Serverless Workflows for Indexing Large Scientific Data

Conference Paper · December 2019

DOI: 10.1145/3366623.3368140

9 authors, including:

Tyler J. Skluzacek
University of Chicago
11 PUBLICATIONS **44** CITATIONS
SEE PROFILE

Ryan Chard
Argonne National Laboratory
47 PUBLICATIONS **461** CITATIONS
SEE PROFILE

Zhuozhao Li
University of Virginia
35 PUBLICATIONS **282** CITATIONS
SEE PROFILE

Yadu Babuji
University of Chicago and Argonne National Laboratory
27 PUBLICATIONS **136** CITATIONS
SEE PROFILE

Serverless Workflows for Indexing Large Scientific Data

Tyler J. Skluzacek
University of Chicago
skluzacek@uchicago.edu

Ryan Chard
Argonne National Laboratory
rchard@anl.gov

Ryan Wong
University of Chicago
rewong03@uchicago.edu

Zhuozhao Li
University of Chicago
zhuozhao@uchicago.edu

Yadu N. Babuji
University of Chicago
yadunand@uchicago.edu

Logan Ward
Argonne National Laboratory
lward@anl.gov

Ben Blaiszik
Argonne National Laboratory
bblaiszik@anl.gov

Kyle Chard
University of Chicago
chard@uchicago.edu

Ian Foster
Argonne & University of Chicago
foster@anl.gov

ABSTRACT

The use and reuse of scientific data is ultimately dependent on the ability to understand what those data represent, how they were captured, and how they can be used. In many ways, data are only as useful as the metadata available to describe them. Unfortunately, due to growing data volumes, large and distributed collaborations, and a desire to store data for long periods of time, scientific "data lakes" quickly become disorganized and lack the metadata necessary to be useful to researchers. New automated approaches are needed to derive metadata from scientific files and to use these metadata for organization and discovery. Here we describe one such system, Xtract, a service capable of processing vast collections of scientific files and automatically extracting metadata from diverse file types. Xtract relies on function as a service models to enable scalable metadata extraction by orchestrating the execution of many, short-running extractor functions. To reduce data transfer costs, Xtract can be configured to deploy extractors centrally or near to the data (i.e., at the edge). We present a prototype implementation of Xtract and demonstrate that it can derive metadata from a 7 TB scientific data repository.

CCS CONCEPTS

• **Information systems → Computing platforms**; *Search engine indexing*; Document structure; • **Applied computing → Document metadata**.

KEYWORDS

data lakes, serverless, metadata extraction, file systems, materials science

ACM Reference Format:
Tyler J. Skluzacek, Ryan Chard, Ryan Wong, Zhuozhao Li, Yadu N. Babuji, Logan Ward, Ben Blaiszik, Kyle Chard, and Ian Foster. 2019. Serverless Workflows for Indexing Large Scientific Data. In *5th Workshop on Serverless Computing (WOSC '19), December 9–13, 2019, Davis, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3366623.3368140

WOSC '19, December 9–13, 2019, Davis, CA, USA

ACM ISBN 978-1-4503-7038-7/19/12…$15.00
https://doi.org/10.1145/3366623.3368140

1 INTRODUCTION

Advances in scientific instruments, computational capabilities, and the proliferation of IoT devices have increased the volume, velocity, and variety of research data. As a result, researchers are increasingly adopting data-driven research practices, in which data are collected, stored for long periods of time, shared, and reused. Unfortunately, these factors create new research data management challenges, not least of which is the need for data to be adequately described for them to be generally useful. Without descriptive metadata, data may become unidentifiable, siloed, and in general, not useful to either the researchers who own the data or the broader scientific community. Unfortunately, organizing and annotating data is a time-consuming process and the gulf between data generation rates and the finite management capabilities of researchers continues to grow. To increase the value and usefulness of scientific data, new methods are required to automate the extraction and association of rich metadata that describe not only the data themselves, but also their structure and format, provenance, and administrative information.

Data lakes have become a popular paradigm for managing large and heterogeneous data from various sources. A data lake contains a collection of data in different formats, accompanied by metadata describing those data. Unfortunately, the data deluge and the desire to store all data for eternity can quickly turn a data lake into a "data swamp" [18]—a term used to describe the situation in which the data stored in the data lake lack the metadata necessary to be discoverable, understandable, and usable. Without automated and scalable approaches to derive metadata from scientific data, the utility of these data are reduced. This problem is especially prevalent in science as datasets can be enormous (many petabytes), are created and shared by dynamic collaborations, are often collected under tight time constraints where data management processes become afterthoughts, and for which preservation is important for purposes of reproducibility and reuse.

