High-Throughput Neuroanatomy and Trigger-Action Programming: A Case Study in Research Automation

Conference Paper · June 2018

DOI: 10.1145/3217197.3217206

8 authors, including:

Ryan Chard
Argonne National Laboratory
35 PUBLICATIONS **312** CITATIONS
SEE PROFILE

Ming Du
Northwestern University
20 PUBLICATIONS **48** CITATIONS
SEE PROFILE

Kyle Chard
University of Chicago
122 PUBLICATIONS **1,504** CITATIONS
SEE PROFILE

Steven John Tuecke
University of Chicago
145 PUBLICATIONS **26,928** CITATIONS
SEE PROFILE

High-Throughput Neuroanatomy and Trigger-Action Programming: A Case Study in Research Automation

Ryan Chard
Argonne National Laboratory
Data Science and Learning Division
Chicago, IL

Rafael Vescovi
University of Chicago
Dept. of Neurobiology
Chicago, IL

Ming Du
Northwestern University
Depts. of Materials Science and Engineering
Chicago, IL

Hanyu Li
University of Chicago
Dept. of Neurobiology
Chicago, IL

Kyle Chard
University of Chicago and Argonne National Laboratory
Chicago, IL

Steve Tuecke
University of Chicago and Argonne National Laboratory
Chicago, IL

Narayanan Kasthuri
University of Chicago
Dept. of Neurobiology
Chicago, IL

Ian Foster
University of Chicago and Argonne National Laboratory
Chicago, IL

ABSTRACT

Exponential increases in data volumes and velocities are overwhelming finite human capabilities. Continued progress in science and engineering demands that we automate a broad spectrum of currently manual research data manipulation tasks, from data transfer and sharing to acquisition, publication, and analysis. These needs are particularly evident in large-scale experimental science, in which researchers are typically granted short periods of instrument time and must maximize experiment efficiency as well as output data quality and accuracy. To address the need for automation, which is pervasive across science and engineering, we present our experiences using Trigger-Action-Programming to automate a real-world scientific workflow. We evaluate our methods by applying them to a neuroanatomy application in which a synchrotron is used to image cm-scale mouse brains with sub-micrometer resolution. In this use case, data is acquired in real-time at the synchrotron and are automatically passed through a complex automation flow that involves reconstruction using HPC resources, human-in-the-loop coordination, and finally data publication and visualization. We describe the lessons learned from these experiences and outline the design for a new research automation platform.

KEYWORDS

Research Automation, Ripple, Neuroanatomy

ACM Reference Format:
Ryan Chard, Rafael Vescovi, Ming Du, Hanyu Li, Kyle Chard, Steve Tuecke, Narayanan Kasthuri, and Ian Foster. 2018. High-Throughput Neuroanatomy and Trigger-Action Programming: A Case Study in Research Automation. In *AI-Science'18: Autonomous Infrastructure for Science, June 11, 2018, Tempe, AZ, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3217197.3217206

AI-Science'18, June 11, 2018, Tempe, AZ, USA

ACM ISBN 978-1-4503-5862-0/18/06…$15.00
https://doi.org/10.1145/3217197.3217206

1 INTRODUCTION

Researchers are inundated with data as scientific instruments, simulation codes, and machine learning models continue to generate data at unprecedented rates. Increasing data velocities and volumes are not only overwhelming data management software but also creating new challenges for researchers who are faced with the need to perform a variety of data management tasks but who have finite time available. In this paper, we suggest that in order for researchers to keep up with data management needs, that new methods are needed to outsource and automate a broad spectrum of currently manual research data manipulation tasks, including data acquisition, transfer, publication, cataloging, and analysis.

We explore the benefits of automation in the context of tomography imaging for neuroanatomy studies conducted at the Advanced Photon Source (APS). We analyze the requirements and challenges of automating a distributed research pipeline comprising high-throughput instruments, distributed compute and storage resources, and human intervention. This pipeline uses X-ray microtomography at the APS to characterize the neuroanatomical structure of large (cm) unsectioned brain volumes. Datasets are generated at a rate of >20GB per minute and are processed with a complex reconstruction pipeline comprising many machines, tools, and services. The pipeline is initiated as data are generated at the APS, involves transferring data to and from leadership computing resources for reconstruction, and requires the cataloging and publishing of datasets across the collaboration. The pipeline also requires human intervention and validation of results at various points throughout the pipeline.

