A Lightweight Framework for Research Data Management

Extended Abstract

Dimitar Nikolov
Indiana University Research Technologies
Bloomington, Indiana
dnikolov@iu.edu

Esen Tuna
Indiana University Research Technologies
Bloomington, Indiana
metuna@iu.edu

ABSTRACT

We describe a framework for managing live research data involving two major components. First, a system for the scalable scheduling and execution of automated policies for moving, organizing, and archiving data. Second, a system for managing metadata to facilitate curation and discovery with minimal change to existing workflows. Our approach is guided by four main principles: 1) to be non-invasive and to allow for easy integration into existing workflows and computing environments; 2) to be built on established, cloud-aware, open-source tools; 3) to be easily extensible and configurable, and thus, adaptable to different academic disciplines; and 4) to integrate with and take advantage of infrastructure and services available on academic campuses and research computing environments. These principles give our solution a well-defined place along the spectrum of research data management software such as sophisticated electronic lab notebooks and science gateways. Our lightweight and flexible data management framework provides for curation and preservation of research data within a lab, department or university cyberinfrastructure.

CCS CONCEPTS

• **Applied computing → Document management and text processing**; *Document metadata*; *Document searching*; • **Information systems → Data management systems**.

KEYWORDS

data management, data curation, metadata management, data policies

ACM Reference Format:
Dimitar Nikolov and Esen Tuna. 2019. A Lightweight Framework for Research Data Management: Extended Abstract. In *Practice and Experience in Advanced Research Computing (PEARC '19), July 28-August 1, 2019, Chicago, IL, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3332186.3333157

1 INTRODUCTION

The success of data science, machine learning and similar disciplines in recent years has driven a sharp increase in the application of computational and data-intensive methods to many fields of science. As a result, academic labs, departments, and IT providers on university campuses have to adapt and respond with cyberinfrastructure for managing the large volumes of data being produced. One class of cyberinfrastructure services is software systems for curation and discovery of *live* research data. Such data is actively used in research activities, and does not necessarily constitute the final product of the research, or *published* data.

The heterogeneity of live data and the tools and platforms for analyzing it across different fields are significant challenges in creating robust, general-purpose data management systems. On the ad-hoc spectrum of data management solutions, a research group might utilize a set of available university services, such as for file storage and document sharing, and adopt conventions for handling the data. This is a non-invasive approach as it keeps the abstraction level low, and handles the data in its original form. In addition, it does not interfere with workflows around existing computing environments and analysis tools. This strategy, however, is not scalable to large teams or large data volumes, and can lead to inconsistencies, omissions, and duplication in the storing and preservation of data, which hinders research output, reproducibility and open science. For example, failing to keep copies of original data as it was collected from sensors or instruments, or failing to keep track of process parameters, may lead to publication retraction. In case of funded research, not meeting research agency requirements in data management plans may have funding implications.

On the sophisticated end of the spectrum of data management systems, shared electronic lab notebooks (ELNs) [1] can be deployed as a data management solution to organize the activities of a research group into a single system. ELNs can be deployed on-premises or accessed on the cloud, and are a user-friendly, but less customizable solution for data management. They require the adaptation of existing workflows, so their input and output data can be captured in the ELN. Another class of sophisticated services for managing live data is science gateways [6] — highly-configurable middleware usually delivered through a web interface for organizing a variety of university resources for storage and computation. Science gateways can be tailored more specifically to a research group's workflows, but require significant expertise to deploy and operate. In addition, their purpose and scope extend beyond data management to the utilization of high-performance computing and other infrastructure frequently found in academic environments.

In this paper, we describe a data management framework for cases when ELNs and science gateways represent too significant a change in existing workflows to be pracical. The framework is based on open-source tools widely used in industry and can scale

PEARC '19, July 28-August 1, 2019, Chicago, IL, USA

ACM ISBN 978-1-4503-7227-5/19/07.
https://doi.org/10.1145/3332186.3333157

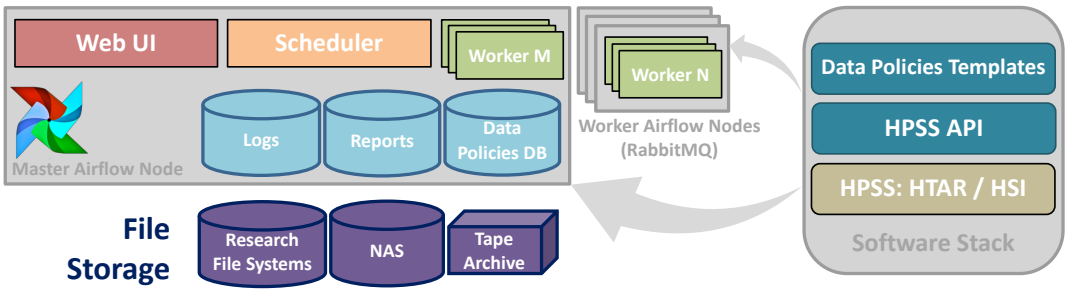

Figure 1: Architecture for the service that handles the definition, scheduling and execution of data policies.

to a wide range of workflows. It complements data management features of ELNs and science gateways by providing:

- Annotation of file system objects with metadata, the schema of which may need to change frequently over time or based on the type of data.
- Interfaces for intelligent search and browsing of file system objects and their accompanying metadata.
- Definition, scheduling and execution of automated data policies for moving data between different storage environments, or applying transformations to the data.