Extracting metadata from scientific data is a complex task. Scientific repositories may exceed millions of files and petabytes of data; data are created at different rates, by different people; and there

exist an enormous number of data formats and conventions. While metadata can be rapidly extracted from some data types, others, such as images and large hierarchical file formats can require the use of multiple extraction methods. As a result, the metadata extraction process must be scalable to process large numbers of files, flexible to support different extraction methods, and extensible to be applied to various scientific data types and formats.

Serverless computing, and in particular function as a service (FaaS), provides an ideal model for managing the execution of many short-running extractors on an arbitrarily large number of files. Serverless computing abstracts computing resources from the user, enabling the deployment of applications without consideration for the physical and virtual infrastructure on which they are hosted. FaaS allows users to register programming functions with predefined input signatures. Registered functions can subsequently be invoked many times without the need to provision or scale any infrastructure.

In this paper we propose the use of FaaS for mapping the metadata extraction problem to a collection of granular metadata extractor functions. We describe how such a model can support the flexibility, scalability, and extensibility required for scientific metadata extraction. Rather than rely on commercial FaaS systems, we use a distributed FaaS model that overcomes the limitation of moving large amounts of data to the cloud. Instead, we are able to push metadata extractors to the edge systems on which the scientific data reside.

Our prototype system, Xtract, provides high-throughput and on-demand metadata extraction that enables the automated creation of rich, searchable data lakes from previously unsearchable data swamps. Xtract implements dynamic metadata extraction workflows comprised of serverless functions that may be executed in the cloud or at the edge. Xtract runs as a centralized management service that orchestrates the crawling, transfer (when necessary), and execution of metadata extraction functions. Xtract uses the *funcX* serverless supercomputing platform [5] to execute functions across diverse and distributed computing infrastructure. We evaluate Xtract's performance by extracting metadata from materials science data stored in a 7 TB subset of the Materials Data Facility (MDF) [3, 4].

The remainder of this paper is organized as follows. §2 outlines example scientific data lakes. §3 describes Xtract's serverless architecture and presents the set of available extractors. §4 provides initial results of system performance on hundreds of thousands of scientific files. Finally, §5 and §6 present related work and concluding remarks, respectively.

2 SCIENTIFIC DATA LAKES

Data lakes are schemaless collections of heterogeneous files obtained from different sources. Unlike traditional data warehouses, data lakes do not require upfront schema integration and instead allow users to store data without requiring complex and expensive Extract-Transfer-Load (ETL) pipelines. This low barrier to data archival encourages the storage of more bytes and types of data. However, the lack of a well-defined schema shifts responsibility from upfront integration to descriptive metadata to allow users to search, understand, and use the data. To motivate our work we briefly describe three scientific data lakes below.

The *Carbon Dioxide Information Analysis Center (CDIAC)* collected an emissions dataset from the 1800s through 2017. The dataset contains more than 500 000 files (330+ GB) with over 10 000 unique file extensions. The archive contains little descriptive metadata and includes a number of irrelevant files, such as as debug-cycle error logs and Windows desktop shortcuts. The data are currently being moved to the Environmental System Science Data Infrastructure for a Virtual Ecosystem (ESS-DIVE) archive [9]. In prior work we extracted metadata from files in this repository and created a data lake for users [17, 18].

DataONE [12] provides access to a distributed network of biological and environmental sciences data repositories. DataONE manages a central index across these distributed repositories, enabling users to discover datasets based on metadata queries. Member data repositories provide dataset- and file-level metadata to DataONE. As of May 2019, DataONE contains over 1.2 million files and 809 000 unique metadata entries.

The *Materials Data Facility (MDF)* [3, 4] is a centralized hub for publishing, sharing, and discovering materials science data. The MDF stores many terabytes of data from many different research groups, covering many disciplines of materials science, and with a diverse range of file types. The downside of the expansive range of materials data held by the MDF is that it can be difficult for users to find data relevant to their science. The MDF reduces the "data discovery" challenge by hosting a search index that provides access to metadata from the files (e.g., which material was simulated in a certain calculation). The data published by the MDF is primarily stored on storage at the National Center for Supercomputing Applications (NCSA) and is accessible via Globus.

