows and material science analysis of Spallation Neutron Source data. On the other hand, many communities have developed specific tools or infrastructure to manage their workflows (e.g., LSST, JGI). These tools often support monitoring and provenance collection. Workflows in ad hoc scripts are also fairly commonly developed and used. Figure 1 shows the Accelerated Climate Modeling for Energy (ACME) workflow for climate modeling.

Many simulation workflows are complex and consist of a large number of simulation codes with many tunable parameters and input data sets. The complexity of the applications coupled with the complexity of the underlying hardware often means that much human effort goes into setting up these workflows on HPC machines. In addition, many simulation codes (e.g., in materials science) require human intervention to arrive at converged, reliable results. Simulations can fail and sometimes need a user to intervene; for example, the arbitrary Lagrangian-Eulerian simulation can fail because of mesh tangling or mesh overrelaxation and in such cases requires real-time human intervention. Increased automation to replace some of the human intervention, and supporting human intervention when it absolutely is needed, can improve such workflows.

Workflows today are able to capture static task structure and dependencies. However, users often have to rely on their own codes to manage conditional and/or dynamic execution. For example, the lattice QCD vacuum gauge configuration workflow, in addition to a resource-intensive generation step and an overall campaign management level, has decision points where additional chains may be formed.

Challenges:
Constrained I/O bandwidth in exascale architectures will necessitate in situ analysis and visualization. Workflows provide an opportunity to accelerate in situ coupling in the HPC

ecosystem. However, complex interactions with the HPC resources—batch queues, burst buffers, storage and network resources—present new challenges for workflow automation. Many operations that used to be performed by individual tasks (e.g., through a library interface) must be redesigned in the context of a workflow where they affect other tasks. Launching tasks, coordinating information in data flows, allocating short- and long-term buffering space, providing resilience to faults, and capturing provenance will need to become systemwide services to which tasks can subscribe. In short, the hard-coded method of writing in situ analytics directly into simulations, while simple to implement, has reached its limits. There is no way to accommodate new requirements—different data rates, dynamic task launching, resilience, or provenance—in the current state of the art.

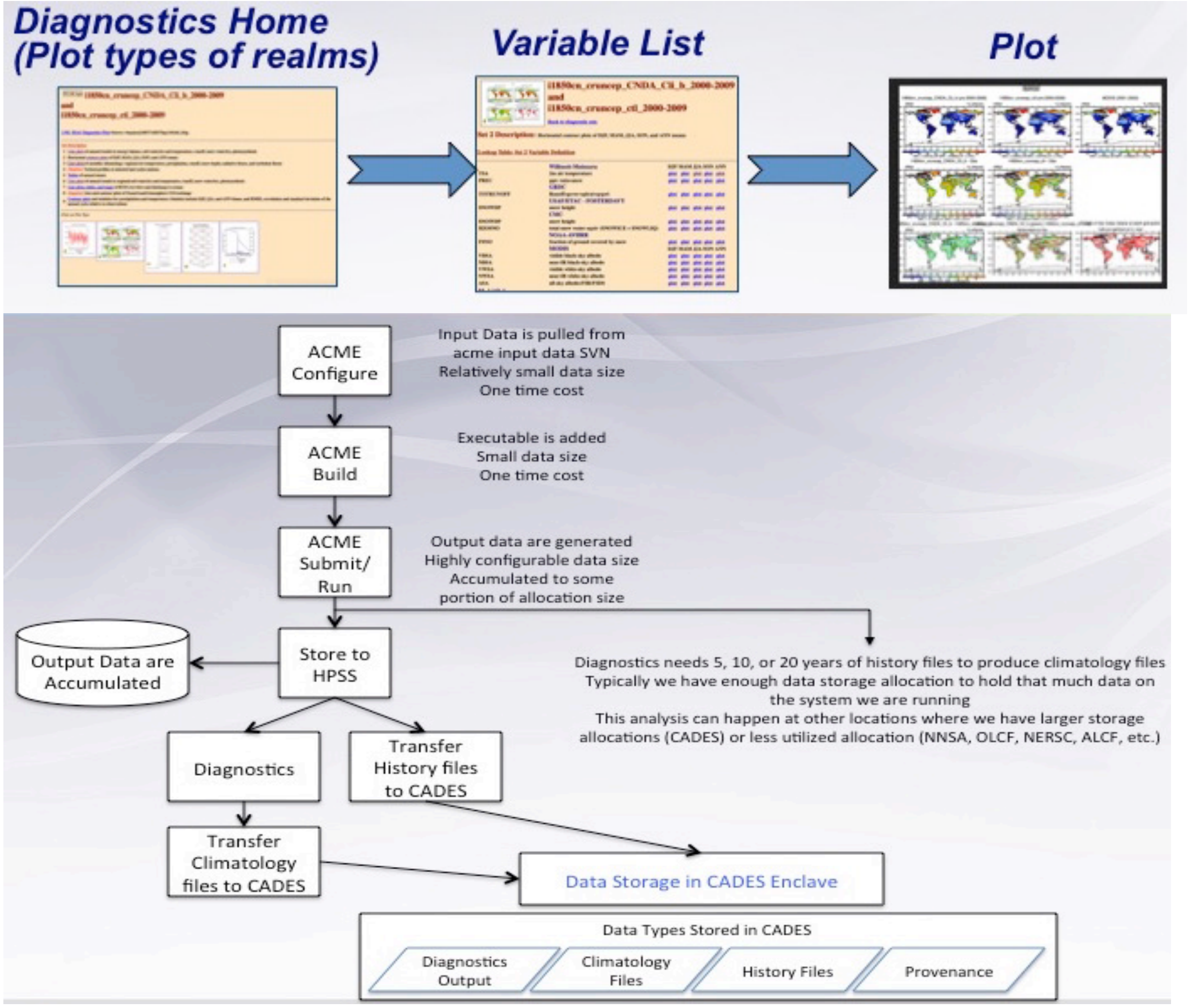

Diagnostics Home
(Plot types of realms)
Variable List
Plot
ACME Configure
Input Data is pulled from acme input data SVN
Relatively small data size
One time cost
ACME Build
Executable is added
Small data size
One time cost
ACME Submit/ Run
Output data are generated
Highly configurable data size
Accumulated to some portion of allocation size
Output Data are Accumulated
Store to HPSS
Diagnostics needs 5, 10, or 20 years of history files to produce climatology files
Typically we have enough data storage allocation to hold that much data on the system we are running
This analysis can happen at other locations where we have larger storage allocations (CADES) or less utilized allocation (NNSA, OLCF, NERSC, ALCF, etc.)
Diagnostics
Transfer History files to CADES
Transfer Climatology files to CADES
Data Storage in CADES Enclave
Data Types Stored in CADES
Diagnostics Output
Climatology Files
History Files
Provenance

Figure 1: Top: ACME diagnostics and analysis workflow. The workflow is used to quickly evaluate models and validate their results. The Python scripts from UV-CDAT produce climatology files and static HTML and plots (gif, jpg, etc.) Bottom: Data storage workflow of the DOE Earth system model.

R&D Needed:
The following key research challenges need to be addressed in order to support effective and efficient simulation workflows.

1. Automation: Simulation workflows still rely heavily on user intervention for setup, convergence, and failure recovery. Research is needed into methods (e.g., machine learning algorithms) that leverage knowledge of the simulation and resources, in order to increase the level of automation in these workflows and to minimize user intervention.

2. Human in the loop: While it is important to increase automation, it is impossible to replace human knowledge and input. Human-computer interaction research is needed to study and identify human-in-the-loop scenarios and devise methods and user interfaces that allow seamless interaction with the workflow-based applications.

3. Provenance and validation: Today, workflow tools support provenance collection as part of the workflow. However, it is important to consider whether the provenance collected is sufficient for validation and to determine how such validation might be performed: scientists may want control of the provenance capture and the ability to specify what is collected and how. Additionally, we need more research to identify how predictive analytics of provenance data can be used for validation of workflows.

3.1.2 Instruments

State of the Art:
Scientific facilities and associated instruments are increasingly generating large amounts of data thanks to improvements in underlying hardware and software technologies. For example, recent improvements in detector resolution and speed have resulted in unprecedented data rates at national light- and neutron-source facilities. Beamlines generate terabytes of raw and derived data each day and serve thousands of researchers each year. Similarly, new technologies are enabling the detection, transmission, and storage of astrophysics data; consisting of electromagnetic, gravity, and particle spectra, at scales never seen before. Currently four large-scale photometric and spectroscopic sky surveys are under way, each generating and/or utilizing hundreds of terabytes of data per year.

These instruments increasingly rely on HPC centers such as the Argonne Leadership Computing Facility (ALCF), Oak Ridge Leadership Computing Facility, and the National Energy Research Scientific Computing Center (NERSC) for their computational and analytic requirements. For example, both Advanced Light Source (ALS) and Palomar Transient Factory (PTF) data are processed at NERSC. In the PTF, data taken with the camera are transferred to automated pipelines at NERSC using the Energy Sciences Network. Figure 2 shows an example of imaging workflows for materials science where beam lines at Argonne's Advanced Photon Source (APS) rely on HPC capability at the ALCF.

Many of the instrument-based workflows have been developed for specific use cases. The workflow system provides data access, management, and analysis and integration with simulation codes, and often has web interfaces for publishing data. Workflow systems drive the automated, near real-time processing, and user-triggered actions. Typically, they also provide extensive monitoring of distributed workflows for system operators, resource providers, and end users.

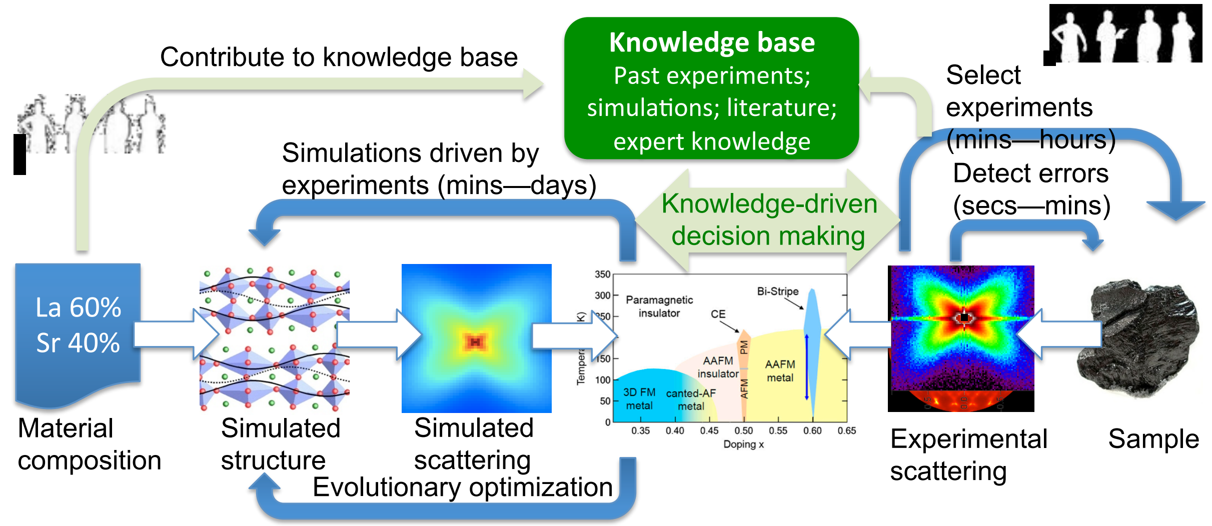


Figure 2: Materials structures workflows can include both instruments and computations.

Challenges:

Instrument workflows need to seamlessly incorporate the scientific instrument, the network, storage, and compute resources to provide scientists with real-time access to data and processing. However, a number of challenges arise in realizing this vision. For example, collecting provenance of all the artifacts of the workflow is difficlt. Data analysis should match the rates at which data are generated. Scientists should be able to inspect results on the fly and make near-real-time decisions in order to modify the analysis or control the instrument. However, HPC systems have limited or no support for real-time processing. For example, although real-time feedback during beam time is critical to light source users, it is difficult to obtain, resulting in precious time lost at the beamline.

R&D Needed:

Several research areas need in-depth investigation to address the challenges of experimental workflows.

1. End-to-end workflow lifecycle. The scientific workflow needs to seamlessly incorporate the instrument, the HPC center, and the network in order to provide convenient and efficient access to scientists. It will be necessary to investigate how the end-to-end workflow that encompasses the traditional computational workflow is represented and executed. Critical points in the infrastructure will need to be identified for provenance collection, verification, and validation of the workflow execution. Additional research is needed to develop capabilities that will allow users to seamlessly transition between development and production workflows, and to share workflows among researchers and students.

2. Real-time resource scheduling. It is important to investigate how real-time resources can be scheduled in order to allow users to interact dynamically and adaptively with their instruments based on the results of their computations. HPC resource management systems and schedulers will need to accommodate interactive and real-time workloads in order to satisfy the needs of experimental user facilities.

3. Execution modalities. Instrument workflows often harness resources at one or more computational facilities. It will be necessary to define the interface between DAC and HPC workflows: where will the workflow be managed and at what time scales? The coupled experiment-computation system provides a rich source of research opportunities in organizing, moving, analyzing, sharing, and tracking large quantities of data. Additional research is needed to allow for data-driven processing, the ability to track and search data products, and integration of experimental data with other knowledge sources such as simulations.

3.1.3 Collaborations

State of the Art:
Workflows enable scientific collaborations. Scientific collaborations such as the Large Hadron Collider and the International Thermonuclear Experimental Reactor involve scientists with different roles. Often, one team of scientists is present at the instrument site to monitor the progress of the ongoing data collection and adjust the control settings, while other scientists analyze the data generated, and still others use the output of the analysis to draw conclusions and publish findings.

Projects such as Integrated Microbial Genomes and the Metagenomics RAST server (MG-RAST) provide access to the analysis and simulation results through web interfaces to databases that are viewed and used by thousands of scientists. Projects such as the Systems Biology Knowledgebase (KBase), Accelerated Climate Modeling for Energy (ACME), and Palomar Transient Factory (PTF) provide an analysis workflow environment where users can run their own workflows on the data products generated from other workflows. For example, Figure 3 shows an example of different science teams and facilities connected with the PTF project.

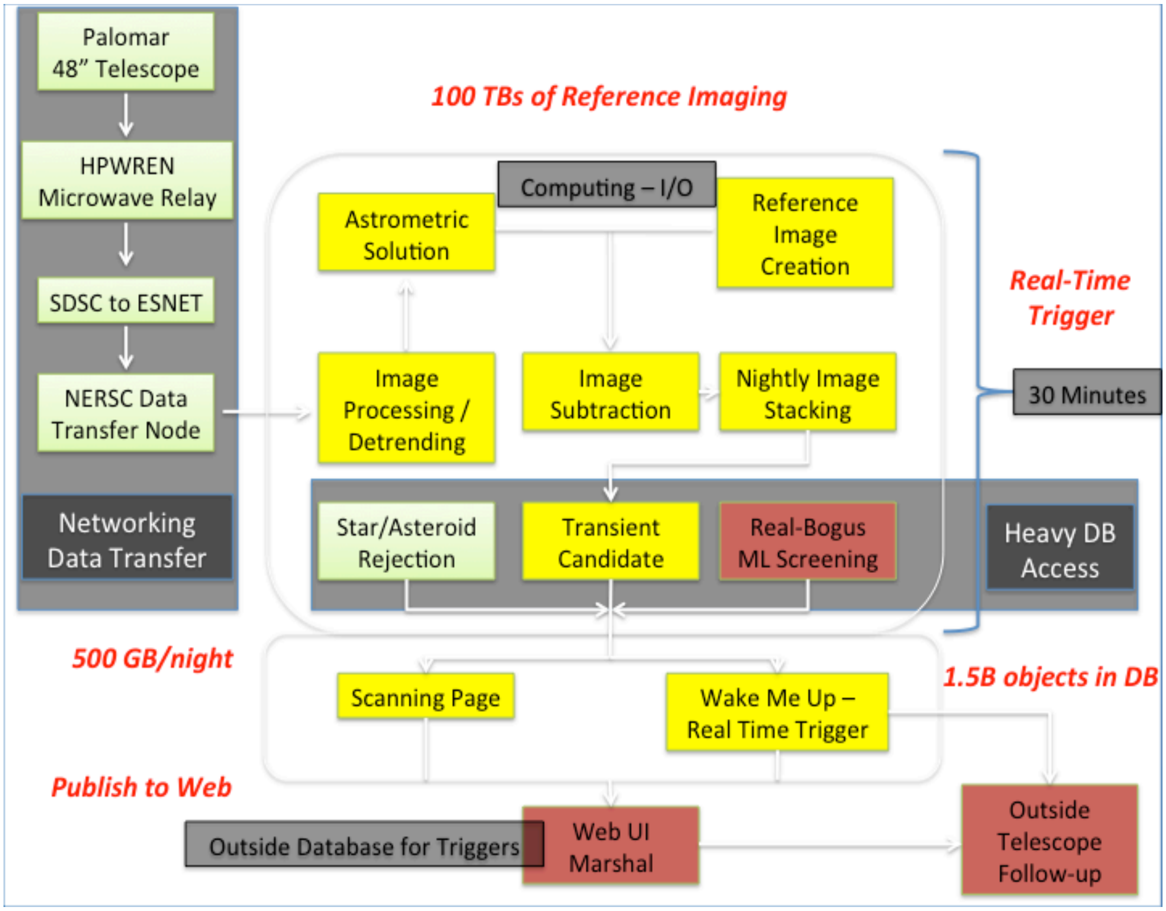


Figure 3: Approximately 100 GB of raw optical imaging data is taken every night at Palomar Observatory. A database of nearly 1 TB containing over 1.5 billion objects is scoured to compare this event with previous detections at that location on the sky. The results are published on the web in 40 minutes after an individual image is taken at Palomar.

Challenges:

Scientific collaborations still face numerous challenges in the context of developing, verifying, sharing, and managing workflows. It is difficult for users to seamlessly transition between environments and share their workflows, computation, and data resources; for example, scientists may want to compose workflows from a pool of existing components and rerun parts of a workflow with different input data.