The system satisfies these requirements without placing significant burden on the users to use specific interfaces for ingesting and staging out data by using a combination of file system monitoring and regularly scheduled tasks. Thus, the system overcomes some of the deployment and workflow adjustments required by ELNs and science gateways.

In summary, the framework we describe makes the following contributions:

- It provides a lightweight model and tools for data management that is non-invasive and integrates into existing workflows and computing environments.
- It provides a model for integrating industry-tested, cloud-aware tools with academic infrastructure such as tape archives, HPSS [5], and Globus [2].
- It is customizable and configurable with popular tools and formats, such as Python and JSON.

2 ARCHITECTURE

The framework consists of distinct services for data policies and metadata management, which can be integrated according to the specific needs of a particular organization.

2.1 Data Policies

The architecture of the service for defining, scheduling and executing data policies is shown in Figure 1. The service is based on Apache Airflow [3] — an open-source platform used widely in industry for authoring workflows. In Airflow, workflows are defined as directed acyclic graphs (DAGs) consisting of one or more dependent tasks. A given data policy can be implemented by one or more DAGs. The DAGs are defined as Python code and are scheduled and executed by a scheduler. The schedules are specified in *crontab* format. A web server delivers a user interface (UI) for inspecting the DAG code and schedule, monitoring execution status and examining the logs. The scheduler executes each of a DAG's tasks in a separate process, which can reside on the master node, or the tasks can be distributed over multiple worker nodes via an asynchronous message queue such as RabbitMQ. In addition to the scheduler, the service consists of APIs and templates for data policies that interface with cyberinfrastructure available on university campuses. For example, we have written a Python API on top of the High Performance Storage System (HPSS) and its suite of tools for high-performance transfer to and from tape archive systems. DAG templates for common activities such as archiving, backup, and cleanup allow for the deployment of these policies after minimal customization. Such policies can be applied to a variety of storage environments over SAMBA and NFS protocols — from centralized research file systems (GPFS) and tape archives (HPSS) to lab and department resources like network-attached storage devices.

We chose this architecture over a more monolithic system because it allows us to use tools that are well-suited to their specific tasks, while taking advantage of expertise that already exists in academic cyberinfrastructure environments. Researchers can benefit from the extensive logging and user-friendly interface for monitoring DAGs provided by Airflow. IT specialists can scale the performance of the system as the needs of the research group change. The ability to define data policies as DAGs using an accessible and widely used programming language like Python enables researchers in addition to IT specialists to author data policies. Finally, unlike systems like iRODS [4], this architecture does not use an abstraction layer over the data storage systems and data can be handled in its native form, thus allowing researchers to keep their existing workflows unchanged.

2.2 Metadata Management

The architecture of the metadata management service is shown in Figure 2. At the core of the system are two RESTful APIs for file system and metadata manipulation. The file system API allows for manipulating the contents of the file system or downloading files remotely. Access to file manipulation functions can be disabled to comply with security protocols at each deployment environment. The metadata API allows the creation, editing, deletion and searching of metadata entries associated with file objects. The user interacts with the system through a mobile-friendly web UI, but

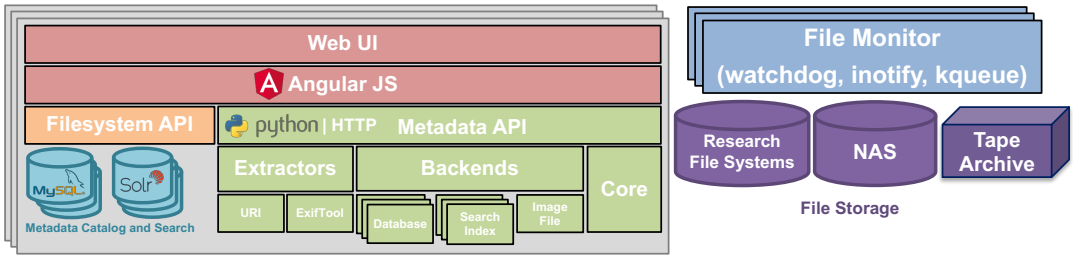

Figure 2: Architecture for the metadata management service.

the decoupling of the UI from the APIs means that other interfaces, such as command-line, can be integrated into the system.