3 XTRACT

Xtract is a metadata extraction system that provides on-demand extraction from heterogeneous scientific file formats. Xtract can operate in one of two modes: centralized or edge metadata extraction. In the centralized mode, Xtract processes files stored on a Globus endpoint by first staging them to a centralized location and then executing metadata extraction pipelines on those files. In the edge mode, Xtract can execute metadata extraction pipelines on edge computers near the data by deploying a collection of metadata extraction functions to distributed FaaS endpoints.

In order to extract metadata, Xtract applies various extractors—functions that take a file as input and generate a JSON document of metadata for that file. In either centralized or edge mode, Xtract assembles a pipeline of extraction functions based on file contents and metadata extracted from other extractors. Xtract begins this process by applying a file type extractor which informs selection of subsequent extractors. Subsequent extractors are selected based on their expected metadata yield. This allows Xtract to apply the appropriate extractors to a specific file.

Xtract creates a metadata document for each file it processes. When an extractor is applied to a file, Xtract appends the extractor's output metadata to the file's metadata document. Once all applicable extractors are applied to a file, Xtract loads this metadata document into a Globus Search index [1].

3.1 Extractors

Xtract includes extractors for many file types commonly used in science. Each extractor is implemented as either a Python function or Bash script. The remainder of this section outlines Xtract's library of extractors and the data types they have been designed to process.

The universal extractor extracts generic file information such as file extension, path, and size. It also computes an MD5 hash (for duplicate detection). The universal extractor is typically the first applied to a file.

The file type extractor applies a machine learning model to infer the type of a file. This information is crucial for determining which downstream extractors could yield metadata from that file. Users can opt to use a pre-trained model, or to automatically train one on their data. In the latter case, training commences by selecting $n\%$ of the files in the repository at random and running those files through all other Xtract extractors. We record whether or not the extractor produces metadata and build a $file \rightarrow extractor$ label mapping based on those that yield metadata. It then trains a random forests model using these labels and the first 512 bytes of the file as features. The primary goal of this extractor is to save time—the amount of time to incorrectly apply metadata extractors to a file can take seconds, whereas predicting which extractors will likely produce metadata via the byte footprint of a file is tens of milliseconds.

The tabular extractor extracts metadata from files with strict row-column structure, such as *.csv* and *.tsv* files. It first extracts the delimiter, location of the column-labels or header, and applies binary search over the file to identify the existence and location of a free text preamble. The tabular extractor then processes the columns in parallel to collect aggregate information (means, medians, modes). The free text preamble is re-queued for separate processing by the keyword extractor.

The keyword extractor identifies uniquely descriptive words in unstructured free text documents such as READMEs, academic papers (e.g., *.pdf*, *.doc*), and abstracts. The keyword extractor uses word embeddings to curate a list of the top-n keywords in a file, and an associated weight corresponding to the relative relevance of a given keyword as a proper descriptor for that document.

The semi-structured extractor takes data or pre-existing metadata in semi-structured formats (e.g., *.json* or *.xml*) and returns a metadata summary of the document, such as the maximum nesting depth and the types of data represented at each level (e.g., structured, unstructured, or lists). Furthermore, free text fields are isolated and re-queued for processing by the keyword extractor.

The hierarchical extractor processes hierarchical HDF5 files (and HDF5-based file formats such as NetCDF) commonly used in science. The hierarchical extractor uses HDF5 libraries to extract both the self-describing, in-file metadata as well as metadata about various dimensions of the data.