R&D Needed:

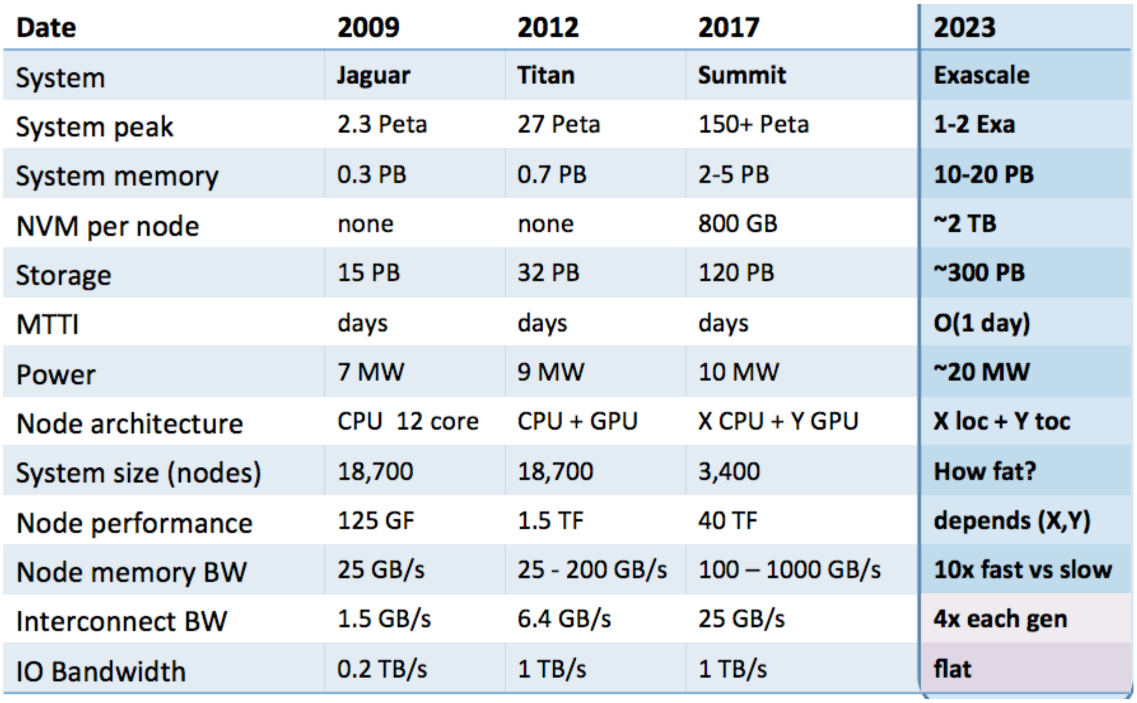
1. User roles. Data producers, data analysts, and data consumers each have a unique set of requirements in the global workflow. Scientific collaborations need a way to capture these relationships among the users and the workflows. Additional research is needed to identify these roles, examine the interplay between them, and define adequate representations of the people, resources, and data in the global workflow.

2. Shared resource environments. Existing workflow tools rely on web servers, data publication tools, and database servers that are not part of the traditional HPC environment. Scientific collaborations often struggle with providing an infrastructure that can be shared across geographically separated sites. New container and virtualization technologies (e.g., Docker) attempt to bridge this gap. However, more research is needed to create portable, provisionable, and shared resource environments for scientific collaborations.

3. Shared workflows. Research is needed to understand how workflows are shared in a collaboration, how changes to the workflow are captured, and how provenance metadata are made available to users. Provenance is the key to allowing others to reuse and/or repurpose data that were generated by someone else, for example, students in an undergraduate class. Provenance can enable students to use existing analytic workflows and study how to construct new workflows. (Section 4.5 discusses training the next generation of workflow scientists.)

3.2 Computing Systems

Future computing systems will be very different from today's infrastructures, and so will scientists' use of them. Billion-way concurrency, deep memory hierarchy, hardware heterogeneity, reduced memory per core, and limited network and I/O bandwidth will be the norm (see Table 1).[3]



Date	2009	2012	2017	2023
System	Jaguar	Titan	Summit	Exascale
System peak	2.3 Peta	27 Peta	150+ Peta	1-2 Exa
System memory	0.3 PB	0.7 PB	2-5 PB	10-20 PB
NVM per node	none	none	800 GB	~2 TB
Storage	15 PB	32 PB	120 PB	~300 PB
MTTI	days	days	days	O(1 day)
Power	7 MW	9 MW	10 MW	~20 MW
Node architecture	CPU 12 core	CPU + GPU	X CPU + Y GPU	X loc + Y toc
System size (nodes)	18,700	18,700	3,400	How fat?
Node performance	125 GF	1.5 TF	40 TF	depends (X,Y)
Node memory BW	25 GB/s	25 - 200 GB/s	100 – 1000 GB/s	10x fast vs slow
Interconnect BW	1.5 GB/s	6.4 GB/s	25 GB/s	4x each gen
IO Bandwidth	0.2 TB/s	1 TB/s	1 TB/s	flat

Table 1: Future HPC architectures.

[3] http://extremescaleresearch.labworks.org/events/workshop-future-scientific-workflows
Lucy Nowell presentation, data courtesy Al Geist and Susan Coughlin.

Software systems will feature hierarchical organization of the global operating system in local units called enclaves that will be tailored to the hardware requirements of a subset of nodes (e.g., fat nodes, thin nodes, GPUs, FPGAs, burst buffer nodes). Programming models are also evolving from bulk-synchronous to task-based approaches.

Workflow systems will need to evolve in light of these changes. In situ data reduction and analysis will be essential in order to mitigate the I/O and network bottlenecks. Interactive guiding of in situ analytics and simulations is an outgrowth of the availability of real-time results. The provenance of the entire HPC + DAC workflow (the in situ and post hoc operations) will need to be captured so that it can be used to share, validate, and reproduce published results.

Some of the hardware and software changes are welcome additions from the standpoint of workflow execution. The 2015 “Storage Systems and Input/Output to Support Extreme Scale Science” (SSIO) report [25] says, “The ongoing integration of SSD devices into compute infrastructures, both as burst buffers and as extended memory, offers opportunities to enhance science workflow productivity.” Likewise, there is a natural mapping of enclaves in the OS/R to the management of tasks in workflows. However, innovations such as NVRAM and software enclaves need to have appropriate interfaces in order for them to be useful for workflows. The following sections outline the state of the art, research challenges, and R&D needed in hardware and software systems in support of workflows.

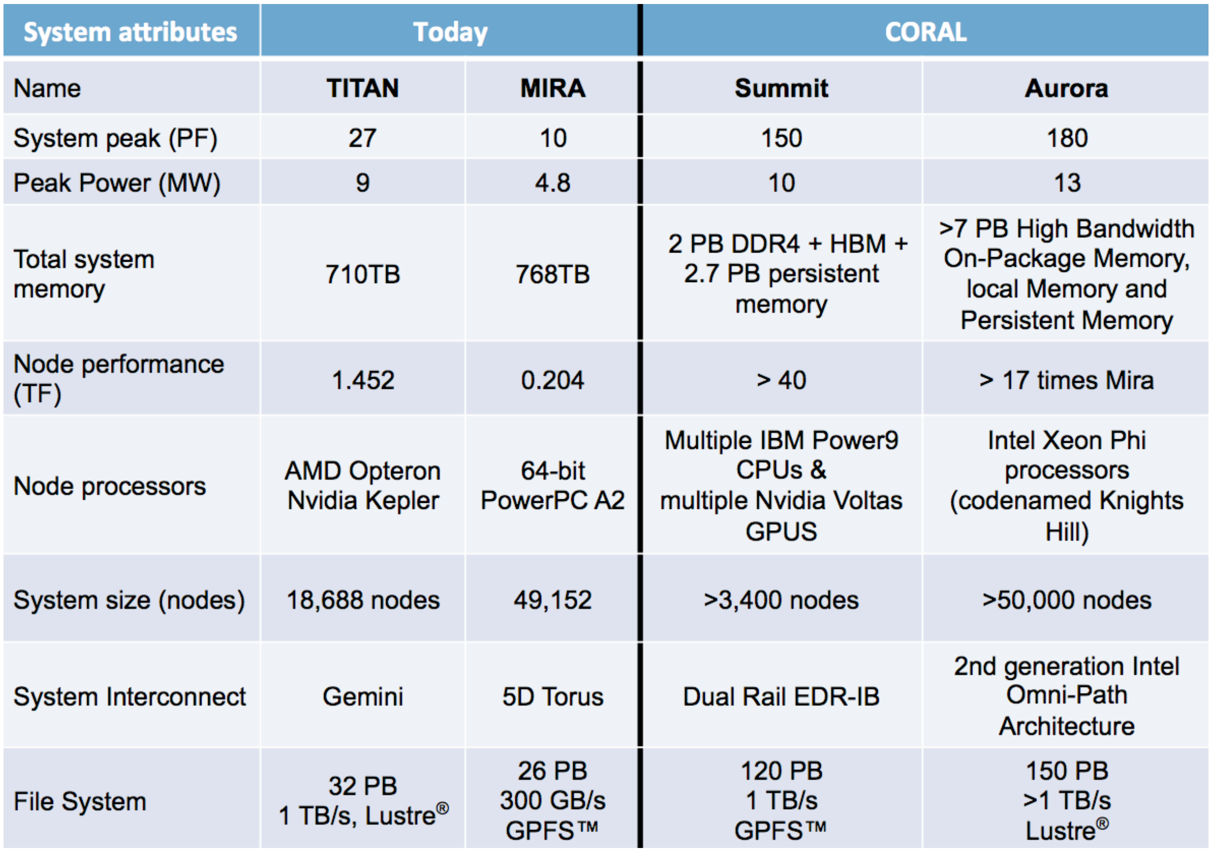
3.2.1 Hardware Systems

Technology trends are being driven by a belief that power limits system size and performance. Workshop discussions, however, indicate that scientists value productivity above all. Furthermore, systems must be designed to account for science workflows, not only running one benchmark on one platform.

State of the Art:

The scientific computing community is facing major challenges in the next decade: power, performance, resilience, and productivity. Although these challenges have been with us for some time, they are growing more acute as facilities and scientists are confronted with a new level of complexity and required investment, derived from the complex new architectures with a multitude of features and dynamics. One need look no further than the contemporary extreme scale systems being deployed and procured today, as illustrated in Table 2.[4] Titan, MIRA, Aurora, and Summit have entirely new features and configurations that have not be present in earlier systems. These changes require significant planning and investment in system software, programming environments, applications, and workflow management systems.

[4] http://extremescaleresearch.labworks.org/events/workshop-future-scientific-workflows
Lucy Nowell presentation, data courtesy Al Geist and Susan Coughlin.



System attributes	Today		CORAL	
Name	**TITAN**	**MIRA**	**Summit**	**Aurora**
System peak (PF)	27	10	150	180
Peak Power (MW)	9	4.8	10	13
Total system memory	710TB	768TB	2 PB DDR4 + HBM + 2.7 PB persistent memory	>7 PB High Bandwidth On-Package Memory, local Memory and Persistent Memory
Node performance (TF)	1.452	0.204	> 40	> 17 times Mira
Node processors	AMD Opteron Nvidia Kepler	64-bit PowerPC A2	Multiple IBM Power9 CPUs & multiple Nvidia Voltas GPUS	Intel Xeon Phi processors (codenamed Knights Hill)
System size (nodes)	18,688 nodes	49,152	>3,400 nodes	>50,000 nodes
System Interconnect	Gemini	5D Torus	Dual Rail EDR-IB	2nd generation Intel Omni-Path Architecture
File System	32 PB 1 TB/s, Lustre®	26 PB 300 GB/s GPFS™	120 PB 1 TB/s GPFS™	150 PB >1 TB/s Lustre®

Table 2: DOE Systems expected through 2022.

In particular, several trends are already emerging: heterogeneous computing, new memory systems including nonvolatile memory, and small or no growth in bandwidth to external storage systems or networks.

Heterogeneous computing is apparent in many of today’s top HPC systems. In earlier systems, such as Titan and Tsubame2, the addition of GPUs to systems gave performance improvements while keeping power constraints satisfied. As these capabilities evolve, we see tighter integration of heterogeneous and special purpose capabilities onto general processors. For example, over the past several years, Intel has integrated GPUs, compression and encryption engines, random number generators, and other capabilities directly onto their main processors. Although this functionality may be exposed to users in a number of ways, it will be imperative to provide portable solutions to HPC users. As seen in Table 2, both Titan and Summit will have heterogeneous instruction set architectures within the node that scientists will need to carefully program and orchestrate. Meanwhile, the same applications will be expected to port and run efficiently on many other platforms such as Cori and departmental x86 clusters.

Nonvolatile memory (NVM) systems, in addition to alternative memory architectures, are emerging as a solution to the limits of dynamic random-access memory (DRAM) scaling, power, and cost. Depending on the architectural solution, this change to the memory system could be more disruptive to applications teams than the change to heterogeneous computing. NVM devices have major differences from DRAM: lower write durability; higher latencies and power costs for writes relative to reads; and persistent state without the need for standby power. Again, as with heterogeneous systems, programming systems, and system software, workflow management systems will need to hide these often subtle differences from applications while taking advantage of the particular capabilities. Although NVM devices have been transparently introduced into existing systems as replacements for hard-disk drives, they typically use existing I/O block-oriented interfaces, although the software stacks have been optimized. In the Summit and Aurora configurations, we must prepare applications for potentially tighter integration of these NVM devices with main memory and processors, bypassing the I/O interface. In both of these cases, a number of open research questions about high-level system design remain. These questions include the amount of NVM versus DRAM memory, the number of latency-tolerant cores versus the number of throughput cores, and the extent to which application data structures are a good fit for the characteristics of NVM when compared with DRAM?

As noted in Table 2, storage systems and connections to external networks will continue to see the aggregate storage bandwidth plateau or increase slowly. This trend will force users to consider other strategies for defensive checkpointing of application state (e.g., burst buffers), and postprocessing and analysis of application output (e.g., in situ analysis). While these changes could force major changes in application design and architectures (e.g., increasing the amount of NVM for in situ analysis of time-series data), workflow management systems must be improved in order to recognize these dramatic differences and integrated in order to enable fluid migration of workflows and data across geographically distributed, dynamically changing resources and within HPC systems.

Challenges:

- The technology trends can still be influenced; therefore application and system people must work together to determine future architectures.
- Systems are likely to be more heterogeneous and diverse.
- Managing flow is a complex problem in both HPC and DAC WMSs.
- Systems are likely to have increasing scratchpad and NVM.
- There is a lack of workflow codesign workloads.

R&D Needed:

1. Metrics. There is a need to define metrics for valuing performance, power and productivity. How do we measure productivity?

2. Scheduling. Research is needed to develop efficient, low-overhead, and robust task scheduling on heterogeneous systems and across systems.

3. Data movement. New mechanisms need to be developed for passing data between tasks without unnecessary data movement.

4. Memory management. Research is needed in memory management systems that support for scratchpad and NVM.

5. Mini workflows. There is a need for a library of proxy or mini applications and benchmarks to test workflow systems.

3.2.2 Software Systems

Today's HPC software systems are designed to support the execution of single jobs that are batch-scheduled. This mode of operation contrasts with workflows consisting of multiple, coordinated, individual tasks. DAC software systems are perhaps better equipped to manage workflows, but their loosely coupled decentralized design (e.g., communicating through files) limits performance and scalability. One outcome of the workshop is that system software for workflows at extreme scale is a severe challenge for both HPC and DAC.

State of the Art:
From single-user batch jobs to workflows. Today's large machines [2], [3] are generally engineered to assume that the client is an interactive user that connects to the head node in order to submit jobs, transfer data, and perform other tasks at the command line. This approach translates into security requirements constructed around interactive users, such as the requirement for short-lived Kerberos passwords, and S/Key authentication. Unfortunately, this assumption often conflicts with the design of long-running workflows that require submitting jobs, transferring files, and taking other actions on behalf of the users while they are not present. A common occurrence is that a user will log into a machine manually to start a workflow, then log out, only to discover hours later that the workflow has failed because the user's time-limited credential has expired [4].

The solution to this involves both policy and technology. First, security policy must be developed with the understanding that workflows are an important use-case for these machines, and hence appropriate login mechanisms and long-lived credentials should be supported where appropriate. Second, workflow technology must be developed to explicitly support security constraints: when necessary, the WMS can notify a human to approve an action or renew credentials or to halt activities in an orderly manner.

Reasoning about robustness in the distributed setting. In current systems, there is little guidance or understanding how to deal with unexpected situations across software layers. For example, there are approaches to deal with individual component failures [5], [6] as well as generalized algorithms for fault detection and recovery [7]–[9]. However, simple recovery techniques do not scale up: components that always retry failures will accidentally cause denial- of-service attacks, while components that always return failures to the user will be perceived as unreliable. A one-hour system outage might be reasonably considered a fatal event for a workload that is expected to take five minutes but would be considered a mere hiccup for a workload that is expected to run for days. A methodology for understanding failures and proportionate response within the context of the larger system should be developed and widely applied across all relevant components.

Storage management. In current systems, storage is the least-effectively managed resource at runtime. While there do exist global user quotas on many filesystems and systemwide monitoring services such as Darshan [10], [11], no mechanism is available to allocate storage capacity or bandwidth for a particular task, much less a workflow of multiple tasks. As a result, it is much too easy for a process to overflow individual disk or network capacity with temporary data or for a workflow manager to overflow a shared storage space by running too many tasks at once. The inability of supporting storage allocations also often results in unnecessary failures.

Many use cases for runtime management of intermediate storage were discussed at the workshop, including those proposed by [12]–[22]. For example: applications may attempt to access remote data via the network and cache it locally during a run. This capability requires allocation of local storage as well as careful garbage collection upon completion. Applications may "park" data for medium time scales (days to weeks) in order to pass data from one task to the next, while other users access computational resources. This requires explicit allocation of storage for a given time period and arbitration of storage consumption between different users. Data might also be migrated up and down the storage hierarchy as needs dictate. It is becoming more common to couple different resources together at runtime. A complete application might consist of an HPC message-passing task, a high-throughput task, and a commodity database, all of which must be coallocated and linked via storage.

Challenges:

Future DOE systems impacts on workflow design. Because the machine itself will be (relatively) CPU-rich and I/O-poor, the metrics of success will focus on I/O performance, which will drive the design and evaluation of WMSs that are currently oriented around CPU performance. multiple workflow implementations are likely to remain, particularly distinguishing between DAC and HPC/in situ workflows, but there was a strong desire for common representation languages for workflows, as well as the ability to compose heterogeneous workflow implementations.

Data management. Because future machines will be data-constrained, they must treat storage, I/O, and network capacity as first-class resources that must be allocated, managed, and measured to the same degree as computing capacity is today. Execution of data-intensive tasks with acceptable performance will require advanced data-sharing mechanisms within the machine itself, without using the global file system. This involves solving problems of both coordination (naming and rendezvous) and resource management (allocation, enforcement, garbage collection) so that intermediate storage resources are not exhausted. To the extent that unexpected events happen within a machine, it must be possible to either over-provision or reprovision storage resources as necessary. For example, if two processes are to be connected as producer and consumer using intermediate storage, a delay in starting up the consumer process entails either increasing the intermediate storage allocation, or delaying writes by the producer. The multiplicity of memory and storage technologies may result in a variety of communication and storage mechanisms, which requires that users be given information and tools to select the appropriate mechanism.