The metadata API is meant to be extensible to accommodate a variety of data management needs across fields. Metadata is attached to file system objects in the form of JSON strings that are stored in a catalog. The metadata is also indexed in a Solr engine, which facilitates intelligent search and faceted browsing. Also part of the metadata API are metadata extractors, which are configured per file type or location using regular expressions. Additional extractors can be added to the system by extending a simple Python interface. Auto-extraction of metadata is triggered either through the UI by a user, or by the file monitor service installed at each storage system being monitored by the metadata manager. The file monitor detects changes to the file system and uses the metadata API to update the metadata catalog and the search engine. Thus, when a file is moved, its metadata is reassigned to the correct file system object. When a new file enters the system, the relevant auto-extractor is triggered, and when a file is deleted, the metadata entries for it are removed. This combination of auto-extraction and file monitoring is transparent to the users and alleviates the necessity to manually enter and maintain the metadata in the vast majority of cases. It also allows the users to complete the majority of their file manipulation through familiar OS interfaces, while only using the metadata manager UI in specific cases when metadata needs to be modified or searched.

The metadata service is customizable in a number of additional ways. It makes use of different storage back-ends, so that a range of database storage systems can be supported for the metadata catalog. In addition, the metadata service allows specification of extraction rules based on regular expressions to accommodate naming conventions that research labs can use to automate metadata extraction.

2.3 Integration

The data policies and metadata services can be deployed separately or integrated more closely. For example, the metadata API can be accessed from DAGs that implement an archiving policy based on metadata fields. The use of RESTful API and a flexible JSON schema for metadata makes this possible without any further development on the metadata service. The structure of the data policy service also does not change for such an integration, but only the addition of a new policy (DAG) is required.

3 DEPLOYMENT AND IMPLEMENTATION

The data management framework arose from many conversations with researchers about their specific needs and workflows. As such, it was important that we could receive and integrate feedback frequently throughout the development process. To this end, we have automated the deployment of the different parts of the framework using Vagrant. This allows us to rapidly stand-up and configure iterations of the frameworks, receive feedback, and incorporate it in the development process.

Because this framework opens up research data to web access, it was important that it integrates with university security protocols. Both the metadata manager and the data policies service use single sign-on campus authentication. Authorization to the services can be further restricted using LDAP queries to restrict access to specific groups in a directory service. Access to university cyberinfrastructure resources is handled through Kerberos-based keytab security, NFS and Samba.

For the long-term sustainability of this framework, it is important that we use established open-source tools as well as widely used and accessible programming languages such as Python and JavaScript, for which there is significant expertise on university campuses. This has allowed us to leverage existing resources and expertise in providing a highly-customizable framework.

4 FUTURE WORK

The framework is actively being extended to adapt to new use cases. We are currently leveraging the framework for the development of a centralized service for managing archives that allow users to schedule large-scale transfer to archival storage. Such a service allows users to better manage their storage in a centralized system where they are subject to quotas. From the service provider perspective, the framework facilitates more effective use of various types of storage systems.

We are also planning to extend the metadata management service with additional storage backends, as well as interfacing with cloud storage resources such as Box.

Of central importance to the framework is the file-monitoring component that ensures metadata is updated and extracted as users manipulate their files. This component depends on a diverse set of operating system and kernel APIs for file monitoring. We are working on benchmarks to assess the scalability of file monitoring to different file systems and environments.

Finally, we are continuing to pilot the framework for different research groups in order to validate and extend its supported functions.

ACKNOWLEDGMENTS

We would like to thank Professor Hui-Chen Lu and Bruna Kutche from the Gill Center for Biomolecular Science at Indiana University for their enthusiasm and feedback as first adopters of the framework described here. We are also thankful Wolf Hey for his feedback and support towards wider adoption.

REFERENCES

[1] Declan Butler. 2005. A new leaf. *Nature* 436 (06 07 2005), 20–21. https://doi.org/10.1038/436020a

[2] I. Foster. 2011. Globus Online: Accelerating and Democratizing Science through Cloud-Based Services. *IEEE Internet Computing* 15, 3 (May 2011), 70–73. https://doi.org/10.1109/MIC.2011.64

[3] Apache Foundation. 2019. Apache Airflow. https://airflow.apache.org. (2019). [Online; accessed 20-February-2019].

[4] Arcot Rajasekar, Mike Wan, Reagan Moore, and Wayne Schroeder. 2006. A prototype rule-based distributed data management system. (2006).

[5] R. W. Watson. 2005. High performance storage system scalability: architecture, implementation and experience. In *22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST'05)*. IEEE Computer Society, Washington, DC, USA, 145–159. https://doi.org/10.1109/MSST.2005.17

[6] Nancy Wilkins-Diehr. 2007. Special Issue: Science Gateways — Common Community Interfaces to Grid Resources. *Concurrency and Computation: Practice and Experience* 19, 6 (2007), 743–749. https://doi.org/10.1002/cpe.1098 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.1098