The image extractor utilizes an SVM trained on a manually labeled set of over 600 images to derive the class of an image, which is useful for both downstream extractors and general file categorization. The image classes include scientific graphics (e.g., Figure 1), geographic maps, map plots (i.e., geographic maps with an interesting color index), photographs, and scientific plots (e.g., Figure 3). The features for this model include a color histogram, a

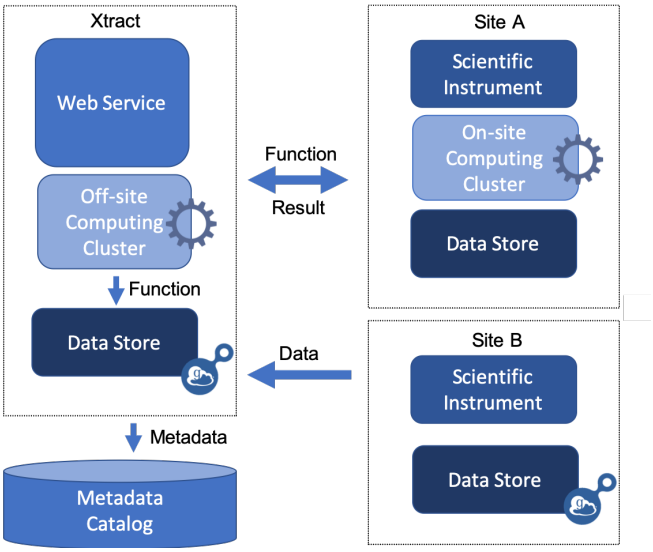

Figure 1: Overview of the Xtract architecture. For *Site A*, functions are transmitted to the remote resource and performed on local computing resources, returning metadata to the Xtract service. *Site B* lacks suitable local computing capabilities, requiring data to be staged to Xtract for analysis.

standard grayscaled version of the original image, and the image size. Further, the Xtract library also contains the downstream **map extractor** that can isolate geographic entities from a photograph of a map.

The materials extractor provides a thin wrapper over MaterialsIO [8], a metadata extractor designed to process common file formats used in materials science. MaterialsIO contains file and file-group parsers for atomistic simulations, crystal structures, density functional theory (DFT) calculations, electron microscopy outputs, and images. If the Xtract sampler classifies a file as a materials file, the materials extractor is invoked, launching each parser at the contents of a directory.

3.2 Prototype Implementation

Xtract is implemented as a service via which users can submit requests to extract metadata from a collection of files. Xtract first crawls the specified files and determines an initial set of extractors to apply to them. As outlined above, the extractors may be executed either centrally on the Xtract server or remotely alongside the data. As processing continues, Xtract assembles a metadata document for each file and dynamically selects other extractors to apply.

Xtract is deployed on Amazon Web Services (AWS) and makes use of various services. The main Xtract service is deployed on an AWS Elastic Compute Cloud instance. Xtract manages state in an AWS Relational Database Service (RDS) instance. Each extraction request is stored in the database and the state is updated throughout the extraction process. Xtract is able to send the derived metadata to an external metadata catalog such as a Globus Search index. The Xtract architecture is shown in Figure 1.

3.2.1 Metadata Extractors. Xtract is designed to execute its extractors centrally or on edge storage systems near to where data are stored. Our implementation uses the *funcX* [5] FaaS platform to deploy and run extractors. *funcX* is specifically designed to integrate with research computing cyberinfrastructure and enable a FaaS execution interface. *funcX* builds upon the Parsl [2] parallel programming library to manage the execution of functions in containers on arbitrary compute resources. *funcX* enables Xtract to execute metadata extraction functions at any registered and accessible *funcX* endpoint. We deploy the prototype with endpoints located both on the central service and at the edge to enable both centralized and edge extraction.

Each metadata extractor and its dependencies are wrapped in a Docker container so that it can be executed on heterogeneous compute environments. We have published each extractor container to *funcX* and registered a *funcX* function for each extractor. The function is responsible for invoking the extractor and returning the resulting metadata as a JSON dictionary.

funcX enables Xtract to reliably scale to thousands of nodes and deploy metadata extraction tasks on arbitrary computing resources. Xtract can make use of any accessible *funcX* endpoint to process data at the edge, sending extractor codes and containers to the *funcX* endpoint for execution. In addition, *funcX* supports Singularity and Shifter, allowing extractors to be executed on various high performance computing systems.

3.2.2 Data Staging. Irrespective of the extraction mode, either centralized or edge, the data to be processed must be available to the deployed extractor container. In the case where data cannot be directly accessed within the container (e.g., where the container does not mount the local filesystem), data are dynamically staged for processing. Each container includes Xtract tooling to stage data in and out of itself. We use Globus as the basis for data staging, using Globus HTTPS requests, to securely download remote data into the container.