Planning, provisioning, and scheduling. Experience in the distributed computing realm suggests that constructing one large metascheduler through hierarchical queues is not effective, because the interaction of scheduling policies does not serve any party well. Rather, the approach proven effective is to submit "pilot" jobs to a local scheduler in order to claim resources. Once the pilots are scheduled, they connect to a user-specific scheduler that assigns tasks. In this way, the site retains control over allocating resources to users, while the user retains control over the ordering of tasks.

Systems software stack. WMSs need the OS/R to move from its traditional role of managing single tasks to managing global (i.e., internode, distributed) services. In OS/R research projects such as Argo [23] and Hobbes [24], the global management is through a hierarchy of resource groupings (i.e., enclaves) with some global resource management and scheduling, and various programming models and runtimes managing resources within a given enclave or task.

Similar to the runtime for a single task, the WMS can be seen as the runtime orchestrating execution of the tasks of the workflow, where each task uses its own software infrastructure (programming model, runtime, OS) from the global software stack. However, to date there is no clear delineation of who owns scheduling on these systems. The WMS could be responsible for systemwide scheduling, perhaps replacing a traditional job scheduler. However, it is not clear that current WMS implementations have any of the standard cost models and priority capabilities typical of a batch scheduler.

Alternatively, rather than the WMS taking over the role of a runtime system, the WMS can negotiate with the OS/R on behalf of the entire application workflow. The WMS would be a tool to enable the scientist to describe the coordination that is needed among various components (such as the simulation and data analysis codes) and would capture the types of provenance that scientists need to support writing papers and validating scientific results. In particular, the WMS needs to capture any changes from the initial workflow that result from human-in-the-loop interactive analysis and steering. Such capability is not handled in the programming model or OS/R. The WMS can also provide portability across different hardware platforms.

Performance and predictability. Naturally, end users want predictability of performance, such that similar workflows achieve similar performance when repeated. However, predictability is hard to achieve in the presence of contention for resources: for example, an overcommitment of the network may have cascading effects on the performance of all other components. Effective resource allocation or adaptivity to conditions may be effective at improving performance predictability. To understand such interdependent effects in a complex system requires a good set of modeling and simulation tools to represent and evaluate workflows.

Reproducibility, portability, and integrity. Reproducibility of scientific results is an ongoing challenge. Workflow systems are in a privileged position to assist with this problem, because they sit in a place where the structure and dependencies of an application are either explicitly stated or can be observed through measurement. The workflow community has considerable experience in recording and evaluating provenance data generated by workflow systems. To proceed in this direction, we will need system components such as batch schedulers and file systems in order to be sufficiently transparent that provenance data can be extracted at runtime. A significant challenge will be data management—for machines with high component counts, the recording of provenance data may present problems of performance and capacity in and of itself. This topic generated considerable discussion at the workshop. It was generally agreed that reproducibility was a significant challenge; however, it was not clear whether users would be willing to accept the costs of tracking such data. In order to be accepted by users, provenance tools must deliver value commensurate with the costs that they impose.

Other related challenges. Other open questions include the following.

- In what ways does a WMS differ from the runtimes needed for other programming models?
- What external dependencies/requirements does a WMS have that single-task programming models/runtimes do not?
- What gaps need to be filled by vendors (e.g., communication between enclaves)?
- What WMS benchmarks are needed to drive vendors to meet WMS performance metrics during system acquisition?
- What relationship does WMS scheduling have to the "traditional" scheduler in these systems?
- What are the right metrics, and how do we develop the right cost models, such that a WMS can make good use of system resources?
- What is the interface API that the OS/R needs to provide to the WMS?

R&D Needed:

1. Collaborative WMS and OS/R provisioning, planning, and scheduling. An open research and engineering challenge here is the intersection of provisioning, planning, and scheduling across the OS and WMS software stacks. Currently, these tasks are performed independently: The user provisions appropriate resources, a planning tool constructs a suitable plan, and then a workflow manager executes it with some handoff to the OS/R but without feedback on progress unless provided by the application. Clearly the tasks are connected: the appropriate quantity of resources to provision depends on the structure and performance of the workflow. There is a research question to understand the interconnections, and an engineering/design question to construct tools that can perform all three without overwhelming complexity. We need to examine the interplay of workflow management, resource provisioning, and OS/runtime systems.

2. Data management. Research and development is needed to address integration of in-system storage and campaign storage with traditional parallel file systems and archive as well as metadata, name spaces, and data provenance in the context of workflow management systems. This research direction was echoed in the 2015 SSIO report [25].

3. Co-Design "at scale" modeling simulation tools. Because of the stringent up-time demands placed on current leadership-class supercomputer systems, it is infeasible for them to serve as WMS testbeds, especially where the OS/R and/or core system schedulers might need to be modified. Yet, how a WMS operates can impact a whole leadership-class supercomputer systems. Thus, research and development is needed in models and simulation tools that an allow research investigations on whole systems to identify cause-and-effect relationships between the WMS and various software and hardware components.

The community needs good benchmarks, conceptual models of applications, miniapps that stand-in for real applications, system simulations, performance archives, and similar tools that enable the performance evaluation of complex system designs.

4. Extreme-Scale Workflow Systems

Summary

In system design and execution, the efficient management of DAC and HPC workflows present challenges at multiple levels of WMS software. At the base layer, coupling control and data flow between heterogeneous components requires expanding workflow links into dataflows that can buffer, prefetch, aggregate, and distribute data. Data models need to be defined in a uniform way so they can be managed in the data flow. Moving to a higher level of abstraction, managing the workflow must address concurrency, locality, and system topology if data movement is to be optimized at extreme scale. WMSs must be designed with monitoring mechanisms in place so that provenance can be captured and analyzed. Faults (both hard and soft errors) will be more prevalent and must be mitigated.

In programming and usability, a lack of standardization between numerous programming models for both tasks and workflows of tasks, the interconnection between the programming of individual tasks and entire workflows, and the portability of both code and data across different locations in a potentially heterogeneous workflow that spans both HPC and DAC resources were identified as challenges. The scale and complexity of components in extreme scale workflows complicate provenance capture; no uniform format exists to trace, validate, and reproduce both HPC and DAC operations. The velocity of provenance data generated at extreme scale is another bottleneck—to the extent that provenance data can easily outpace scientific data—requiring new methods to subset, compress, mine, analyze, store, and share it.

The reason for capturing provenance data is to be able to predict performance and to validate and reproduce the science results. The increasing scale and complexity of hardware and software systems, however, coupled with the composition of multiple tasks by workflows, complicate those validation steps. Validation of data accuracy is no longer as simple as bitwise difference, and validation of expected performance is a complex system of a high-dimensional space of metrics over a heterogeneous computing architecture. Important research directions include extending single-application performance validation tools to workflows of applications and developing methods to learn what level of differences in science results are statistically significant. All of the above can be considered part of a new field—workflow science—that studies the formal theory and design principles of workflow systems, develops analytical models for their performance and validation, and empirically measures the predicted behavior through experimentation.

Findings

- System design and execution
  - Coupling control and data flow between heterogeneous components requires research in dataflows that can buffer, prefetch, aggregate, and distribute data.
  - Data management research must solve challenges in the transport, layout, attributes, and provenance of data

    - The convergence of DAC and HPC workflows into a single, possibly hierarchical, WMS presents challenges in managing nonuinform latency, task granularity, and reliability.
    - System introspection is needed to understand complex system behaviors.
- Programming and usability
    - The diverse needs of workflows can impact usability. For example, real-time workflows need immediate cognition. In addition, there are many different ways to express workflows that span the range from simple to complex. Research to identify common needs and expression patterns (akin to design patterns in software engineering) in workflows with respect to a number of properties including data management, error control, reproducibility, programmability, and mapping to physical resources are needed.
    - Research is needed to determine appropriate levels of abstraction to define the user interface for workflow systems and their component modules, including an interface for the human in the loop in interactive workflows.
    - Many commonalities exist between HPC and DAC workflows. Although there exist several examples of workflow applications that work across HPC and DAC boundaries, most are specific to a particular computing environment. Developing WMS and application components that operate with good performance across heterogeneous platforms is needed. Understanding the role of containers, virtualization, and security—features found in DAC—is needed in HPC. Understanding the effect of disruptive HPC architectures such as deep memory hierarchies and NVRAM is needed as well.
- Provenance capture
    - Research is needed into new provenance models suitable to support new usage models. Moving from one provenance model with one stream of provenance to many streams that align and intersect at certain time intervals requires new research.
    - The capture and utilization of provenance information across system software levels and distributed systems will require investigations at many different levels. Needed are effective capture and communication mechanisms and effective selection, storage, and delivery approaches for provenance objects that support fast runtime storage, search and retrieval, in situ triage, and analysis.
    - Mining of provenance data has emerged as a key approach for its analysis, for both relational and graph databases.
- Validation
    - The increasing complexity of workflows and their computational environments makes it critical to provide the science community with the approaches, methods and tools to ensure workflow execution correctness and to validate that workflows are executed with sufficient reproducible accuracy and performance to meet their scientific goals.
    - Exascale systems with their higher degrees of concurrency will further accentuate this community need.

    - Validation is expanding from a post hoc to a runtime function that can enable mitigation and optimization.
- Workflow science
    - Research is required to develop the science of workflows to fully understand how workflows operate. Did the workflow operate as expected? Did the infrastructure (computer, instrument, network, storage) operate as expected? Can the data or metadata be trusted? Is the experiment repeatable?
    - One role of workflow science is to develop and execute theoretical models that describe the expected behavior of workflows.
    - Workflows can be modeled computationally. The community should strive to design and execute computational simulations and models that describe the workflow behavior.
    - More research is needed to design and execute experiments to validate the function of scientific workflows and the facility resources (computers, storage devices, networks, instruments) used by these workflows.

4.1 System Design and Execution

The design of WMS for extreme-scale DAC and HPC workflows presents multiple challenges; some such challenges are unique to each class of workflows, while others are common across different classes. Furthermore, the increasing exploration of end-to-end workflows that combine both DAC and HPC aspects means that solutions for efficient management of these workflows must be able to seamlessly support both classes. As a result, their development results in a challenging research agenda, which is discussed below.

4.1.1 Control and Data Flow

State of the Art:
Managing control and data flow during workflow execution is critical to workflow's enactment, and the design of solutions largely depends on both the nature of the workflows (e.g., coordination/coupling requirements, data volumes) and the execution environment. While DAC solutions have addressed the latencies, unreliability, and heterogeneity of distributed systems, HPC solutions have focused on optimization for performance and scalability. HPC workflows have an additional focus on the coupling of workflow components via different data sharing mechanisms. For example, DAC solutions have largely relied on files for control and data flow [1], [26], while HPC solutions have explored in-memory staging and RDMA [27], [28].

The nature of the programming and execution environments has also impacted the abstractions presented for specifying these workflows. For example, while DAC WMS support more dynamic and service-oriented compositions, HPC WMS typically tend to be more static and closely coupled in their specifications. HPC workflows also tend to make stronger assumptions about the compatibility of workflow components compared with DAC workflows, that is, whether components were designed to work together and how much they share with each other (e.g., data formats). Recent research in HPC is, however, exploring how service-oriented architecture (SOA)-type compositions can be supported on extreme-scale systems [12].

Another important issue is the “links” that connect the component applications of the workflow and the semantics associated with these links. For example, link components can be used for buffering, prefetching, and aggregating data but may also be used for transforming and redistributing data so that it is more compatible with the needs of the consumer. Staging-based approaches for specifying these links and their semantics for HPC workflows have been explored by recent projects [29], [30]. Lofstead [31] has also proposed the use of “glue components” that encapsulate such link semantics and can be defined as part of a workflow.

A related issue is allocating and managing resources to achieve these link semantics. Because the resources requirements can be dynamic and may not be known a priori (e.g., they may depend on the volume/type of data, the type of transformation, the network state), adaptive runtime resource allocation and management become critical. Furthermore, this resource management must be able to effectively address and respond to technical issues, such as failures. Efforts such as [32] and [33] have explored such adaptive runtime management schemes for DAC and HPC workflows, respectively.

Challenges:
As DAC and HPC workflows are integrated into end-to-end application workflows, providing abstractions and mechanisms that can effectively support control and data flow requirements in a consistent and scalable manner becomes a critical challenge. For example, moving across the HPC--DAC interface entails not only moving data products, but also control information about the state of the workflow. Similarly, other important concerns include providing support for specifying link components with meaningful semantics for both DAC and HPC workflows, as well as effectively managing these components in an adaptive and autonomic runtime.

R&D Needed:
1. *Abstractions*. Abstractions and mechanisms for control and data flow that can effectively support integrated DAC and HPC workflows in a seamless and scalable manner.
2. *Primitives.* A catalog of link component semantics that can meet the needs of emerging application workflows, as well as the efficient implementation of such components that can be used as part of DAC and HPC workflows.
3. *Runtimes*. Adaptive/autonomic runtimes for managing the behavior of link components.

4.1.2 Data Management

State of the Art:
Emerging workflows incorporating both DAC and HPC aspects are generating unprecedented amounts of data, some of which is required by certain stages but not necessary for the end result (i.e., intermediate data). Further, such data are being created and consumed in new and intricate ways because of the complexity of such workflows. As a result, fast and efficient

management of these data is a critical concern of next-generation workflow management systems, in order to translate data into scientific insight in a timely manner.

Recent research in data management for DAC and HPC workflows has addressed issues such as data transport (e.g., GridFTP [34], DART [35]), data distribution and layout (e.g., D2Worm [36]), data staging (e.g., DataSpaces [28], DataStager [12]), data replication (e.g., adaptive data placement Pd-Loc [37]), data models, and metadata management including logs and provenance. Each of these topics presents its own challenges and research issues, which are outlined as follows.

Challenges:

Data transport. Research challenges include providing more stringent quality-of-service (QoS) guarantees for end-to-end workflows, including DAC and HPC, as well as developing abstractions for applications to express QoS requirements. Furthermore, application-driven mechanisms, aware of relative importance of data, can utilize data various transport QoS levels in order to optimize performance. Visibility of expected performance and QoS (e.g., using models) is important. Addressing QoS issues across multiple concurrent transfers with different requirements is also important.

Data layout. Executing end-to-end data exchanges between the component applications in the workflow requires understanding and representing their associated data models and data distributions.

Data attributes. Schemas to represent attributes such as data lifetime, reliability, and security, can serve as additional elements of user/application intent. These aspects may be used to optimize data placement or manage resources. These attributes can also be used as a basis for developing cost models that can be used to evaluate and prioritize associated tradeoffs.

Provenance. Interfaces and mechanisms must support capturing, curating, querying, and managing possibly distributed provenance information (system and application) so that it can be effectively used for audit, replay/regeneration, reproducibility, and publication.

Data-driven diagnosis. Capturing, managing, and linking job logs, RAS data, and performance data on different systems can support data-driven problem diagnosis and forensics. These data may be distributed.

R&D Needed:

1. Quality of service. Develop mechanisms to formalize and standardize better communication of QoS, throughput, latency, deadline, and priority are required. We need to develop methods for associating properties, such as lifetime, with data. Also needed is a data-centric view in which data are not just tokens in a graph but first-class objects.

2. Data layout. Investigate ways to represent, reason over, and manipulate the layouts of data structures to optimize performance for different settings.

3. Data semantics. Investigate ways to represent, reason over, and manipulate the semantics of data structures to optimize performance for different settings
4. Data models. Conduct research on the intended purposes and the type of data needed for those purposes.
5. Data provenance. Policies and methods for linking logs of different types and resources in a way that preserves privacy and permits distributed querying.

4.1.3 Workflow Management

State of the Art:
Even though workflow management systems for DAC and HPC workflows are conceptually similar, they address different design points. While DAC WMSs were traditionally motivated by automation and productivity perspectives, HPC workflows are driven by I/O costs and performance requirements.

The two WMSs also typically implement different design tradeoffs. For example, the latencies involved can be very different. Similarly, mapping and scheduling solutions for DAC and HPC represent different design points and optimizations. While concurrency, locality, system topology awareness, and minimizing data movement are key concerns for HPC and DAC workflows [38], [39], DAC workflows additionally address issues related to security, crossing administrative domains, and so foth that do not exist in HPC. Monitoring the execution progress of HPC workflows and adapting the execution also tends to be harder, largely because of restrictions from the system.

A related concern is dealing with failure. Although being able to detect and handle unreliable resources and failures—has been an integral part of DAC WMSs, only recently have HPC WMSs started to address failures [40], [41].

Challenges:
Emerging extreme-scale systems are expected to exhibit many of the characteristics of distributed systems (e.g., heterogeneity, failures, nonuniform access latencies). Similarly, the applications in Section 3.1 demonstrate that workflows are expected to combine aspects of both DAC and HPC. As a result, exploring crosscutting solutions that can integrate these design points is a critical research challenge. This includes support for workflow specification and enactments. For example, workflows may be thought of as composed, possibly hierarchically, of well-defined DAC and HPC pieces that must be executed as part of a single coordinated workflow. Several challenging implications arise. For example, how can a single WMS support both design points effectively, or how can failures be handled in a consistent manner across the execution modes? Furthermore, DAC and HPC tend to work at different time scales and task granularity, so multilevel, multiscale decision-making is needed.

DAC and HPC may have different optimization considerations. For example, analytics may be scheduled in situ or on a different system depending on the relative costs of the latencies associated with data movement and the loss in performance due to the repurposing of some compute nodes for analytics. Several research efforts are already exploring such a convergence. For example, Wide-Area-Staging [42] is extending data staging abstractions to distributed environments, and DataSpaces-as-a-Service is exploring persistent staging and SOA models on extreme-scale systems.

R&D Needed:
1. *Design patterns*. Understand patterns for integrating DAC and HPC workflows, and define execution semantics for integrated DAC plus HPC workflows.
2. *Resilience.* Define consistent semantics for handling failures and recovering from faults for integrated DAC plus HPC workflows.
3. *Integration*. Develop integrated WMSs that can support DAC and HPC/in situ workflows.
4. *Scheduling*. Develop mapping and scheduling strategies that can handle DAC and HPC resources in a consistent and integrated manner and can optimize execution across these executions.
5. *Monitoring*. Develop integrated tools for monitoring the execution state and progress of workflows, possibly adapting executions as needed.
6. *Resource provisioning*. Examine the interplay of resource provisioning and workflow management.

4.1.4 Monitoring

State of the Art:
Workflows are playing an increasingly important role in orchestrating complex scientific processes in extreme scale and highly heterogeneous environments. To date, however, we cannot reliably predict, understand, or optimize workflow performance. Sources of performance variability and, in particular, the interdependencies of workflow design, execution environment, and system architecture are not well understood. While there is a rich portfolio of tools for performance analysis, modeling, and prediction for single applications in homogeneous computing environments, these are not applicable to workflows because of the number and heterogeneity of the involved components and their strong interdependencies.