3.2.3 Security Model. Xtract implements a comprehensive security model using Globus Auth [21]. All interactions with the Xtract Web service are secured with Globus Auth. Users can authenticate with Xtract using one of several hundred identity providers, including many institutions. Xtract uses Globus Auth OAuth 2 flows to stage data on behalf of authenticated users. Xtract first verifies a user identity, requests an access token to perform data transfers on their behalf, and then uses Globus to stage data from remote storage to the Xtract extractor. Finally, the resulting metadata are published into the search index using a Globus Search access token. The search index is configured with a *visible_to* field, restricting discovery to authenticated users.

4 EVALUATION

We evaluate Xtract by extracting metadata from more than 250 000 files stored in MDF. We deployed the Xtract service on an AWS EC2 t2.small instance (Intel Xeon; 1 vCPU; 2 GB memory) and deployed a private *funcX* endpoint on ANL's PetrelKube—a 14-node Kubernetes cluster. The MDF data are stored on the Petrel data service, a Globus-managed 3 PB data store at ANL. While Petrel and PetrelKube are located within the same network, they do not share a file system. Thus, when executing extractors close to the data we still stage the data from Petrel to PetrelKube for processing.

In this section we first evaluate Xtract's performance by crawling all files on the MDF as a means of initializing the downstream metadata extraction workflow and providing summary statistics about the data. We next profile the downstream metadata extractor functions on hundreds of thousands of heterogeneous files in MDF. Finally, we illustrate the performance of batching multiple files into one extractor function across multiple representative file types.

4.1 Crawling Performance

First we crawl MDF to identify and record file locations, as well as general attributes about each file such as size and extension. Xtract manages the crawling process from its central service, employing a remote breadth-first search algorithm on the directory via Globus. In processing MDF, Xtract crawled each of the 2.2 million files in approximately 5.2 hours—at an effective rate of 119 files crawled per second. As part of crawling, Xtract generated descriptive treemaps about general attributes of the data. One such treemap illustrating the proportion of the most common extensions in MDF is shown in Figure 2. Here we observe that atomistic structure (*.xyz*), unknown (*nan*), and image files (*.tiff/.tif*) are most common in MDF relative to other types.

4.2 File Type Training

We next evaluate the time taken to perform the optional model training step for the file type extractor. Automated training of this model occurs by trying every extractor on each file in a 5-10% subset of the total data set, denoting the first extractor that returns serviceable metadata without error. This extractor represents the file's label and the first 512 bytes represent the features for the random forests model. Xtract conducted such automated training on 110 900 MDF files. The entire label creation, feature collection, and training workflow took approximately 5.3 hours. We found that label generation constitutes a majority of this time, as feature generation and model training total just 45 seconds. It is important to note that, in the future, increasing the number of PetrelKube pods concurrently serving feature label-collection functions can drastically reduce the time taken to train the model.

4.3 Extractor Performance

We next evaluate the performance of Xtract's extractors by invoking extraction functions on MDF's data. We process a variety of file types including all available tabular and structured files and at least 25 000 of each other type of file, selected randomly from MDF. The performance of each extractor is summarized in Table 1.

We observe that a majority of extractor function invocations finish within milliseconds, and a majority of a file's round-trip processing time occurs due to file staging. This observation exemplifies the need to invoke functions near to data, directly mounted to the file system housing the data, whenever possible. Moreover we note that a few dozen large hierarchical files, exceeding 10 GB in size, were not processed due to ephemeral storage constraints on PetrelKube.

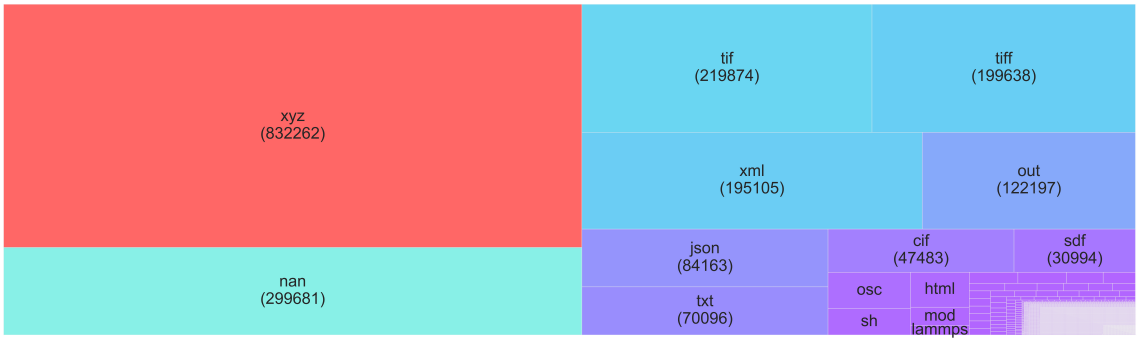

Figure 2: A treemap of the MDF extension frequency. The proportion of the size of each box relative to the entire treemap is equivalent to the proportion of the frequency of that file extension in MDF out of the 2.2 million total files

Table 1: Extractor performance.

Extractor	# Files	Avg. Size (MB)	Avg. Extract Time (ms)	Avg. Stage Time (ms)
File Type	25,132	1.52	3.48	714
Images	76,925	4.17	19.30	1,198
Semi-structured	29,850	0.38	8.97	412
Keyword	25,997	0.06	0.20	346
Materials	95,434	0.001	24	1,760
Hierarchical*	3,855	695	1.90	9,150
Tabular	1,227	1.03	113	625

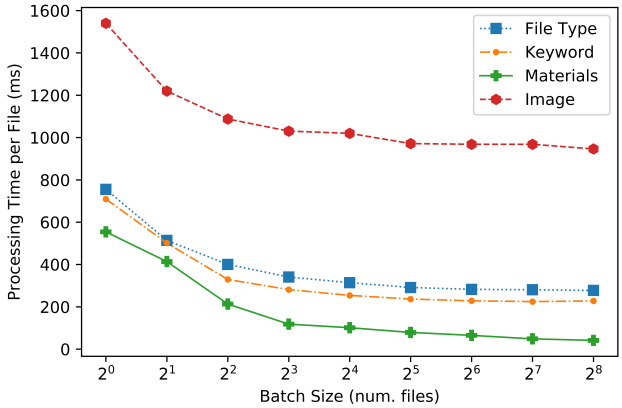

Figure 3: Batching: extraction time per file (ms) on batches sized 1-256 for representative files processed by the file type, image, keyword, and materials extractors

Finally, we explore the benefits of batching multiple files into one function invocation request. We choose the four most applicable extractors: file type, keyword, images and materials. We randomly select representative files of all four types, each within 2% of the average file sizes shown in Table 1. We implement batching by aggregating 1-256 files into a single Xtract request and have modified the *funcX* function to download and process files in parallel. Figure 3 shows that the average time required to process a file decreases as the batch size increases. Thus, batching can increase the performance of metadata extraction workflows, especially those requiring file transfers.

5 RELATED WORK

Xtract builds upon a large body of work in both metadata extraction systems and serverless computing. Xtract is not the first to provide a scalable solution to extract metadata from datasets. Pioneering research on data lakes developed methods for extracting standard metadata from nonstandard file types and formats [20]. Most data lakes are designed with a specific target domain for which they are optimized, whether they primarily focus on transactional, scientific, or networked data. Xtract is designed to be easily extensible and therefore can be easily applied to different domains.

We [23] and others [7, 10, 14, 22] have created systems to manage metadata catalogs that support the organization and discovery of research data. However, these approaches typically require that users provide metadata and that curators continue to organize data over time. A number of systems exist for automatically extracting metadata from repositories. For example, ScienceSearch [15] uses machine learning techniques to extract metadata from a dataset served by the National Center for Electron Microscopy (NCEM). Most data in this use case are micrograph images, but additional contextual metadata are derived from file system data and free text proposals and publications. Like Xtract, ScienceSearch provides a means for users to extensibly switch metadata extractors to suit a given dataset. Brown Dog [13] is an extensible metadata extraction platform, providing metadata extraction services for a number of disciplines ranging from materials science to social science. Unlike Xtract, Brown Dog requires that files are uploaded for extraction. The Apache Tika toolkit [11] is an open-source content and metadata extraction library. Tika has a robust parser interface in which users can create and employ their own parsers in metadata extraction workflows. While Apache Tika has parsers that support thousands of file formats, the automated parser-to-file mapping

utilizes MIME types to find suitable parsers for a file, which is often misleading for many scientific data use cases. Xtract could be extended to support Tika extractors and to enable execution on a serverless platform.