New research is needed to study workflow performance-influencing factors and to devise programming models and templates for workflows. This research will enable reliable performance and evaluation of runtime optimization strategies and mechanisms that work in harmony with the underlying execution environments and system architectures. To date, workflow performance studies are usually focused on a specific workflow or WMS. Many different approaches are used for these investigations; most commonly provenance-based descriptions of the workflow are utilized to describe the executed tasks. These descriptions are then linked to performance measurements such as execution time, average memory access, and I/O load [43]–[48].

Challenges:
Management systems for extreme-scale systems, including exascale, are optimized to support single application performance, based on the fundamental understanding of their performance

characteristics. Unfortunately, these capabilities have not been brought to bear on complex workflows. The reason is that these workflows differ from single applications executed in one homogeneous environment in a number of significant ways: they combine multiple different programming/execution models and heterogeneous execution platforms. As discussed above, the number of workflow and system components, coupled with their heterogeneity and strong interdependencies, provides key challenges in understanding workflow performance variability. Furthermore, their ability to react to hardware and software systems and user-created events at runtime adds another level of complexity.

R&D Needed:

1. Performance variability. Identify and quantify sources of workflow performance variability as they relate to different performance goals, taking into account the workflow tasks, the workflow management system, the execution environments, and system architectures.

2. Event handling. Investigate the impact of system, scientific data, and user events on workflow performance.

3. Tools. Develop suitable performance analysis and modeling tools.

4. Design patterns. Conduct research in effective workflow design templates and programming models.

5. Optimization. Investigate runtime optimization strategies and methodologies, including suitable interactions between the workflow management system and the system software stack including resource provision, runtime system, SSIO, and software defined networking.

4.1.5 Fault Tolerance and Recovery

State of the Art:

With increasing scale and complexity, fault tolerance and recovery are key challenges for both DAC and HPC workflows. Many existing WMSs handle task and system failures and incorporate fault-tolerant mechanisms (e.g., task re-execution or rollback from checkpoints). Research efforts are addressing these issues by extending existing programming systems to support fault tolerance (e.g., ULFM). Emerging task DAG-based programming models also include resilience features [49]. Recent research is exploring fully online mechanisms for resilience, for example, [41], [50].

In addition to addressing system and process failures, there is an increasing need for data validation mechanisms. Data corruption, whether occurring as a result of bugs, attacks, or background radiation is much more likely in complex infrastructure and applications. Research by Croubois et al. [40], [51] aims to design a new generic method that provides efficient error detection capabilities for both systematic and nonsystematic errors using an external algorithmic observer.

Challenges:

As system scales increase and the mean time between failures (MTBF) become smaller, process and node failures, including correlation effects, become important. These failures are often recovered by terminating the job and restarting it from the last checkpoint found in stable storage. However, it is unclear whether this approach will work when the MTBF approaches the time needed to execute the checkpoint.. Similarly, traditional global recovery mechanisms will not scale. As a result, exploring online, application-aware, and possibly local recovery mechanisms is important. Furthermore, resilience algorithms tend to be power hungry, so understanding power requirements as well as costs and tradeoffs between power and resilience guarantees of different fault tolerance mechanisms is critical.

In order to protect against data corruption in workflows, verification is commonly done by using simple pipeline replication. Comparing outputs of both replicas of the same pipeline can help detect corruptions affecting one of the replicas. The downside of this approach is the high cost of replication; and because all the replicas are identical, systematic errors affecting all replicas cannot be detected.

R&D Needed:

1. Online recovery. Research is required into developing online and possibly local recovery mechanisms that can handle the scales, complexities, and failures rates of emerging systems. Exploring application-aware mechanisms will be critical.

2. Resilience management. Programming and runtime support for cross-layered power and resilience management is needed so that application programmers can choose the minimum level of resilience required in each code segment, as well as control the knobs to balance tradeoffs and meet power budgets.

3. Silent data corruption. Develop a generic method that provides efficient error detection capabilities for both systematic and nonsystematic errors. For example, models to detect silent data corruption in a pipeline of several application tasks operating on a single time step of data, without requiring a time series of multiple time steps, can be used to develop new adaptive and resilient WMSs.

4. Software stack vertical error propagation. Explore error propagation and failure recovery across software layers in a way that makes enables users to understand application and system behavior.

4.1.6 Interplay of Workflow Management, Data Management, and Runtime Systems

State of the Art:

Computing at extreme scale greatly complicates the coordination of data and computation management. Data are necessarily distributed across multiple storage systems and are too large to be moved in as a unit to the computation; either it will have to be moved incrementally, or the computation will have to move to the data. In order to analyze the data in a reasonable amount of time, computations on the data will be highly parallel, and this parallelism must take into account data dependencies. To stay within local storage limits, data space usage needs to be managed as part of the overall computation management. At the same time as data distribution in the wide area needs to be considered, it is critical to consider the data management within a high-performance system. Issues not only of performance but also of energy efficiency need to be taken into account.

Challenges:

Today the execution of workflows is handled by a number of different, independent, and un-coordinated systems that each in its own way tries to optimize the workflow or its tasks by considering internal optimization criteria. For example, the workflow management system may try to optimize the overall cost of a workflow (monetary or resource usage related). The data management system that handles individual data transfers or data placement decisions may want to optimize for reliability and storage constraints and may balance its objectives across a number of workflows or other applications. Imagine a scenario where a critical workflow needs to be fast-tracked and the data management system that it needs to query is busy doing data replication operations. When running on HPC systems, workflow tasks are handled by the OS present on the machines. This OS makes its own decisions about data placement for the tasks and task distribution. Although the workflow management has knowledge about the upcoming workload, the data locality present between the workflow tasks, and the task dependencies, this knowledge is not communicated to the HPC-residing OS. Imagine a scenario where a large amount of input data is shared among a subset of workflow tasks that are not scheduled at the same time. After placing the data in memory, the HPC OS, focusing on its own tasks execution order policies and unaware of the data locality at the workflow level, may pick tasks for execution that need a different data set rather than picking the tasks that could reuse the data. The result is additional data movement within the HPC system, which leads to poor task performance and energy usage inefficiency. One solution would be to increase the communications between the workflow management system and the OS; another solution is to distribute and migrate some of the workflow management capabilities into the HPC system so that it can interact more closely with the OS. As these issues are being explored, the OS on the HPC machines is evolving, and it may be it enhanced to handle more complex workloads.

R&D Needed:
1. HPC-DAC interface. What is the interplay between the workflow management system (WMS) and the wide area data placement/data management systems?
2. WMS-OS/R interface. What is the interplay between the workflow management system and the operating system or hardware present on the HPC platform?
3. Streaming. How does WMS support streaming data models and manage tradeoffs of performance, persistence and resilience of data movements?
4. Scalability. Are current centralized workflow management systems sufficient to handle the scale of today's and tomorrow's workloads, or are new distributed workflow management capabilities needed?

4.2 Programming and Usability

Programming and usability were identified as key factors determining the extent of adoption of workflow methods at extreme scale. The challenges are threefold: trelationship between programming models for the workflow and those used for individual tasks in the workflow, the need for workflow design patterns, the different user interfaces in WMSs, and the need to program multiple types of information transfer between tasks. This section addresses how to meet these challenges in a portable way between HPC and DAC workflows.

4.2.1 Programming Models

State of the Art:
Although many HPC workflows are hand-constructed through shell scripts, batch scheduling, and human intervention, there exist programmable tools such as Swift [52], Tigres [53], Kepler [1], Trident [54], Weaver [55], Triana [56], Pegasus [26], Galaxy [57], and Taverna [58] to better manage complicated workflows. The Open Provenance Model (OPM) [59] is an open standard specification of a provenance data model with multiple existing compliant implementations [60]–[62]. Programming models for cloud, web service, and other big data applications are abundant. The programming model for MapReduce [63] is probably the most well known, but many others exist [64]–[66] Many of these models may be leveraged for use in high-performance computing.

Challenges:
As leadership-class machines and the workflows applied to them increase in complexity, the division between workflow and programming model becomes blurred. The workflow system's representation of a mix of coarse data- and task-parallelism mirrors the finer-grained task-parallel computations that are predicted for increasing parallelism on extreme-scale systems beyond the common data-parallelism in applications today. It is unclear where the line between responsibilities of workflow and programming model should lie or indeed if there should be a separation at all. A combined hierarchical representation of tasks characterizing the spectrum of parallelism from coarse-grained jobs to fine-grained processing threads may be most effective.

In addition to better understanding the relationship between workflows and programming models, several other challenges exist.

Mapping. The workflow exists in its own abstract model. This workflow abstraction must be mapped to the physical compute, network, and storage resources while taking into account an accurate model of their relative costs. These decisions may define how workflows are composed.

Provenance. The capture of provenance is critical for future analysis and reproducibility in workflow systems. It is important to capture all relevant information including that available only after a workflow completes (such as user information regarding the quality of the solution).

Pragmatics. Just as science teams exhibit different workflow patterns, their modes of interaction with systems, the data they use and generate, and their rate of adoption of workflow-related tools will vary.

R&D Needed:
Because both workflows and programming models address high concurrency, dynamic application execution, dynamic resource availability, architectural diversity, and new forms of in-system storage in extreme-scale architectures, research must manage the gaps in increased complexity.

1. Horizon and coordination. Since the resource allocation responsibilities of workflows and programming models overlap, integration and coordination become fruitful and perhaps necessary. Programming models and system introspection could provide knowledge to the workflow manager or even to the programmer.
2. Mapping. Based on the application context, domain-specific workflow systems could be used to exploit proposed programming systems that assemble applications composed of different algorithms or implementations. Compared with a more general workflow system, domain specificity could potentially simplify the mapping of workflows to diverse resources.
3. Provenance. For workflows that change dynamically as a result of system resource allocation or user steering, we need to capture runtime decisions that affect execution, even if it is not possible to replicate the exact execution.
4. Pragmatics. Workflow systems must provide productive and bidirectional interfaces, and workflow tools should be decomposed in such a way that science teams can iteratively adopt components into their existing work practices.

4.2.2 Design Patterns

State of the Art:
The basic mental model for a workflow is a directed acyclic graph (DAG) representing the tasks to perform and the dependencies between the tasks. Workflow management tools such as Swift [52] and Tigres [53] center their API on building DAGs and internally manage parallel execution and dependencies within them. Other tools such as AVS [67], SCIRun [68], and VisTrails [69] allow users to build and view workflow DAGs visually with a graphical representation and user interface. Wings [70] uses the idea of templates to represent the overall workflow structure and then automatically fills out the template based on user needs.

Many scientists build workflows by example, which is an informal use of design patterns; they iteratively construct one workflow using a previous one as a template. This incremental use simplifies the process of building and running workflows. This iterative modification of existing workflows shortens the development time and can also be integrated in software design to accelerate workflow tool development [71].

Some recent tools use patternlike structures as part of the creation and execution of workflows. For example, Tigres has a collection of templates that can be applied when building workflow structures [72]. VisTrails can collect the provenance of many previously built workflows to find common patterns that can automatically assist users in other endeavors [73].

Challenges:
Many workflows, particularly those from the same research group, are similar. These can potentially be defined as patterns with data, error control, and reproducibility. They can be characterized by their resource access—network traffic, high throughput, an so forth.—to take the best advantage of resources while being portable. Science teams exhibit a variety of different workflow patterns, particularly across communities. Understanding these patterns can aid in the design of workflows and workflow systems.

However, it is unclear that any single WMS would be effective for supporting such a variety of patterns. Scientific workflow needs tcan change depending on the domain or the mode of operation, such as batch versus real time versus interactive.

Although a DAG is the fundamental model used to describe workflows and their patterns, interdependencies among tasks result in cycles in the dependency graph. Such interdependencies are common in, for example, multiphysics codes where independent tasks communicate with each other and take turns to converge to the appropriate solution. Such cycles can sometimes be handled by conditionals or dynamic scripting in the workflow tool, but better descriptions of the workflow could be made.

R&D Needed:
1. Design patterns. It is important to understand and classify various workflow and workflow needs through user studies. Identifying common patterns, akin to design patterns in software engineering [74], for next-generation in situ and distributed workflows, is needed to address programmability and usability concerns.
2. Cycles. Workflows need to correctly and more formally handle task dependency loops. Part of this effort requires workflows to understand and manage time and data that change over time, as demonstrated in the similar VTK dataflow network [75].
3. Templates. Research is needed into ways to help users easily develop high-level templates and to instantiate them for concrete problems.
4. Ensembles. Scientists often use ensembles of workflows to represent and overall analysis. Research in ensemble management is needed to support the end-to-end computational methods.

4.2.3 User Interface

State of the Art:
Most workflow management tools use a scripting language to define tasks and dependencies and to manage execution [1], [26], [52]–[58]. Within the scripting language is an API that scripts use to define and execute a workflow. There also exist examples of workflow building tools that use a graphical interface [68], [69]. Such interfaces provide a tradeoff between simplicity and expressiveness.

Challenges:
Today, workflows are modeled in many different ways including scripts and application/programming models. The boundaries and interactions between the representation of workflow constructs and application interaction such as loops and parameter convergence/divergence are not well understood.

A constant tension exists between generalizing and specializing the workflow user interface. A generalized interface provides a greater amount of expression that can address more application domains and more anomalous cases, whereas a specialized interface tends to be easier to use and could provide more opportunities to optimize the workflow process.

The human-in-the-loop model occurs in many different types of workflows including exploration and failure recovery in production pipelines. The real-time status of the workflow needs to be accessible to users. Many simulations require human monitoring for erroneous conditions such as the entangling of a mesh. Real-time status is also important for observational and experimental data. For example, a human may be required to determine whether missing telescope data is a result of a cloudy night or failures in the hardware or software. Similarly, light source experiments can benefit from real-time feedback to improve experimental decisions. Today, only limited support is provided to integrate the human in workflows and to automatically track the provenance from such activities.

R&D Needed:
1. Abstraction. The appropriate level of abstraction for workflows is unclear. Indeed, this level of abstraction may not be uniform for all use cases; different domains may require different abstractions. Having different levels of abstractions for early and advanced users could also be advantageous. Given a level of abstraction, what hints can a user provide to better map the workflow to the available resources?
2. Human in the loop. Providing the interface for human-in-the-loop workflows is critical, but as yet no way exists to properly capture the provenance of human interaction. To preserve repeatability, we must find ways to capture when a human makes changes, what changes were made, and the reason for the changes. It may be advantageous to study how business workflow models and techniques [76] can be helpful in this regard.

4.2.4 Task Communication

State of the Art:
Many workflows have used files to transmit data between programs. For such an interface to work seamlessly, the programs must understand each other's file format. Thus, many HPC file formats are "self-describing" in that arrays are organized using names, attributes, and hierarchies [31], [77]. Similar organization is present in array databases [78] and NoSQL databases [79]. This organization, however, has little meaning without an agreement on the semantics. Thus, conventions [80] and schemas [81]–[83] are often applied.

Other work has focused on providing a unified interface to data that can come from a variety of storage implementations. For example, ADIOS can support numerous I/O backends and switch between them at runtime [31]. Tools such as Google Dremmel [84] and Apache Drill provide a unified interface to multiple data backends.

Challenges:
Today's workflows and workflow systems have limited support for considering data sources, storage needs, and data models. As workflows integrate ever-varying tasks, it becomes more challenging to communicate data between software that uses different data models. Solutions attempting to provide a unified data model have often been unsuccessful. Many file and database systems provide successful mechanisms for declaring data format, but these mechanisms need to be expanded and applied to intertask communication within a workflow.

R&D Needed:
1. Data heterogeneity. We need to develop appropriate infrastructure that allows seamless integration of various data sources including streaming, data management across the memory-storage hierarchy of next generation systems, and data semantics in user workflows.
2. Data independence. Greater understanding is needed about how data can be communicated among tasks that are developed independently and have different data models.
3. Data sources. Data can be thought of as stored or streamed in from multiple sources. The mechanism for connecting these sources and the ability to identify, convert, and verify data models must be well established. This communication will need to take place over a variety of levels in a deep memory hierarchy.

4.2.5 Portability

Many analysis codes are run as in situ processes, meaning that all tasks run within a local supercomputer's resources, and sometimes run as distributed area processes where tasks are coordinated across multiple independent systems that may be physically distant from each other. These workflows should not be implemented twice. Therefore, their interface needs to be designed to work in either in situ or in distributed area modes. For example, the same code may need to point to data in memory, read data from a file, output data to memory or a file, run in serial or parallel, compute a small-scale or large-scale job, process data in core or out of core, and be built as a library or as an executable. All of this ought to be uniform so that data and control can seamlessly flow between HPC and DAC environments.

State of the Art:
No widely adopted general-purpose workflow tools are available that work seamlessly across both HPC and DAC. However, many specific applications are designed to work in both domains. CyberShake, a seismic hazard model from the Southern California Earthquake Center [85], combines components that use high-performance computing and high-throughput computing. The Advanced Photon Source coordinates high-performance computation with detector hardware and other processing systems [86]. Likewise, the National Synchrotron Light Source II has initial processing in situ; the results then are sent to remote users. The HACC cosmology simulation can interface its high-performance computation with analysis codes on other systems through the CosmoTools analysis framework [87]. KBase, the DOE systems biology knowledgebase, contains in situ modeling and reconstruction tools as well as offloading to cloud-based distributed-area systems [88].

Challenges:
To address workflows across HPC and DAC boundaries and over multiple applications, we need to find a common language for building workflow tools. This may be a job for the emerging field of "workflow science," which itself is highly related to data science.

We expect steering and human interaction to become more important. Therefore, better tools are needed to express and enable dynamic control within the workflow execution. Many problems require a human-in-the-loop because the most complex decisions cannot be programmed. Furthermore, scientists need real-time feedback on simulation progress and perhaps need to decide what to do in case of failures. It is not clear how human interaction can be accommodated in situ and ex situ.

Portability is difficult on complex, heterogeneous systems. Part of the solution falls to other areas of research such as programming models, but workflows can also help. Workflows can help match tasks to the architecture best suited to run them; this situation is more common in DAC workflows, but it is still an area of research for HPC. Containers such as Docker, with workflows operating above the container level, are a possible solution for some distributed area and in situ workflow issues.

R&D Needed:
1. Performance portability. How can we build WMSs and common application components that operate with good performance in both HPC and DAC?
2. Containerization. What is the role of containers and other virtualization technologies in workflows? Virtualization can have a profound effect both for workflow components (tasks) and for the workflow systems themselves.
3. Security. Security for in situ workflows often relies on the access controls of the system, whereas distributed area workflows must be more cognizant of security since they run across different systems in different security domains. Often the security policies between HPC, DAC, and the compute facilities conflict with each other. How can security be unified across all these elements to allow application workflows to best be developed, deployed, and executed?
4. Architecture portability. Can workflows be leveraged to manage deep memory hierarchies? Given an appropriate decomposition of the problem, as workflows orchestrate tasks, they can manage the movement of data up and down this memory hierarchy. For example, workflows could potentially manage movement of data between out-of-core and in-core, between NVM and main memory, and between main RAM and high bandwidth RAM.
5. Human interaction. As high-performance computing and workflows become more complex, the interaction between human and system becomes more important. For large-scale applications, it is not feasible to continually monitor tasks and restart jobs when problems occur or when steering is necessary. Rather, tasks need a real-time communication channel to a human who can interact, modify, or correct behavior when necessary. The human interaction can become even more complex as we mix HPC and DAC in the same workflow.

4.3 Provenance Capture

WMSs offer a unique opportunity for provenance capture because they encapsulate the process of solving a computational problem. They are the central managers of many different operations and, at the same time, interact closely with many other relevant components and resources related to the execution of the workflow. However, the number of components, the complexity of their connections, and the rate of execution complicates provenance capture in workflows.

4.3.1 Content, Format, and Level of Detail

Today, provenance usually represents a simple directed graph, describing one level of abstraction of an environment or process (e.g., a single workflow representation [89], [90]). Some systems such as Pegasus provide details about the environment in which the workflow is executed. Workflow are treated as black boxes irrespective of their internal complexity. However, scientific discovery rarely follows a straight path and instead is characterized by trial and error of different approaches and reuse of partial prior results and methods. Science is inherently collaborative, with the processes of different researchers aligning and intersecting at different intervals. We can see similar collaborative behaviors in complex, extreme-scale computing environments, where the interactions of applications, workflows, system software, and hardware intersect and align at different times in different ways. Because these interactions can have a profound impact on workflow performance, accuracy, and traceability, it is important to adequately capture these interactions to support prediction, optimization, and validation. In terms of provenance, such interactions can be seen as independent streams of provenance that need to connect, interact, and align.

State of the Art:
Before 2011, the community developed a range of workflow provenance models including tool-specific solutions (e.g., Vistrails in UVCDAT [91]) and more standardized formats such as D-PROV [92]. To enable greater interoperability between provenance models, a working group for the World Wide Web Consortium (W3C) defined the core specification for an open provenance model (OPM) [59] in 2011. Subsequently, a range of OPM-compliant workflow provenance models have been developed, such as OPMW [61], D-OPM [60], and the work of Lim et al. [62]. All these models focus exclusively on the description of the workflow, usually in graph form, describing its components and the data utilized [57], [93]–[95]. Some of the existing provenance models can account for decisions made by the workflow; few capture information such as resource scheduling. All models treat workflow tasks as black boxes, with few details attached and none of them captures details of the execution environment.

Challenges:
Validation, traceability, and reproducibility. With the advent of complex workflows in extreme-scale computing environments, new opportunities for provenance usage are emerging, such as validating workflow performance, traceability of workflow execution, and workflow reproducibility. Addressing these tasks requires the additional capture of more detailed information about the tasks, the execution environment, and WMS itself. Furthermore, we need to be able to account for the interconnected nature of different levels of the system software, application stack, distributed systems, hierarchical workflows, and scientific collaboration itself. These developments require a fundamental shift in how we describe provenance: moving from solutions for one abstraction level to a time series of different states, with different components and resource needs that are potentially interconnected during certain time ranges.

Describing in situ data deduction. With increasing costs of data movement and I/O bottlenecks at extreme scale, workflows for computational science must shift from saving data for post hoc analysis to incorporating various forms of data analysis and visualization in situ. A key challenge is the development of sufficiently descriptive and detailed provenance models to capture adaptive data reduction processes at runtime to enable further processing, validation, and interpretation post hoc.

Real-time provenance. Today, provenance captures what the workflow has done, but it might be worthwhile considering whether the same models could express what is likely to happen. If so, they could be used to express future resource needs to the workflow's execution environment.

R&D Needed:
1. Scalability. Research will be needed into new provenance models that are suitable for supporting new usage models. This work will need to consider the tradeoffs between the required expressivity and level of detail versus the volume of information produced.
2. Heterogeneity. We need to investigate how to move from one provenance model with one stream of provenance (i.e., from a workflow) to a system where we have many streams that align and intersect at certain time intervals.

4.3.2 Capture

Workflow provenance to date is captured through a variety of mechanisms; few have, however, been designed to work efficiently in extreme-scale environments for complex workflows.

State of the Art:
The methods for provenance collection generally fall into three categories: workflow event listeners, application logs, and direct calls to a provenance vocabulary-based API. Workflow listeners provide a means to directly collect and record workflow events such as start/stop time stamps and parameters/data used. Workflow provenance is typically asynchronously collected, and workflow events are ordered by the calling order, making transitive closure possible. For application provenance vocabulary APIs, provenance is collected through API calls at application execution time; any provenance collected relies on the developer making calls to the API [96], [97]. For log files, event history is derived from logs at runtime and reconstructed as provenance by using a provenance vocabulary API [98]. Logger APIs support logging at different granularities: fatal, error, warn, info, debug, and trace. Interpretation of the log file entries is done through a monitor application that analyzes the log files and creates the provenance entries [99], [100]. The majority of the available solutions focused on the collection of relatively low velocity and volume provenance data; only a few projects have started to explore high-volume, high-velocity capture mechanisms. One approach is the use of messaging services such as Apache Axis2/RabbitMQ [101] or Apache Kafka/AVRO [102] to facilitate high-velocity provenance transmission. In distributed or extreme-scale computing environments, provenance capture can at times be unreliable [103] and lead to incomplete provenance records.

Challenges:

High-speed provenance capture mechanisms. Existing workflow provenance capture solutions are effective for low-volume capture requirements. If the community wants to support new provenance applications that require high-velocity provenance capture, then new approaches, in particular for extreme-scale systems with deep memory hierarchies, have to be found.

Provenance completeness. The structure and content of provenance records can be incomplete [104], in particular in extreme-scale environments. Dropped messages can result in missing nodes or edges in the provenance graph. Soft, hard, or silent errors can lead to missing or incorrect content in provenance messages. Furthermore, interrupted or failed workflow processes due to system errors can lead to incomplete provenance graphs.

R&D Needed:
1. Velocity. Research is needed in high-velocity provenance capture mechanisms in extreme-scale HPC and DAC environments.
2. Completeness. Reliable capture of complete and correct provenance data in extreme-scale environments will require new capture approaches, in particular when coupled with high-velocity capture methods.

4.3.3 Communication across System Software

In complex HPC and DAC environments, the resources used, their properties, and resilience strategies can have a significant impact on the WMS performance, correctness, and reproducibility. For optimized performance and correctness, increasingly adaptive WMSs would ideally be aware of OS/R actions. Conversely, the WMS might provide critical information to the OS/R to help with its operation and resource allocation. To date, no common approach is available that would enable processes to communicate across the system stack with each other. Provenance could provide such a mechanism, enabling the initial exchange and capture of relevant events, as well as postevent analysis.

State of the Art:
While the majority of provenance work is focused on data lineage and workflow documentation, a number of attempts have been made to develop provenance-enabled system services such as provenance-aware storage [105], [106], distributed storage [107], file system [108], distributed file systems [109], kernel [110], and networks [111], [111]. Few solutions, however, cover multiple aspects and levels of the operating system. Hi-Fi [110] is one of the most comprehensive conceptual approaches and offers a kernel-level approach to trace the data flow through systems, processes, and threads across files and file systems, memory mappings, pipes, message queues, and sockets. Other approaches [89], [90], [105] also aim to facilitate provenance capture across different system layers. However, none goes as deeply into the operating and storage systems layers as Hi-Fi. To date these systems have been tested only in small-scale, single-system environments.

Challenges:

Runtime provenance exchange across system software levels and systems. None of the systems available today can communicate provenance across system layers. Indeed, the majority of the systems were implemented with post hoc forensic analysis in mind (i.e. storage in log files or databases); thus, they do not include provisions for real-time exchange, negotiation, and analysis. Only a few solutions have started to explore how to exchange and capture this type of extensive provenance across systems, in extreme-scale systems, or through complex memory hierarchies.

Communicating runtime provenance to workflows and users to aid decision making. Similarly, no system offers the ability to communicate with WMSs or humans.

R&D Needed:

1. Software stack individual layer capture. The capture and utilization of provenance information across system software levels and distributed systems constitute a new field of research that will require investigations at many different levels.

2. Interlayer communication. We need to study, characterize, and model the type of processes that are usefully supported by such an approach; what information needs to be captured, when, and to whom it needs to be communicated. Effective capture and communication mechanisms, in situ triage, and analysis are other areas of R&D need.

4.3.4 Archival

Provenance information, if captured in some detail, can create a significant volume of data. At extreme scale, even more provenance information will need to be captured, not only about the workflow itself, but also about the tasks it is executing and the environment in which it runs. At the same time, we foresee a much more active usage of the provenance information in these environments. Effective archival and access mechanisms are needed to enable scientists and applications to utilize the captured provenance information.

State of the Art:

In the past, provenance was predominantly collected in log files [91], XML files, and relational databases [112], [113]. Today, we see an increasing shift to using RDF and graph databases [114] or systems such as HBASE [115]. However, all solutions have their limitations in terms of the speed at which provenance can be ingested, the volume of information that can be stored, and the efficiency with which information can be extracted. Proprietary commercial graph databases offer the best performance but are not a practical solution because of their costs. Hybrid graph and relational solutions [116], or in-memory, parallel graph databases such as SGEM [117], [118] could provide an alternative. In all cases, provenance can reach a significant size, at times far exceeding the size of the original data or workflow implementation [113]. For this reason, a significant research effort has focused on provenance compression techniques that leverage the inherent duplication across provenance collections. Factorization and inheritance are utilized to remove duplicate provenance entries, only storing a single instance [112]–[114]. All these methods are designed to work over collections of provenance, rather than individual records as one might see in situ. For streaming data environments, such as wireless sensor networks, methodologies such as arithmetic coding [119], dictionary-based compression [120], event-only recording [121], and overlay of provenance data [122] are used to reduce the footprint of the individual provenance object. Today, all provenance records are assumed to be of value in perpetuity; however, in extreme-scale environments with a wealth of new sources of provenance, this might not be the case. System health performance provenance might only be needed until the correct execution of a task has been confirmed.

Challenges:
Fast storage and retrieval. A key challenge when supporting complex workflows in extreme-scale environments is the provision of fast storage and retrieval for large volumes of provenance information from different sources. Of particular concern is the traversal of potentially large geographical distances between provenance-creating resources in DAC environments and the deep memory hierarchies of HPC systems, not to mention the sheer volume and velocity of the provenance information.

R&D Needed:
1. Storage and delivery. Research is needed to evaluate effective storage and delivery approaches for provenance objects including fast storage, search, and retrieval in DAC and HPC environments. A deeper understanding is needed of provenance flow and usage characteristics for different classes of extreme-scale workflows.
2. Tools. Tools are required that can characterize and model this flow for classes of workflows in a variety of execution environments.
3. Performance and energy tradeoffs. Also needed is investigation of performance and energy tradeoffs between different archival and delivery locations and methods: specifically, approaches that span multiple systems and memory hierarchy levels need to be evaluated.
4. Speed. While much research has been done covering different archival methods and organization, further work is needed on the fast storage and retrieval of extreme-scale provenance information in order to support real-time decision-making.
5. Resilience. Furthermore, if the archival system includes unreliable resources (e.g., interim storage in network devices or exascale memory), resiliency measures, and their performance and energy costs need to be considered.

4.3.5 Usage, Analysis, and Data Mining

While major research efforts have focused on general approaches to describe, capture, and manage provenance data, analysis of these data today is ad hoc and focused on specific application cases.

State of the Art:
Provenance analysis generally is used to extract knowledge from a provenance collection, or in rare cases to validate the quality of the provenance. Mining of provenance data has emerged as a key approach for its analysis, for both relational and graph databases. The main usage of provenance mining focuses on extracting key features and trends from large volumes of

provenance information that is complex and rich in features. A range of approaches have been developed to aid the design or amendment of workflows through recommender systems based on past provenance information. Methods include event log file mining [123], [124], recommendations based on data dependencies [125]–[127], or a combination of the two approaches [128]. Provenance can be structurally incomplete, include erroneous or incomplete information, or be inconsistent. Its correctness can be assessed through contextual analysis, while completeness can be assessed through structural analysis [104].

Challenges:
Understanding new usage models. New types of provenance information and extreme-scale DAC and HPC usage environments are opening new avenues to utilize provenance information effectively for the validation and verification of workflows, workflow performance variability investigations, performance improvements, workflow resilience, workflow result interpretation, and provenance as a communication media between workflows and underpinning system software services. Some of these use cases will require the ability to interactively explore and investigate large volumes of complex, multilevel provenance information, turning it into actionable insights. Others ill require the summation and delivery of provenance information to operational processes in fixed-time windows. Overall, there is a particular interest in researching and characterizing the new use cases and determining the analytical approaches that will be required to satisfy these.

Representing provenance information. Provenance is usually represented as a graph of interconnecting information: few, if any, representation paradigms available today would allow a user to explore, query, and interact with this information effectively. This challenge is compounded in extreme-scale DAC and HPC environments, where provenance not only will be available in large volumes but is likely to be very complex in its structure.

R&D Needed:
1. Representation. Research is needed to understand and characterize provenance analysis models as they will be required in HPC and DAC extreme-scale workflows.
2. Analysis.This research will tie directly to the other research themes described in this section, providing the means to analyze large and complex provenance information, both in situ and post hoc.

4.4 Validation

The increasing complexity of both workflows and their computational environments makes it critical to provide the HPC community with the approaches, methods, and tools to ensure that workflows are executed with sufficient reproducibility in terms of performance and accuracy. The dynamics of complex workflows can range from a simple bag of tasks (e.g., MG-RAST [129] and DOCK [130]) and sets of distributed applications with intermediate key-value pairs that from MapReduce (e.g., data histograms for high energy physics [131] and object ordering [132]), to more sophisticated iterative chains of MapReduce jobs (e.g., graph mining [133]), sets of distributed applications with multiple stages using files for intermediate data (e.g., Montage [134], BLAST [135], and CyberShake postprocessing [136]), to iterative applications with a varying sets of tasks that must be run to completion in each iteration (e.g., Kalman filtering [137]). Extreme-scale systems, with their higher degrees of concurrency, and in situ data analysis that is triggered by specific events in very large-scale modeling and simulation applications, will further accentuate the community's need for validation and reproducibility criteria [138].

Performance and accuracy validation are expected to become more important, from being purely post hoc to becoming a key run-time function that ultimately enables mitigation and optimization at the extreme-scale.

4.4.1 Performance Validation and Prediction

State of the Art:
Application-specific performance tools are available for performance validation and prediction [139]. While we have sophisticated tools to monitor, model, and predict the performance of single applications, no comparable capabilities exist for complex HPC or DAC workflows. The situation is aggravated by the fact that large parts of the WMS are hidden to stakeholders. Attempts to address the situation include skeletons for performance prediction [140]–[142] and targeted distributed-computing middleware such as Pegasus [26], [143], [144] and Swift [52], [145], [146].

Challenges:
Only a few applications are expected to run in isolation on exascale systems, and for them the traditional tools for measuring performance will likely work. Most applications, however, will run in concert, sharing and competing for resources. Under these conditions, performance analysis and prediction must deal with multiple workflows running through the set of resources, some of them compute-intensive, others I/O-intensive, and others both. Performance goals can be expressed and measured either for the entire facility or for the specific workflow, and how to best express performance goals and assess their achievement for a facility versus for a specific workflow is an open challenge. Both cases have a wide range of objectives (e.g., resource utilization, in-time delivery of results, energy usage minimization, and accuracy). It is unclear exactly what metrics need to be monitored, how they can be captured, and at what granularity performance must be predicted and validated, all the while ensuring that stakeholder goals are being met.

Communities other than HPC can provide hints for addressing these issues. For example, service-level agreements (SLAs) in cloud computing are not HPC-oriented. Instead, they build in abort mechanics for which repeated results of the same workflow are considered sufficiently similar (and thus delivered to the stakeholder) if within a defined region of interest or tolerance. In other words, cloud executions deploy the concept of an accurate-enough service within a certain time threshold and amount of resources. For example, a search does not explore all the data but just finds the top 50 results within an acceptable time. An open question here is whether we can use this approach from the cloud community and develop a related vision of performance—a vision that is latency driven but preserves a sufficient level of scientific accuracy and result reproducibility.

R&D Needed:

1. *Capabilities*. Research is needed to extend performance validation capabilities and approaches from single applications to ensembles of applications and their workflows.
2. *Tools*. New research should promote the design and implementation of tools capable of monitoring and modeling runtime performance—in order to validate and improve future predictions and models—by capturing event information at different levels and granularity.
3. *Analysis*. I/O bottlenecks make the in situ analysis of the performance data essential.
4. *Feedback*. Intelligent schedulers should use the knowledge of events' occurrences to optimize the performance according to defined goals, despite the challenges introduced by new extreme-scale architectures, execution environments, geographical distribution, and scientific instruments. Runtime use of acquired knowledge can be used to continually improve performance and optimize resource use at the scheduler level—for example, by simultaneously running storage- and compute-intensive jobs.

4.4.2 Validation and Performance Predictability

State of the Art:

Performance validation goes hand in hand with performance predictability. Science validation is achieved by repeating simulations and comparing the repeated results with real-world results for reproducibility [147]. For extreme-scale workflows, however, the meaning of performance prediction and result reproducibility needs to be reviewed. When referring to performance prediction, instead of the traditional definition of performance as average time to solution for the single workflow, stakeholders use ad hoc performance metrics ranging from executing as fast as possible, to being done by a certain time, to minimizing resource use (e.g., energy, disk, memory, or core-hours). Moreover, the definition of result reproducibility often depends on the community or even the individual scientist's point of view. It may range from stringent bitwise reproducibility to reproducibility of the scientific conclusions across multiple executions. When different methods are used, reproducibility refers to "closeness of agreement among repeated simulation results under the same initial conditions over time," and accuracy refers to "conformity of a resulted value to an accepted standard (or scientific laws)" [148]. Some disagreement exists in defining who is responsible for performance and results validation (i.e., the application, workflow, or system).

Challenges:

As the complexity of applications increases, the workflow must play a major role in application performance prediction and result reproducibility. The question is how the stakeholder can keep track of aspects of the workflow execution at the exascale, especially when competing for resources with other workflows and when frequent unexpected events occur (e.g., silent errors and node failures). Information that can provide insights into the entire execution environment can be overwhelming and substantially slow the entire workflow execution. Clear standards are needed that define the responsibility of the workflow and the information that needs to be tracked at a particular level, be it at the application, workflow, or system level.