While most related research performs metadata extraction to enable search, Xtract-like systematic sweeps across repositories can also be used for analysis. For example, the Big Data Quality Control (BDQC) framework [6] sweeps over large collections of biomedical data without regard to their meaning (domain-blind analysis) with the goal of identifying anomalies. BDQC employs a pipeline of extractors to derive properties of imaging, genomic, and clinical data. While BDQC is implemented as a standalone system, the approach taken would be similarly viable in Xtract.

6 CONCLUSION

The growing volume, velocity, and variety of scientific data is becoming unmanageable. Without proper maintenance and management, data lakes quickly degrade into disorganized data swamps, lacking the necessary metadata for researchers to efficiently discover, use, and repurpose data. The growing size and heterogeneity of scientific data makes extracting rich metadata a complex and costly process, requiring a suite of customized extractors and advanced extraction techniques. We have described a serverless-based approach for metadata extraction, called Xtract. Xtract enables the scalable extraction of metadata from large-scale and distributed data lakes, in turn increasing the value of data. We showed that our prototype can crawl and process hundreds of thousands of files from a multi-terabyte repository in hours, and that batching files and parallelizing file staging and extraction tasks can improve the performance of metadata extraction times.

In future work [16] we are focused on scaling the Xtract model and exploring the use of Xtract on larger and globally distributed datasets. We will investigate strategies for guiding extractor placement across scientific repositories, weighing data and extractor transfer costs to optimize placement. Finally, we will extend Xtract to facilitate the integration of custom metadata extractors.

ACKNOWLEDGMENTS

This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. We gratefully acknowledge the computing resources provided and operated by the Joint Laboratory for System Evaluation (JLSE) at Argonne National Laboratory as well as the Jetstream cloud for science and engineering [19].

REFERENCES

[1] Rachana Ananthakrishnan, Ben Blaiszik, Kyle Chard, Ryan Chard, Brendan McCollam, Jim Pruyne, Stephen Rosen, Steven Tuecke, and Ian Foster. 2018. Globus platform services for data publication. In *Proceedings of the Practice and Experience on Advanced Research Computing*. ACM, 14.

[2] Yadu Babuji, Anna Woodard, Zhuozhao Li, Daniel S Katz, Ben Clifford, Rohan Kumar, Lukasz Lacinski, Ryan Chard, Justin M Wozniak, Ian Foster, et al. 2019. Parsl: Pervasive parallel programming in python. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*. ACM, 25–36.

[3] Ben Blaiszik, Kyle Chard, Jim Pruyne, Rachana Ananthakrishnan, Steven Tuecke, and Ian Foster. 2016. The Materials Data Facility: Data services to advance materials science research. *JOM* 68, 8 (2016), 2045–2052.

[4] Ben Blaiszik, Logan Ward, Marcus Schwarting, Jonathon Gaff, Ryan Chard, Daniel Pike, Kyle Chard, and Ian Foster. 2019. A Data Ecosystem to Support Machine Learning in Materials Science. (apr 2019). arXiv:1904.10423 http://arxiv.org/abs/1904.10423

[5] Ryan Chard, Tyler J Skluzacek, Zhuozhao Li, Yadu Babuji, Anna Woodard, Ben Blaiszik, Steven Tuecke, Ian Foster, and Kyle Chard. 2019. Serverless Supercomputing: High Performance Function as a Service for Science. *arXiv preprint arXiv:1908.04907* (2019).

[6] Eric Deutsch, Roger Kramer, Joseph Ames, Andrew Bauman, David S Campbell, Kyle Chard, Kristi Clark, Mike D'Arcy, Ivo Dinov, Rory Donovan, et al. 2018. BDQC: a general-purpose analytics tool for domain-blind validation of Big Data. *bioRxiv* (2018), 258822.

[7] MP Egan, SD Price, KE Kraemer, DR Mizuno, SJ Carey, CO Wright, CW Engelke, M Cohen, and MG Gugliotti. 2003. VizieR Online Data Catalog: MSX6C Infrared Point Source Catalog. The Midcourse Space Experiment Point Source Catalog Version 2.3 (October 2003). *VizieR Online Data Catalog* 5114 (2003).

[8] Materials Data Facility. 2019. MaterialsIO. https://github.com/materials-data-facility/MaterialsIO.