R&D Needed:
1. Predictability. Research is needed that targets the integration of applications, programming models, and OS/R to pursue performance predictability and results reproducibility. Solutions should handle information flow between all layers of the OS/R, WMS, and applications, with the goal of observing and attributing performance and accuracy deviations to specific workflow paths and system resources.
2. Reproducibility. Automatic annotations embedded in multilayer and modular workflows are desirable, where workflows provide documentation of paths taken and resources used, with the aim of making workflows reproducible. While applications must continue to be responsible for providing results of acceptable accuracy, in many instances the workflow is the right place to validate and propagate performance and accuracy expectations and achievements.

Provenance as defined in Section 4.3 can provide a framework for capturing and relating information critical to in situ and post hoc validation of workflow performance, reproducibility, and accuracy.

4.4.3 Accuracy and Reproducibility

State of the Art:
The responsibility for guaranteeing accuracy, validation, and data reproducibility is traditionally the user's, through choice of algorithm and implementation [149], [150]. The inability to reproduce data at the application level can be the result of arithmetic and algorithm factors [151], [152] or a simulation setting. Arithmetic and algorithm factors include the nonassociativity of floating-point arithmetic and nondeterminism in the order of operations. In response to these challenges, mathematical techniques can be applied to mitigate the degree to which computed sums exhibit sensitivity to reduction order. Such techniques can range from simple fixed-reduction orders (for which imposing the same order adds cost in time and power [153]), to interval arithmetic (for which the interval range can explode and 10x slowdowns have been observed [154]), and to extended precisions (for which it is difficult to predict how much precision a specific application run needs [155]). Compensated summation algorithms [156], composite precision [157]–[159], and prerounded algorithms [160], [161] show promising results but require some level of code modification and performance tuning [162].

One problem is that users may not know or fully understand all the variables that need to be controlled and hence do not report these variables or do not appropriately control them in the simulations. Additionally, physical mappings, as occur in the solutions to differential equations for most of nature's most interesting systems, are themselves chaotic at some level. For example, two trajectories of the same folding protein initiated infinitesimally close to one another can diverge because of the level of ergodicity of the system [163]. Thus, a repeated simulation may not yield the same result or the same scientific conclusions. No test suite exists to check regularly for workflow and system inaccuracy associated with arithmetic errors, algorithmic factors, or simulation settings. In other words, no component of the entire system is watching the correctness of the workflow manager.

Challenges:
By establishing the accuracy of components a priori, workflow systems can monitor specific variables to see whether the workflow is progressing as intended. The use of data mining, machine learning, and statistical methods can identify deviations and possibly correct them. However, the hidden parts of increasingly complex workflows remain a major challenge when pursuing accuracy and reproducibility. In order to answer the questions of how to convey only the needed information to the stakeholder and to determine exactly what that information is, a tradeoff must be made between the workflow's information capture and the simulation's accuracy. Specifically, if a workflow does not provide enough information to repeat the simulation, then its scientific validity cannot be tested or improved. Conversely, too much information is a deterrent to workflow performance [164], [165].

The tradeoff should be driven by the use of accuracy metrics that determine what data to keep or throw away. Open questions include how much the stakeholder can be added in the loop and what role the stakeholder can play in determining accuracy (i.e., what deviation is acceptable) and in deciding on suitable actions. Workflows need to provide documentation of the paths taken and to check that the overall simulation is behaving correctly. In other words, we envision workflows capable of supporting the stakeholder in validating science data.

R&D Needed:
1. *Accuracy of results*. We need to pursue reproducibility of results in order to guarantee accurate science. Reproducibility is the basis for many validation approaches. Typically, reproducibility relies on a comparison between different models or between models and observations or experiments. When publishing and disseminating results, we need to provide sufficient provenance so that others can reproduce the results.
2. *Validation driven by science needs*. Reproducibility, enabled by provenance, is a fundamental requirement for the validation of complex workflows; however, the level, detail, and lifetime of the information gathered must be regulated by the reproducibility drivers (e.g., scientific explanation, performance validation, accuracy validation) and any events that occur during execution (e.g., delays, user interaction, errors).
3. *Algorithms and system software*. Any solution explored for exascale systems should consider the impact on reproducibility of numerical approaches, programming models, execution environments, system architecture, system optimization, component faults, workflow optimization, and user steering.

4.5 Workflow Science

Over the past decade, scientific workflows have emerged and proved to be an enabling technology for computational science. This situation has happened at a time when there have been many advances in scientific computing technologies including web, grid, cloud, big Data, and tera/peta/exascale computing systems. Now, more than ever, efficient WMSs are needed to connect computationally intensive codes and instruments, in situ analysis methodologies, near-real-time processing, and interactive access to analytical visualizations.

To date, workflow research has focused mainly on automation of the science data collection process; conformance with standards and agreements to move data, metadata, and instructions between workflow tasks; and emerging of in situ and real-time experimental data. As requirements for workflow management expand and workflow technologies mature, new research needs to pave the way for *workflow science*: a scientific approach to prove the correct operation of workflows and infrastructure leading to development of new workflow models and experiments.

In other words, research is required not only to develop and enhance workflow tools and services, but also to fully *understand how workflows operate*. Such research targets questions related to the expected operation of the workflow, its relationship with the infrastructure (computer, instrument, network, storage), and the trustworthiness and reproducibility of experimental results. As in other sciences, workflow science has three main branches that serve as three legs of discovery in this field: theory, computation, and experiment. These three branches have large and complex problem spaces spanning multiple scientific disciplines and communities with disparate legacy tools and services. With myriad disconnected results in the literature, there is a strong need for a common language, metrics, theory, and tools before workflows can become useful at extreme scale. The practitioners of this new discipline will be the *workflow scientists* of tomorrow.

State of the Art:

To date, little work has been done in the area of theoretical workflow science. Most of the effort has focused on workflow development and enabling the use of workflow technologies. Workflow science is needed to develop the foundations for integrating the workflow with applications and computing systems. Such codesign will be based on models and simulations of applications and facility resources (computers, storage, clusters, networks, instruments). A systematic approach to experimentation is needed in order to validate the theories and simulations and to optimize scheduling and resource management.

Challenges:

The experimental workflow science challenges include determining how to decide which data to collect, how to collect that data, and how to analyze it in terms of methods and tools. Along with experimentation come concerns of experiment repeatability and validation against theoretical models and simulations.

Computational workflow science focuses on developing accurate but also forward looking simulations of workflow execution. Thus the challenges include the determination of the necessary models and simulation tools, issues of the efficiency and scalability of the simulations, and simulation validation.

Much of the experimental and computation workflow science needs to rely on solid theory in order to provide important insights and knowledge. The challenges in developing a theoretical understanding of workflow behavior include characterizing the impact of the execution environment on workflow performance and behavior, determining what predictions can be made about workflow behavior on current and future infrastructures, and deciding what future infrastructures should be like in order to support efficient workflows.

R&D Needed:

1. Theory. Theoretical workflow science is complex systems research: new theories and experiments are needed for validation of behavior models of workflows under different computation models. More expressive workflow systems and languages are needed that can capture heterogeneous models of computation for workflows at extreme scale.

2. Experimentation. A deeper analysis of profiling, system workloads, and workflow execution is needed to create new models for predicting performance. Simulations based on the results of these predictive workflow models are needed in order to optimize workflow execution at extreme scale.

3. Dissemination. More R&D is needed to create new knowledge that explains observed behavior of scientific workflows. Community-centered knowledge repositories are needed to openly share the experimental data and benchmarks; these will advance workflow science and lead to more reproducible research.

5. Summary of Findings and Research Priorities

We first summarize here the findings presented in the preceding sections. We then prioritize the recommended research. Each subsection is divided into these areas: application requirements, hardware systems, software systems, WMS design and execution, programming and usability, provenance capture, validation, and workflow science.

5.1 Findings

Application requirements

- Research is needed in the areas of automation, human interaction, provenance, and validation in order to support simulation HPC workflows.
- In order to address the challenges of experimental and observational workflows, the WMS must coordinate the end-to-end workflow life cycle, including real-time scheduling and execution of measurement instruments and supercomputers.
- Collaborative workflows are characterized by heterogeneous users and resources, and WMSs must be portable, shareable, and reproducible across a diverse set of environments in order to be usable by all members of a collaboration.

Hardware systems

- Metrics are needed for valuing performance, power, and productivity. How to measure productivity is an open question. Proxy applications to test workflow workloads are needed for such metrics.
- Mechanisms for passing data between tasks without unnecessary data movement should be investigated.
- Memory management systems that support scratchpad and NVM are needed for workflows.

Software systems

- Efficient low-overhead scheduling of multiple cooperative tasks, various forms of communication (messages, interrupts, publish-subscribe, etc.) between independent tasks, and provisioning of shared resources (e.g., shared storage) among tasks are needed from the OS/R to support the WMS.
- The WMS needs the OS/R to move from its traditional role of managing single tasks to managing global (i.e., internode, distributed) services. Such global management may be through a hierarchy of resource groupings (enclaves), with heterogeneous programming models and runtimes managing the resources within a given enclave or task.
- The WMS needs to negotiate with the OS/R through a well-defined interface on behalf of the entire application workflow. The OS/R must provide the WMS with the system calls to coordinate various tasks (such as the simulation and data analysis codes) and capture the provenance that scientists need in order to support writing papers and

validating scientific results, including capturing any changes from the initial workflow that result from human-in-the-loop interactive analysis and steering.

WMS design and execution

- Coupling control and data flow between heterogeneous components requires research in data flows that can buffer, prefetch, aggregate, and distribute data.
- Data management research must solve challenges in the transport, layout, attributes, and provenance of data
- The convergence of DAC and HPC workflows into a single, possibly hierarchical, WMS presents challenges in managing nonuinform latency, task granularity, and reliability.

Programming and usability

- The diverse needs of workflows impact usability. For example, real-time workflows need immediate cognition. In addition, there are many different ways to express workflows spanning from simple to complex. Research is needed to identify common needs and expression patterns (akin to design patterns in software engineering) in workflows with respect to data management, error control, reproducibility, programmability, and mapping to physical resources.
- Research is needed to determine appropriate levels of abstraction in the user interface for workflow systems and their component modules, including an interface for human-in-the-loop interactive workflows.
- Many commonalities exist between HPC and DAC workflows. Although some examples of workflow applications cross HPC and DAC boundaries, most are specific to a particular application. WMS and application components must be developed that operate with good performance across heterogeneous platforms. Understanding the role of containers, virtualization, and security—features found in DAC—is needed in HPC. Understanding the effect of disruptive technologies such as deep memory hierarchies and NVM is needed as well.

Provenance capture

- Research is needed in new provenance models. These will support new usage models with many provenance streams that align and intersect at certain time intervals.
- The capture and utilization of provenance across system software levels and distributed systems will require investigations at many different levels, including effective capture and communication mechanisms, effective fast storage, search, retrieval, in situ triage, and analysis.
- Mining of provenance data has emerged as a key approach for its analysis, both for relational and graph databases.

Validation

- The increasing complexity of workflows and their computational environments make it critical to provide the community with the approaches, methods, and tools to ensure and validate that workflows are executed with reproducible accuracy and performance to meet scientific goals.
- Exascale systems with their higher degrees of concurrency will further accentuate this community need.
- Validation is expanding from a pure post hoc operation to a key runtime function that can enable dynamic mitigation and optimization.

Workflow science

- One role of workflow science is to develop and execute theoretical models that describe the expected behavior of workflows.
- Workflows can be modeled computationally. The community should strive to design and execute computational simulations and models that describe the workflow behavior.
- More research is needed to design and execute large complex experiments in order to validate the function of scientific workflows and the facility resources (computers, storage devices, networks, instruments) used by these workflows.

5.2 Research Priorities

Application requirements

Workflows for both computational and experimental sciences need investigation, as do workflows to support scientific collaboration. Sharing workflows, migrating workflows between different computing environments, accommodating different user roles, and combining different languages and software tools used by various users all require further research.

Hardware systems

Extreme-scale hardware challenges for workflows—power, performance, resilience, and productivity—require significant planning and investment in system software, programming environments, applications, and WMSs. Heterogeneous nodes, new memory systems including NVM, and little growth in bandwidth to external storage systems or networks dictate new use patterns for WMS that leverage in situ analytics and heterogeneous programming models for individual workflow tasks.

Software systems

Supercomputers today are intended to run single-program batch jobs, the opposite of workflows. Workflows by definition are collections of multiple programs whose execution must be coordinated by the WMS. Human interaction with the workflow (for example, to steer a computation in one program based on a result in another) is another challenge. Resource allocations must treat storage, I/O, and network capacity as first-class resources to be allocated, managed, and measured to the same degree as computing capacity is today. Scheduling HPC and DAC resources over several systems will require cooperative schedulers that can coordinate with the WMS. Schedulers, resource allocators, and file systems will need to expose runtime provenance data.

WMS design and execution

The efficient management of DAC and HPC workflows present challenges at multiple levels of software. At the base layer, coupling control and data flow between heterogeneous components requires expanding workflow links into data flows that can buffer, prefetch, aggregate, and distribute data. At a higher level of abstraction, managing the workflow must address concurrency, locality, and system topology if data movement is to be optimized at extreme scale.

Programming and usability

Major challenges include lack of standardization between numerous programming models for tasks and workflows, the interconnection between the programming of individual tasks and entire workflows, and the portability of both code and data across different locations in a potentially heterogeneous workflow that spans both HPC and DAC resource.

Provenance capture

WMSs offer a unique opportunity for provenance capture, because they encapsulate the process of solving a computational problem. The increasing scale and complexity of hardware and software systems, however, coupled with the composition of multiple tasks by workflows, complicates provenance capture. The velocity of provenance data generated at extreme scale requires new methods to compress, mine, analyze it, store, and share it.

Validation

The increasing complexity of workflows and their computational environments makes it critical to provide the approaches, methods, and tools to ensure that workflows are executed with sufficient reproducibility in terms of performance and accuracy. Validation of expected performance is a complex high-dimensional space of metrics over a heterogeneous computing architecture. Needed research directions include extending single-application performance validation tools to workflows of applications and developing methods to learn what level of differences in science results are statistically significant.

Workflow science

Workflow science is a new field that studies the formal theory and design principles of workflow systems, develops analytical models for their performance and validation, and empirically measures the predicted behavior through experimentation.

Glossary

ACME:	Accelerated Climate Modeling for Energy
ALS:	Advanced Light Source
ALCF:	Argonne Leadership Computing Facility
APS:	Advanced Photon Source
DAC:	distributed-area computing
DAG:	directed acyclic graph
DRAM:	dynamic random access memory
EPSI:	Center for Edge Physics Simulation
FPGA:	field-programmable gate array
JGI:	Joint Genome Institute
HPC:	high-performance computing
KBase:	Systems Biology Knowledgebase
LSST:	Large Synoptic Survey Telescope
MG-RAST:	Metagenomics RAST Server
MTBF:	mean time between failure
NERSC:	National Energy Research Scientific Computing Center
NVM:	nonvolatile memory
OS/R:	operating system and runtime
PTF:	Palomar Transient Factory
QoS:	quality of service
SLA:	service-level agreement
SOA:	service-oriented architecture
SSIO:	storage systems and input/output
ULFM:	user-level fault mitigation
WMS:	workflow management system

Acknowledgments

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Argonne National Laboratory is managed by UChicago Argonne, LLC, for the U.S. Department of Energy Office of Science under contract DE-AC02-06CH11357.

References

[1] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludäscher, 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.

[2] 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.

[3] “Titan,” *Titan*, 2015. [Online]. Available: https://www.olcf.ornl.gov/titan/.

[4] E. Deelman, C. Carothers, A. Mandal, B. Tierney, J. S. Vetter, I. Baldin, C. Castillo, G. Juve, D. Król, V. Lynch, B. Mayer, J. Meredith, T. Proffen, P. Ruth, and R. F. da Silva, “PANORAMA: An Approach to Performance Modeling and Diagnosis of Extreme-Scale Workflows,” *Int. J. High Perform. Comput. Appl.*, p. 1094342015594515, Jul. 2015.

[5] D. A. Patterson, P. Chen, G. Gibson, and R. H. Katz, “Introduction to redundant arrays of inexpensive disks (RAID),” in *COMPCON Spring ’89. Thirty-Fourth IEEE Computer Society International Conference: Intellectual Leverage, Digest of Papers.*, 1989, pp. 112–117.

[6] L. Rizzo, “Effective Erasure Codes for Reliable Computer Communication Protocols,” *SIGCOMM Comput Commun Rev*, vol. 27, no. 2, pp. 24–36, Apr. 1997.

[7] K. Birman, “The Promise, and Limitations, of Gossip Protocols,” *SIGOPS Oper Syst Rev*, vol. 41, no. 5, pp. 8–13, Oct. 2007.

[8] E. N. (Mootaz) Elnozahy, L. Alvisi, Y.-M. Wang, and D. B. Johnson, “A Survey of Rollback-recovery Protocols in Message-Passing Systems,” *ACM Comput Surv*, vol. 34, no. 3, pp. 375–408, Sep. 2002.

[9] L. Lamport, “Paxos made simple,” *ACM Sigact News*, vol. 32, no. 4, pp. 18–25, 2001.

[10] P. Carns, Y. Yao, K. Harms, R. Latham, R. Ross, and K. Antypas, “Production I/O Characterization on the Cray XE6,” in *Cray User Group Meeting*, 2013.

[11] “ALCF I/O Data Repository.” [Online]. Available: http://press3.mcs.anl.gov/darshan/data/.

[12] H. Abbasi, M. Wolf, G. Eisenhauer, S. Klasky, K. Schwan, and F. Zheng, “DataStager: Scalable Data Staging Services for Petascale Applications,” in *Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing*, New York, NY, USA, 2009, pp. 39–48.

[13] E. Barton, J. Bent, and K. Quincey, “Fast Forward Storage and IO Program Documents - HPDD Community Space – HPDD Community Wiki.” [Online]. Available: https://wiki.hpdd.intel.com/display/PUB/Fast+Forward+Storage+and+IO+Program+Documents.

[14] F. Isaila, J. Garcia Blas, J. Carretero, R. Latham, and R. Ross, “Design and Evaluation of Multiple-Level Data Staging for Blue Gene Systems,” *IEEE Trans Parallel Distrib Syst*, vol. 22, no. 6, pp. 946–959, Jun. 2011.

[15] S. Kannan, A. Gavrilovska, K. Schwan, D. Milojicic, and V. Talwar, “Using Active NVRAM for I/O Staging,” in *Proceedings of the 2Nd International Workshop on Petascal Data Analytics: Challenges and Opportunities*, New York, NY, USA, 2011, pp. 15–22.

[16] W.-K. Liao, A. Ching, K. Coloma, A. Choudhary, and L. Ward, “An Implementation and Evaluation of Client-Side File Caching for MPI-IO,” in *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, 2007, pp. 1–10.

[17] 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 *Proceedings of the 2012 IEEE Conference on Massive Data Storage*, 2012.

[18] J. F. Lofstead, S. Klasky, K. Schwan, N. Podhorszki, and C. Jin, “Flexible IO and Integration for Scientific Codes through the Adaptable IO System (ADIOS),” in *Proceedings*

of the 6th International Workshop on Challenges of Large Applications in Distributed Environments, New York, NY, USA, 2008, pp. 15–24.

[19] A. Moody, G. Bronevetsky, K. Mohror, and B. R. de Supinski, “Design, Modeling, and Evaluation of a Scalable Multi-ILvel Checkpointing System,” in *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, Washington, DC, USA, 2010, pp. 1–11.

[20] X. Qin, H. Jiang, A. Manzanares, X. Ruan, and S. Yin, “Dynamic Load Balancing for I/O-Intensive Applications on Clusters,” *CSE J. Artic.*, 2009.

[21] T. Wickberg and C. Carothers, “The RAMDISK Storage Accelerator: A Method of Accelerating I/O Performance on HPC Systems Using RAMDISKs,” in *Proceedings of the 2Nd International Workshop on Runtime and Operating Systems for Supercomputers*, New York, NY, USA, 2012, pp. 5:1–5:8.

[22] 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, pp. 182–189.

[23] S. Perarnau, R. Thakur, K. Iskra, K. Raffenetti, F. Cappello, R. Gupta, P. Beckman, M. Snir, H. Hoffmann, M. Schulz, and B. Rountree, “Distributed Monitoring and Management of Exascale Systems in the Argo Project,” in *Distributed Applications and Interoperable Systems*, A. Bessani and S. Bouchenak, Eeds. Springer International Publishing, 2015, pp. 173–178.

[24] R. Brightwell, R. Oldfield, A. B. Maccabe, and D. E. Bernholdt, “Hobbes: Composition and Virtualization As the Foundations of an Extreme-scale OS/R,” in *Proceedings of the 3rd International Workshop on Runtime and Operating Systems for Supercomputers*, New York, NY, USA, 2013, pp. 2:1–2:8.

[25] Robert Ross, Gary Grider, Evan Felix, Mark Gary, Scott Klasky, Ron Oldfield, Galen Shipman, and John Wu, “Storage Systems and Input/Output to Support Extreme Scale Science,” Rockville, MD, 2014.

[26] 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 Gener. Comput. Syst.*, vol. 46, pp. 17–35, 2015.

[27] C. Docan, M. Parashar, and S. Klasky, “DataSpaces: An Interaction and Coordination Framework for Coupled Simulation Workflows,” in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, New York, NY, USA, 2010, pp. 25–36.

[28] J. Dayal, D. Bratcher, G. Eisenhauer, K. Schwan, M. Wolf, X. Zhang, H. Abbasi, S. Klasky, and N. Podhorszki, “Flexpath: Type-Based Publish/Subscribe System for Large-Scale Science Analytics,” in *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Chicago, IL, USA,* May 26-29, 2014, pp. 246–255.

[29] C. Docan, M. Parashar, J. Cummings, and S. Klasky, “Moving the Code to the Data - Dynamic Code Deployment Using ActiveSpaces,” in *Proceedings of the 2011 IEEE International Parallel & Distributed Processing Symposium*, Washington, DC, USA, 2011, pp. 758–769.

[30] F. Zheng, H. Abbasi, C. Docan, J. F. Lofstead, Q. Liu, S. Klasky, M. Parashar, N. Podhorszki, K. Schwan, and M. Wolf, “PreDatA – Preparatory Data Analytics on Peta-Scale Machines,” in *IPDPS’10*, 2010, pp. 1–12.

[31] J. F. Lofstead, S. Klasky, K. Schwan, N. Podhorszki, and C. Jin, “Flexible IO and Integration for Scientific Codes Through the Adaptable IO System (ADIOS),” in *Proceedings of the 6th International Workshop on Challenges of Large Applications in Distributed Environments*, New York, NY, USA, 2008, pp. 15–24.

[32] V. N. Bhat, “Autonomic Management of Data Streaming and In-Transit Processing for Data Intensive scientific Workflows,” 2008.

[33] J. Dayal, J. Cao, G. Eisenhauer, K. Schwan, M. Wolf, F. Zheng, H. Abbasi, S. Klasky, N. Podhorszki, and J. Lofstead, “I/O Containers: Managing the Data Analytics and Visualization Pipelines of High End Codes,” in *Proceedings of the 2013 IEEE 27th International Symposium on Parallel and Distributed Processing Workshops and PhD Forum*, Washington, DC, USA, 2013, pp. 2015–2024.

[34] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, and I. Foster, “The Globus Striped GridFTP Framework and Server,” in *Supercomputing*, 2005.

[35] C. Docan, M. Parashar, and S. Klasky, “Enabling High-speed Asynchronous Data Extraction and Transfer Using DART,” *Concurr Comput Pr. Exper*, vol. 22, no. 9, pp. 1181–1204, Jun. 2010.

[36] M. Jergler, M. Sadoghi, and H.-A. Jacobsen, “D2WORM: A Management Infrastructure for Distributed Data-centric Workflows,” in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 2015, pp. 1427–1432.

[37] Q. Sun, T. Jin, M. Romanus, H. Bui, F. Zhang, M. Parashar, H. Yu, H. Kolla, and J. Chen, “Adaptive Data Placement for Staging-based Coupled Scientific Workflows,” Rutgers University, Piscataway, NJ, 2015.

[38] F. Zheng, H. Abbasi, J. Cao, J. Dayal, K. Schwan, M. Wolf, S. Klasky, and N. Podhorszki, “In-situ I/O Processing: A Case for Location Flexibility,” in *Proceedings of the Sixth Workshop on Parallel Data Storage*, New York, NY, USA, 2011, pp. 37–42.

[39] J. C. Bennett, H. Abbasi, P.-T. Bremer, R. Grout, A. Gyulassy, T. Jin, S. Klasky, H. Kolla, M. Parashar, V. Pascucci, P. Pebay, D. Thompson, H. Yu, F. Zhang, and J. Chen, “Combining In-situ and In-transit Processing to Enable Extreme-Scale Scientific Analysis,” in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, Los Alamitos, CA, USA, 2012, pp. 49:1–49:9.

[40] F. Cappello, “Fault Tolerance in Petascale/ Exascale Systems: Current Knowledge, Challenges and Research Opportunities,” *Int. J. High Performance Comput. Appl.*, vol. 23, no. 3, pp. 212–226, Aug. 2009.

[41] M. Gamell, D. S. Katz, H. Kolla, J. Chen, S. Klasky, and M. Parashar, “Exploring Automatic, Online Failure Recovery for Scientific Applications at Extreme Scales,” in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Piscataway, NJ, USA, 2014, pp. 895–906.

[42] M. F. Aktas, G. Haldeman, and M. Parashar, “Flexible Scheduling and Control of Bandwidth and In-Transit services for End-to-End Application Workflows,” in *Proceedings of the Fourth International Workshop on Network-Aware Data Management, NDM ’14, New Orleans, Louisiana, USA,* November 16-21, 2014, pp. 28–31.

[43] D. de Oliveira, K. A. C. S. Ocaña, F. A. Baião, and M. Mattoso, “A Provenance-Based Adaptive Scheduling Heuristic for Parallel Scientific Workflows in Clouds,” *J. Grid Comput*, vol. 10, no. 3, pp. 521–552, 2012.

[44] K. A. Talukder, A.K.M, M. Kirley, and R. Buyya, “Multiobjective Differential Evolution for Scheduling Workflow Applications on Global Grids,” Concurrency and Computation: Practice and Experience, vol. 21, no. 13, pp. 1742–1756, 2009.

[45] H.-L. Truong, S. Dustdar, and T. Fahringer, “Performance Metrics and Ontologies for Grid Workflows,” *Future Gener. Comput. Sys.t*, vol. 23, no. 6, pp. 760–772, Jul. 2007.

[46] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, “Characterizing and Profiling Acientific Workflows,” *Future Gener. Comput. Syst.*, vol. 29, no. 3, pp. 682–692, Mar. 2013.

[47] M. Burtscher, B.-D. Kim, J. Diamond, J. McCalpin, L. Koesterke, and J. Browne, “PerfExpert: An Easy-to-Use Performance Diagnosis Tool for HPC Applications,” in *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, Washington, DC, USA, 2010, pp. 1–11.

[48] T. Samak, D. Gunter, M. Goode, E. Deelman, G. Mehta, F. Silva, and K. Vahi, “Failure Prediction and Localization in Large Scientific Workflows,” in *Proceedings of the 6th Workshop on Workflows in Support of Large-scale Science*, New York, NY, USA, 2011, pp. 107–116.

[49] J. Wilke, J. Bennett, H. Kolla, K. Teranishi, N. Slattengren, and J. Floren, “Extreme-Scale Viability of Collective Communication for Resilient Task Scheduling and Work Stealing,” in *Proceedings of the 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, Washington, DC, USA, 2014, pp. 756–761.

[50] K. Teranishi and M. A. Heroux, “Toward Local Failure Local Recovery Resilience Model Using MPI-ULFM,” in *Proceedings of the 21st European MPI Users’ Group Meeting*, New York, NY, USA, 2014, pp. 51:51–51:56.

[51] T. Peterka, H. Croubois, N. Li, S. Rangel, and F. Cappello, “Self–Adaptive Density Estimation of Particle Data,” preprint ANL/MCS-P5334-0115, 2015.

[52] M. Wilde, M. Hategan, J. M. Wozniak, B. Clifford, D. S. Katz, and I. Foster, “Swift: A Language for Distributed Parallel Scripting,” *Parallel Compu.t*, vol. 37, no. 9, pp. 633–652, Sep. 2011.

[53] L. Ramakrishnan, S. Poon, V. Hendrix, D. K. Gunter, G. Z. Pastorello, and D. A. Agarwal, “Experiences with User-Centered Design for the Tigres Workflow API,” in *10th IEEE International Conference on e-Science, eScience 2014, Sao Paulo, Brazil, October 20-24, 2014*, 2014, pp. 290–297.

[54] R. Barga, J. Jackson, N. Araujo, D. Guo, N. Gautam, and Y. Simmhan, “The Trident Scientific Workflow Workbench,” in *Proceedings of the 2008 Fourth IEEE International Conference on eScience*, Washington, DC, USA, 2008, pp. 317–318.

[55] P. Bui, L. Yu, and D. Thain, “Weaver: Integrating Distributed Computing Abstractions into Scientific Workflows Using Python,” in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, 2010, pp. 636–643.

[56] D. Churches, G. Gombas, A. Harrison, J. Maassen, C. Robinson, M. Shields, I. Taylor, and I. Wang, “Programming Scientific and Distributed Workflow with Triana Services: Research Articles,” *Concurr Comput Pr. Exper*, vol. 18, no. 10, pp. 1021–1037, Aug. 2006.

[57] J. Goecks, A. Nekrutenko, J. Taylor, and others, “Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences,” 2010.

[58] T. Oinn, M. Greenwood, M. Addis, M. N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. R. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe, “Taverna: Lessons in Creating a Workflow Environment for the Life Sciences: Research Articles,” *Concurr. Comput. Pr. Exper.*, vol. 18, no. 10, pp. 1067–1100, Aug. 2006.

[59] 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, June 2011.

[60] V. Cuevas-Vicenttín, S. C. Dey, M. L. Y. Wang, T. Song, and B. Ludäscher, “Modeling and Querying Scientific Workflow Provenance in the D-OPM,” in *2012 SC Companion: High Performance Computing, Networking Storage and Analysis, Salt Lake City, UT, USA,* November 10-16, 2012, pp. 119–128.

[61] D. Garijo and Y. Gil, “Towards Open Publication of Reusable Scientific Workflows: Abstractions, Standards, and Linked Data,” Citeseer, 2012. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.222.9322.

[62] C. Lim, S. Lu, A. Chebotko, and F. Fotouhi, “Storing, Reasoning, and Querying OPM-compliant Scientific Workflow Provenance Using Relational Databases,” *Future Gener Comput Syst*, vol. 27, no. 6, pp. 781–789, June 2011.

[63] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” in *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation – Volume 6*, Berkeley, CA, USA, 2004, pp. 10–10.

[64] G. Fox, J. Qiu, and S. Jha, “High Performance High Functionality Big Data Software Stack,” 2014. http://www.exascale.org/bdec/sites/www.exascale.org.bdec/files/whitepapers/fox.pdf

[65] S. Jha, J. Qiu, A. Luckow, P. K. Mantha, and G. Fox, “A Tale of Two Data-Intensive Paradigms: Applications, Abstractions, and Architectures,” in *2014 IEEE International Congress on Big Data, Anchorage, AK, USA,* June 27 - July 2, 2014, pp. 645–652.

[66] J. Qui, S. Jha, A. Luckow, and G. Fox, “Towards HPC-ABDS: An Initial High-Performance Big Data Stack,” presented at the Building Robust Big Data Ecosystem ISO/IEC JTC 1 Study Group on Big Data, SDSC.

[67] C. Upson, T. Faulhaber Jr., D. Kamins, D. H. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, and A. van Dam, “The Application Visualization System: A Computational Environment for Scientific Visualization,” *IEEE Comput Graph Appl*, vol. 9, no. 4, pp. 30–42, Jul. 1989.

[68] S. G. Parker and C. R. Johnson, “SCIRun: A Scientific Programming Environment for Computational Steering,” in *Supercomputing, 1995. Proceedings of the IEEE/ACM SC95 Conference*, 1995, pp. 52–52.

[69] L. Bavoil, S. P. Callahan, P. J. Crossno, J. Freire, and H. T. Vo, “VisTrails: Enabling interactive multiple-view visualizations,” in *IEEE Visualization 2005*, pp. 135–142.

[70] R. Bergmann and Y. Gil, “Similarity Assessment and Efficient Retrieval of Semantic Workflows,” *Inf. Syst.*, vol. 40, pp. 115–127, Mar. 2014.

[71] K. Maheshwari, D. Kelly, S. J. Krieder, J. M. Wozniak, D. S. Katz, M. Zhi-Gang, and M. Mookherjee, “Reusability in Science: From Initial User Engagement to Dissemination of Results,” *CoRR*, vol. abs/1309.1813, 2013.

[72] J. R. Balderrama, M. Simonin, L. Ramakrishnan, V. Hendrix, C. Morin, D. Agarwal, and C. Tedeschi, “Combining Workflow Templates with a Shared Space-based Execution Model,” in *Proceedings of the 9th Workshop on Workflows in Support of Large-Scale Science*, Piscataway, NJ, USA, 2014, pp. 50–58.

[73] T. Silva, J. Freire, and S. P. Callahan, “Provenance for visualizations: Reproducibility and beyond,” in *Computing in Science & Engineering*, 2007.

[74] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-oriented Software*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., 1995.

[75] J. Biddiscombe, B. Geveci, K. Martin, K. Moreland, and D. Thompson, “Time Dependent Processing in a Parallel Pipeline Architecture,” *IEEE Trans. Vis. Comput. Graph.*, vol. 13, no. 6, pp. 1376–1383, 2007.

[76] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. F. Ferguson, *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005.

[77] Prabhat and Q. Koziol, *High Performance Parallel I/O*, Chapman & Hall/CRC Computational Science. CRC Press.

[78] M. Stonebraker, P. Brown, A. Poliakov, and S. Raman, “The Architecture of SciDB,” in *Proceedings of the 23rd International Conference on Scientific and Statistical Database Management*, Berlin, Heidelberg, 2011, pp. 1–16.

[79] R. Cattell, “Scalable SQL and NoSQL Data Stores,” *SIGMOD Rec*, vol. 39, no. 4, pp. 12–27, May 2011.

[80] B. Eaton, J. Gregory, B. Drach, K. Taylor, and M. Juckes, “NetCDF Climate and Forecast (CF) Metadata Conventions, Version 1.6, 5 December, 2011.”

[81] J. A. Clarke and E. R. Mark, “Enhancements to the eXtensible Data Model and Format (XDMF),” in *DoD High Performance Computing Modernization Program Users Group Conference, 2007*, 2007, pp. 322–327.

[82] S. Shasharina, D. Alexander, J. Cary, M. Durant, S. Kruger, and others, “VizSchema – a Unified Visualization of Computational Accelerator Physics Data,” *in Proceedings of IPAC'10, Kyoto, Japan*, vol. C100523, p. TUPEC069, 2010.

[83] R. Tchoua, J. Choi, S. Klasky, Q. Liu, J. Logan, K. Moreland, J. Mu, M. Parashar, N. Podhorszki, D. Pugmire, and M. Wolf, “ADIOS Visualization Schema: A First Step Towards Improving Interdisciplinary Collaboration in High Performance Computing,” in *Proceedings of the 2013 IEEE 9th International Conference on e-Science*, Washington, DC, USA, 2013, pp. 27–34.

[84] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton, and T. Vassilakis, “Dremel: Interactive Analysis of Web-scale Datasets,” in *Proceedings of VLDB Endow*, vol. 3, no. 1–2, pp. 330–339, Sept. 2010.

[85] R. Graves, T. H. Jordan, S. Callaghan, E. Deelman, E. Field, G. Juve, C. Kesselman, P. Maechling, G. Mehta, K. Milner, D. Okaya, P. Small, and K. Vahi, “CyberShake: A Physics-Based Seismic Hazard Model for Southern California,” *Pure Appl. Geophys.*, vol. 168, no. 3, pp. 367–381, 2011.

[86] F. Khan, J. . Hammonds, S. Narayanan, A. Sandy, and N. Schwarz, “Effective End-to-end Management of Data Acquisition and Analysis for X-ray Photon Correlation Spectroscopy,” presented at the ICALEPCS 2013, 2013.

[87] S. Habib, A. Pope, H. Finkel, N. Frontiere, K. Heitmann, D. Daniel, P. Fasel, V. Morozov, G. Zagaris, T. Peterka, V. Vishwanath, Z. Lukic, S. Sehrish, and W. -k. Liao, “HACC: Simulating Sky Surveys on State-of-the-Art Supercomputing Architectures,” *ArXiv E-Prints*, Oct. 2014.

[88] M. N. Benedict, M. B. Mundy, C. S. Henry, N. Chia, and N. D. Price, “Likelihood-Based Gene Annotations for Gap Filling and Quality Assessment in Genome-Scale Metabolic Models,” *PLoS Comput. Biol.*, vol. 10, no. 10, 2014.

[89] K.-K. Muniswamy-Reddy, U. Braun, D. A. Holland, P. Macko, D. Maclean, D. Margo, M. Seltzer, and R. Smogor, “Layering in Provenance Systems,” in *Proceedings of the 2009 Conference on USENIX Annual Technical Conference*, Berkeley, CA, USA, 2009, pp. 10–10.