[9] Environmental Systems Science Data Infrastructure for a Virtual Ecosystem. 2019. ESS-DIVE. https://ess-dive.lbl.gov/.

[10] Gary King. 2007. An introduction to the dataverse network as an infrastructure for data sharing.

[11] Chris Mattmann and Jukka Zitting. 2011. *Tika in action*. Manning Publications Co.

[12] William Michener, Dave Vieglais, Todd Vision, John Kunze, Patricia Cruse, and Greg Janée. 2011. DataONE: Data Observation Network for Earth—Preserving data and enabling innovation in the biological and environmental sciences. *D-Lib Magazine* 17, 1/2 (2011), 12.

[13] Smruti Padhy, Greg Jansen, Jay Alameda, Edgar Black, Liana Diesendruck, Mike Dietze, Praveen Kumar, Rob Kooper, Jong Lee, Rui Liu, et al. 2015. Brown Dog: Leveraging everything towards autocuration. In *2015 IEEE Int'l Conference on Big Data (Big Data)*. IEEE, 493–500.

[14] Arcot Rajasekar, Reagan Moore, Chien-yi Hou, Christopher A Lee, Richard Marciano, Antoine de Torcy, Michael Wan, Wayne Schroeder, Sheau-Yen Chen, Lucas Gilbert, et al. 2010. iRODS primer: integrated rule-oriented data system. *Synthesis Lectures on Information Concepts, Retrieval, and Services* 2, 1 (2010), 1–143.

[15] Gonzalo P Rodrigo, Matt Henderson, Gunther H Weber, Colin Ophus, Katie Antypas, and Lavanya Ramakrishnan. 2018. ScienceSearch: Enabling search through automatic metadata generation. In *2018 IEEE 14th Int'l Conference on e-Science (e-Science)*. IEEE, 93–104.

[16] Tyler J. Skluzacek. 2019. Dredging a Data Lake: Decentralized Metadata Extraction. In *Middleware '19: 20th International Middleware Conference Doctoral Symposium (Middleware '19)*. ACM, New York, NY, USA, 3. https://doi.org/10.1145/3366624.3368170

[17] Tyler J Skluzacek, Kyle Chard, and Ian Foster. 2016. Klimatic: a virtual data lake for harvesting and distribution of geospatial data. In *2016 1st Joint Int'l Workshop on Parallel Data Storage and data Intensive Scalable Computing Systems (PDSW-DISCS)*. IEEE, 31–36.

[18] Tyler J Skluzacek, Rohan Kumar, Ryan Chard, Galen Harrison, Paul Beckman, Kyle Chard, and Ian Foster. 2018. Skluma: An extensible metadata extraction pipeline for disorganized data. In *2018 IEEE 14th Int'l Conference on e-Science (e-Science)*. IEEE, 256–266.

[19] Craig A Stewart, Timothy M Cockerill, Ian Foster, David Hancock, Nirav Merchant, Edwin Skidmore, Daniel Stanzione, James Taylor, Steven Tuecke, George Turner, et al. 2015. Jetstream: a self-provisioned, scalable science and engineering cloud environment. In *Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure*. ACM, 29.

[20] Ignacio G Terrizzano, Peter M Schwarz, Mary Roth, and John E Colino. 2015. Data Wrangling: The Challenging Yourney from the Wild to the Lake.. In *CIDR*.

[21] Steven Tuecke, Rachana Ananthakrishnan, Kyle Chard, Mattias Lidman, Brendan McCollam, Stephen Rosen, and Ian Foster. 2016. Globus Auth: A research identity and access management platform. In *2016 IEEE 12th Int'l Conference on e-Science (e-Science)*. IEEE, 203–212.

[22] Danielle Welter, Jacqueline MacArthur, Joannella Morales, Tony Burdett, Peggy Hall, Heather Junkins, Alan Klemm, Paul Flicek, Teri Manolio, Lucia Hindorff, et al. 2013. The NHGRI GWAS Catalog, a curated resource of SNP-trait associations. *Nucleic Acids Research* 42, D1 (2013), D1001–D1006.

[23] J. M. Wozniak, K. Chard, B. Blaiszik, R. Osborn, M. Wilde, and I. Foster. 2015. Big Data Remote Access Interfaces for Light Source Science. In *2nd IEEE/ACM International Symposium on Big Data Computing (BDC)*. 51–60.