[90] K. Kleese van Dam, E. Stephan, B. Raju, I. Altinas, T. Elsethagen, and S. Krishnamoorthy, “Enabling Structured Exploration of Workflow Performance Variability in Extreme-Scale Environments,” draft, 2015.

[91] F. Chirigati, J. Freire, D. Koop, and C. Silva, “VisTrails Provenance Traces for Benchmarking,” in *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, New York, NY, USA, 2013, pp. 323–324.

[92] P. Missier, S. Dey, K. Belhajjame, V. Cuevas-Vicenttín, and B. Ludäscher, “D-PROV: Extending the PROV Provenance Model with Workflow Structure,” in *Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance*, Berkeley, CA, USA, 2013, pp. 9:1–9:7.

[93] B. Giardine, C. Riemer, R. C. Hardison, R. Burhans, L. Elnitski, P. Shah, Y. Zhang, D. Blankenberg, I. Albert, J. Taylor, W. Miller, W. J. Kent, and A. Nekrutenko, “Galaxy: A Platform for Interactive Large-Scale Genome Analysis,” *Genome Res.*, vol. 15, no. 10, pp. 1451–1455, Oct. 2005.

[94] C. E. Scheidegger, H. T. Vo, D. Koop, J. Freire, and C. T. Silva, “Querying and Re-using Workflows with VsTrails,” in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 2008, pp. 1251–1254.

[95] 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 Res.*, vol. 41, pp. W557–561, Jul. 2013.

[96] P. Macko and M. Seltzer, “A General-Purpose Provenance Library,” in *Proceedings of the 4th USENIX Conference on Theory and Practice of Provenance*, Berkeley, CA, USA, 2012, p. 6.

[97] F. Costa, V. Silva, D. de Oliveira, K. A. C. S. Ocaña, and M. Mattoso, “Towards Supporting Provenance Gathering and Querying in Different Database Approaches,” in *Provenance and Annotation of Data and Processes - 5th International Provenance and Annotation Workshop, IPAW 2014, Cologne, Germany, June 9–13, 2014. Revised Selected Papers*, 2014, pp. 254–257.

[98] D. Ghoshal and B. Plale, “Provenance from log files: a BigData problem,” in *1st International Workshop on Managing and Querying Provenance Data at Scale*, 2013.

[99] 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*, vol. 8628, B. Ludäscher and B. Plale, Eds. Springer International Publishing, 2015, pp. 71–83.

[100]A. Marinho, L. Murta, C. Werner, V. Braganholo, S. M. S. da Cruz, E. Ogasawara, and M. Mattoso, “ProvManager: A Provenance Management System for Scientific Workflows,” *Concurr Comput Pr. Exper*, vol. 24, no. 13, pp. 1513–1530, Sept. 2012.

[101]I. Suriarachchi, Q. Zhou, and B. Plale, “Komadu: A Capture and Visualization System for Scientific Data Provenance,” *J. Open Res. Softw.*, vol. 3, no. 1, 2015.

[102]K. Van Dam, P. Sharma, R. LaMothe, D. Zarshitsky, A. Vishnu, E. Stephan, W. Smith, T. Elsethagen, and M. Thomas, “Building the Analysis in Motion Infrastructure,” PNNL Technical Report, PNNL-24340, 2015.

[103]Y.-W. Cheah, B. Plale, J. Kendall-Morwick, D. Leake, and L. Ramakrishnan, “A Noisy 10GB Provenance Database,” in *Business Process Management Workshops*, vol. 100, F. Daniel, K. Barkaoui, and S. Dustdar, eds. Springer Berlin Heidelberg, 2012, pp. 370–381.

[104]Y.-W. Cheah and B. Plale, “Provenance Analysis: Towards Quality Provenance,” in *IEEE 8th International Conference on E-Science,* 2012, pp. 1–8.

[105]K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. Seltzer, “Provenance-Aware Storage Systems,” in *Proceedings of the Annual Conference on USENIX ’06 Annual Technical Conference*, Berkeley, CA, USA, 2006, p. 4.

[106]J. Ledlie, C. Ng, and D. A. Holland, “Provenance-Aware Sensor Data Storage,” in *Proceedings of the 21st International Conference on Data Engineering Workshops*, Washington, DC, USA, 2005, p. 1189–.

[107]M. Bădoiu, K. Muniswamy-Reddy, A. Sidiropoulos, and M. Vutukuru, *A Distributed Provenance-Aware Storage System*, in Proceedings of the USENIX Annual Technical (USENIX '06), Berkeley, CA, USA, p. 4.

[108]S. Sultana and E. Bertino, “A File Provenance System,” in *Proceedings of the Third ACM Conference on Data and Application Security and Privacy*, New York, NY, USA, 2013, pp. 153–156.

[109]C. Shou, D. Zhao, T. Malik, and I. Raicu, *Towards a provenance-aware distributed filesystem*. CiteSeer http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.365.6946.

[110]D. J. Pohly, S. McLaughlin, P. McDaniel, and K. Butler, “Hi-Fi: Collecting High-fidelity Whole-system Provenance,” in *Proceedings of the 28th Annual Computer Security Applications Conference*, New York, NY, USA, 2012, pp. 259–268.

[111]W. Zhou, E. Cronin, B. T. Loo, W. Zhou, E. Cronin, and B. T. Loo, “Provenance-aware Declarative Secure Networks,” University of Pennsylvania Department of Computer and Information Science, University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-07-27, 2007.

[112]A. P. Chapman, H. V. Jagadish, and P. Ramanan, “Efficient Provenance Storage,” in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 2008, pp. 993–1006.

[113]Y. Xie, D. Feng, Z. Tan, L. Chen, K.-K. Muniswamy-Reddy, Y. Li, and D. D. E. Long, “A Hybrid Approach for Efficient Provenance Storage,” in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, New York, NY, USA, 2012, pp. 1752–1756.

[114]R. Pinheiro, B. Aires, A. F. Araujo, M. Holanda, M. E. T. Walter, and S. Lifschitz, “Storing Provenance Data of Genome Project Workflows Using Graph Database,” in *2014 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2014, Belfast, United Kingdom, November 2-5, 2014*, 2014, pp. 16–22.

[115]A. Chebotko, J. Abraham, P. Brazier, A. Piazza, A. Kashlev, and S. Lu, “Storing, Indexing and Querying Large Provenance Data Sets As RDF Graphs in Apache HBase,” in *Proceedings of the 2013 IEEE Ninth World Congress on Services*, Washington, DC, USA, 2013, pp. 1–8.

[116]J. Weaver, A. Chappell, W. Smith, S. Purohit, P. Fox, P. West, and B. Lee, “Resource Discovery for Extreme Scale Collaboration,” presented at the NGNS 13.

[117]A. Chappell, S. Choudhury, J. Feo, D. Haglin, A. Morari, S. Purohit, K. Schuchardt, A. Tumeo, J. Weaver, and O. Villa, “Toward a Data Scalable Solution for Facilitating Discovery of Scientific Data Resources,” in *Proceedings of the 2013 International Workshop on Data-Intensive Scalable Computing Systems*, New York, NY, USA, 2013, pp. 55–60.

[118]V. G. Castellana, A. Morari, J. Weaver, A. Tumeo, D. Haglin, O. Villa, and J. Feo, “In-Memory Graph Databases for Web-Scale Data,” *Computer*, vol. 48, no. 3, pp. 24–35, March 2015.

[119]S. R. Hussain, C. Wang, S. Sultana, and E. Bertino, “Secure data provenance compression using arithmetic coding in wireless sensor networks,” in *Performance Computing and Communications Conference (IPCCC), 2014 IEEE International*, 2014, pp. 1–10.

[120]C. Wang, S. Hussain, and E. Bertino, “Dictionary Based Secure Provenance Compression for Wireless Sensor Networks,” *IEEE Trans. on Parallel Distrib. Syst.*, 2015.

[121]N. N. Vijayakumar and B. Plale, “Towards Low Overhead Provenance Tracking in Near Real-time Stream Filtering,” in *Proceedings of the 2006 International Conference on Provenance and Annotation of Data*, Berlin, Heidelberg, 2006, pp. 46–54.

[122]S. Sultana, M. Shehab, and E. Bertino, “Secure Provenance Transmission for Streaming Data,” *IEEE Trans. on Knowl. Data Eng.*, vol. 25, no. 8, pp. 1890–1903, Aug. 2013.

[123]W. van der Aalst, T. Weijters, and L. Maruster, “Workflow Mining: Discovering Process Models from Event Logs,” *IEEE Trans Knowl Data Eng*, vol. 16, no. 9, pp. 1128–1142, Sept. 2004.

[124]B. F. van Dongen, A. K. Alves de Medeiros, and L. Wen, “Process Mining: Overview and Outlook of Petri Net Discovery Algorithms,” in *Transactions on Petri Nets and Other Models of Concurrency II*, vol. 5460, K. Jensen and W. P. van der Aalst, eds. Springer Berlin Heidelberg, 2009, pp. 225–242.

[125]D. Leake and J. Kendall-Morwick, “Towards Case-Based Support for e-Science Workflow Generation by Mining Provenance,” in *Proceedings of the 9th European Conference on Advances in Case-Based Reasoning*, Berlin, Heidelberg, 2008, pp. 269–283.

[126]J. Zhang, Q. Liu, and K. Xu, “FlowRecommender: A Workflow Recommendation Technique for Process Provenance,” in *Proceedings of the Eighth Australasian Data Mining Conference - Volume 101*, Darlinghurst, Australia, Australia, 2009, pp. 55–61.

[127]W. Tan, J. Zhang, and I. Foster, “Network Analysis of Scientific Workflows: A Gateway to Reuse,” *Computer*, vol. 43, no. 9, pp. 54–61, Sept. 2010.

[128]R. Zeng, X. He, and W. M. P. van der Aalst, “A Method to Mine Workflows from Provenance for Assisting Scientific Workflow Composition,” in *IEEE World Congress on Services (SERVICES),* 2011, pp. 169–175.

[129]F. Meyer, D. Paarmann, M. D’Souza, R. Olson, E. Glass, M. Kubal, T. Paczian, A. Rodriguez, R. Stevens, A. Wilke, J. Wilkening, and R. Edwards, “The Metagenomics RAST Server – A Public Resource for the Automatic Phylogenetic and Functional Analysis of Metagenomes,” *BMC Bioinformatics*, vol. 9, no. 1, 2008.

[130]D. Moustakas, P. T. Lang, S. Pegg, E. Pettersen, I. Kuntz, N. Brooijmans, and R. Rizzo, “Development and Validation of a Modular, Extensible Docking Program: DOCK 5,” *J. Comput. Aided Mol. Des.*, vol. 20, no. 10–11, pp. 601–619, 2006.

[131]J. Ekanayake, S. Pallickara, and G. Fox, “MapReduce for Data Intensive Scientific Analyses,” in *Proceedings of the 2008 Fourth IEEE International Conference on eScience*, Washington, DC, USA, 2008, pp. 277–284.

[132]S. Brin and L. Page, “The Anatomy of a Large-scale Hypertextual Web Search Engine,” *Comput Netw ISDN Syst*, vol. 30, no. 1–7, pp. 107–117, Apr. 1998.

[133]G. Malewicz, M. H. Austern, A. J. . Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, “Pregel: A System for Large-scale Graph Processing,” in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 2010, pp. 135–146.

[134]J. C. Jacob, D. S. Katz, G. B. Berriman, J. Good, A. C. Laity, E. Deelman, C. Kesselman, G. Singh, M.-H. Su, T. A. Prince, and R. Williams, “Montage: A Grid Portal and Software Toolkit for Science-Grade Astronomical Image Mosaicking,” *CoRR*, 2010.

[135]D. R. Mathog, “Parallel BLAST on Split Databases,” *Bioinformatics*, vol. 19, no. 14, pp. 1865–1866, 2003.

[136]P. Maechling, E. Deelman, L. Zhao, R. Graves, G. Mehta, N. Gupta, J. Mehringer, C. Kesselman, S. Callaghan, D. Okaya, H. Francoeur, V. Gupta, Y. Cui, K. Vahi, T. Jordan, and E. Field, “SCEC CyberShake Workflows—Automating Probabilistic Seismic Hazard Analysis Calculations,” in *Workflows for e-Science*, I. Taylor, E. Deelman, D. Gannon, and M. Shields, eds. Springer, 2006, pp. 143–163.

[137]H. . Sorenson, *Kalman Filter: Theory and Applications*, vol. 38. IEEE Press, 1985.

[138]“Top Ten Exascale Research Challenges,” DOE, Feb. 2014, http://science.energy.gov/~/media/ascr/ascac/pdf/meetings/20140210/Top10reportFEB14.pdf.

[139]R. Thakur, “Parallel I/O Benchmarks, Applications, Traces,” May 2015. [Online]. Available: http://www.mcs.anl.gov/~thakur/pio-benchmarks.html.

[140]Z. Zhang and D. S. Katz, “Using Application Skeletons to Improve eScience Infrastructure,” in *10th IEEE International Conference on e-Science,* Sao Paulo, Brazil, October 20-24, 2014, pp. 111–118.

[141]J. Logan, S. Klasky, H. Abbasi, Q. Liu, G. Ostrouchov, M. Parashar, N. Podhorszki, Y. Tian, and M. Wolf, “Understanding I/O Performance Using I/O Skeletal Applications,” in *Euro-Par 2012 Parallel Processing*, vol. 7484, C. Kaklamanis, T. Papatheodorou, and P. Spirakis, eds. Springer Berlin Heidelberg, 2012, pp. 77–88.

[142]L. Meyer, M. Mattoso, M. Wilde, and I. Foster, “WGL – A Workflow Generator Language and Utility,” 2013. http://wiki.ci.uchicago.edu/pub/VDS/VDSDevelopment/WorkingDocuments/WorkflowGenReport.pdf

[143]E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M.-H. Su, K. Vahi, and M. Livny, “Pegasus: Mapping Scientific Workflows onto the Grid,” in *Across Grid Conference*, 2004.

[144]E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz, “Pegasus: A Framework for

Mapping Complex Scientific Workflows onto Distributed Systems," *Sci. Program.*, vol. 13, no. 3, pp. 219–237, 2005.

[145]M. Wilde, I. Foster, K. Iskra, P. Beckman, Z. Zhang, A. Espinosa, M. Hategan, B. Clifford, and I. Raicu, "Parallel Scripting for Applications at the Petascale and Beyond," *Computer*, vol. 42, no. 11, pp. 50–60, Nov. 2009.

[146]Y. Zhao, M. Hategan, B. Clifford, I. Foster, G. von Laszewski, V. Nefedova, I. Raicu, T. Stef-Praun, and M. Wilde, "Swift: Fast, Reliable, Loosely Coupled Parallel Computation," in *IEEE Congress on Services*, 2007.

[147]N. Chue Hong, "Open Software for Open Science," May 2015. http://figshare.com/articles/Open_Software_for_Open_Science/1424440

[148]D. M. Considine and G. D. Considine, *Van Nostrand's scientific encyclopedia; 5th ed.* New York, NY: Wiley Producer, 1999.

[149]D. Bailey, J. M. Borwein, O. Caprotti, U. Martin, B. Salvy, and M. Taufer, "Opportunities and Challenges in 21st Century Mathematical Computation," ICERM Workshop Report, July 2014.

[150]Y. He and C. Q. Ding, "Using Accurate Arithmetics to Improve Numerical Reproducibility and Stability in Parallel Applications," *J. Supercomput.*, vol. 18, no. 3, pp. 259–277, 2001.

[151]P. Balaji and D. Kimpe, "On the Reproducibility of MPI Reduction Operations," in *10th IEEE International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing, HPCC/EUC 2013,* Zhangjiajie, China, November 13-15, 2013, pp. 407–414.

[152]W.-F. Chiang, G. Gopalakrishnan, Z. Rakamaric, D. H. Ahn, and G. L. Lee, "Determinism and Reproducibility in Large-Scale HPC Systems," *Workshop Determinism Correctness Parallel Program. WoDet*, 2013.

[153]K. Yelick, "Algorithmic Challenges of Exascale Computing," presented at the Synchronization-reducing and Communication-reducing Algorithms and Programming Models for Large-scale Simulations Workshop, Brown University, Jan. 2012.

[154]N. Revol and P. Theveny, "Numerical Reproducibility of High-Performance computations Using Floating-Point or Interval Arithmetic," presented at the Challenges in 21st Century Experimental Mathematical Computation, Brown University, 21-Jul-2014.

[155]D. H. Bailey, "High-Precision Floating-Point Arithmetic in Scientific Computation," *Comput. Sci. Engg*, vol. 7, no. 3, pp. 54–61, May 2005.

[156]W. Kahan, "Pracniques: Further Remarks on Reducing Truncation Errors," *Commun ACM*, vol. 8, no. 1, p. 40–, Jan. 1965.

[157]M. Taufer, O. Padron, P. Saponaro, and S. Patel, "Improving NumericalReproducibility and Stability in Large-Scale Numerical smulations on GPUs," in *IEEE International Symposium on Parallel Distributed Processing (IPDPS),* 2010, pp. 1–9.

[158]D. Chapp, T. Johnston, M. Becchi, and M. Taufer, "On the Need for Reproducible Numerical Accuracy through Intelligent Runtime Selection of Reduction Algorithms at the Extreme Scale," presented at the IEEE Cluster Conference, Chicago, IL, USA, 2015.

[159]A. Thall, "Extended-Precision Floating-Point Numbers for GPU Computation," in *ACM SIGGRAPH 2006 Research Posters*, New York, NY, USA, 2006.

[160]J. Demmel and H. D. Nguyen, "Parallel Reproducible Summation," *IEEE Trans Comput.*, vol. 64, no. 7, pp. 2060–2070, 2015.

[161]A. Arteaga, O. Fuhrer, and T. Hoefler, "Designing Bit-Reproducible Portable High-Performance Applications," in *Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium*, Washington, DC, USA, 2014, pp. 1235–1244.

[162]J. L. Gustafson, *The End of Error: Unum Computing*. CRC Press, 2015.

[163]A. M. O. de Almeida, *Hamiltonian Systems: Chaos and Quantization*. Cambridge University Press, Cambridge, 1998.

[164]F. Jézéquel, P. Langlois, and N. Revol, “First Steps towards More Numerical Reproducibility,” *ESAIM Proc.*, vol. 45, pp. 229–238, Sep. 2014.

[165]P. Langlois, M. Martel, and L. Thévenoux, “Accuracy versus Time: A Case Study with Summation Algorithms,” in *Proceedings of the 4th International Workshop on Parallel Symbolic Computation, PASCO 2010,* July 21-23, 2010, Grenoble, France, pp. 121–130.