We describe here how we have automated this pipeline by using what we term Software Defined Cyberinfrastructure [15] (SDCI).

Using our SDCI implementation, called Ripple [10, 11], we specify a series of data management rules described using a Trigger-Action-Programming (TAP) model. These rules are deployed on participating storage systems to define a distributed, autonomous data management fabric. We discuss our experience using Ripple and a series of TAP rules to automate the reconstruction of neuroanatomical images during beam-time. Based on these experiences, we describe the advantages and disadvantages of our approach and propose a new research automation platform that combines Ripple's user-friendly event-based programming model with a reliable workflow management system. Our proposed platform will allow scientists to define and deploy automation pipelines by composing "flows"—collections of actions initiated in response to an external trigger event. The platform manages the execution of these flows by orchestrating and stepping through a series of actions that are performed by remote services. We describe a prototype implementation of this platform, underpinned by a reliable workflow engine, and describe how it can be used to implement the neuroanatomy pipeline.

The remainder of this paper is as follows. Section 2 describes the neuroanatomy use case that motivates this work. Section 3 presents Ripple and describes how it has been applied to the use case. In Section 4, we present a vision of a service-based automation ecosystem and a platform that orchestrates automation pipelines across remote services. We discuss related work in Section 5 and conclude in Section 6.

2 NEUROANATOMY

To motivate our work we present a neuroanatomy use case that exhibits several requirements for automation, including large data volumes produced at a rapid rate (>20GB per minute), a set of predefined actions that must be performed on data, multi-resource execution, and a desire to execute these steps rapidly. This particular example uses X-ray microtomography at the Advanced Photon Source (APS) 32-ID beamline to quickly characterize the neuroanatomical structure of large (cm) unsectioned brain volumes [18]. Current neuroanatomy approaches lack the speed of X-rays and are loosely coupled with HPC resources. The ability to run different samples and process them automatically presents new opportunities and enables novel statistical anatomy studies.

The acquisition process uses an X-ray probe beam to scan tiles of a specimen in both x and y-directions, as shown in Figure 1(a). An individual HDF5 file is created for a tile before the sample gradually moves to a new position and another tile is scanned. Approximately 12 tiles are scanned for a typical specimen at a rate of roughly one tile per minute. The raw HDF5 files, each roughly 20GB in size, are collected on a commodity computer running acquisition software. This machine is suitable for running the experiment, but not for performing any sort of analysis as overheads can interrupt the acquisition process. These data are replicated to a mid-sized server co-located at the APS to avoid interference when acting on the data. Data are then transferred to the Argonne Leadership Computing Facility to be processed and to create a single image for visualization and biological research.

Using AuTomo [14] to orchestrate reconstruction, DXchange [12] for file management, and tomoPy [16] to perform the tomographic reconstruction, the individual images are first processed to create small preview images that can be used to guide instrument positioning and ensure the data has been correctly collected. Tile registration is then performed to align the relative images using phase correction and to account for positional fluctuations [27]. A center-finding process then attempts to identify the center of rotation in order to perform reconstructions [26]. This center finding process can generate hundreds of images in varying degrees of quality. Both iterative and machine learning algorithms are applied to these images to determine the highest quality image and therefore the optimal center for the tomographic reconstruction. However, this process is unreliable and requires human input to validate the optimal center before reconstruction is performed. Once each tile has been individually reconstructed they are stitched together to create a full visualization of the specimen. The resulting dataset (approximately 200GB in size) is then moved to persistent storage where it is shared with collaborators and visualized (e.g., using Neuroglancer [6]).

There are several characteristics of this workflow that necessitate the need for automation. First, APS beamlines are provided as user facilities, where researchers propose experiments and are granted short-term allocations to use the instrument. It is crucial that *beam-time* is used effectively, as months of preparation are required in anticipation for just a few days for the experiment. Automation can improve researcher productivity and enable the researchers to optimize both the number and quality of the results they generate during their allocation. Second, in many cases these experiments aim to perform the same analysis for many datasets and thus the workflow executed is the same for every dataset. Third, the workflow requires a series of operations, conducted in different places, with little need for manual control.

Based on our experiences we believe that the analysis and data management requirements of this use case are representative of those in many research laboratories. Although the instruments, computing resources, and scale of data generation tend toward the extreme, the underlying principles of moving data, analyzing it with batch computing resources, and cataloging and publishing results are typical across many domains.

3 AUTOMATION WITH RIPPLE

We describe how we have implemented the neuroanatomy pipeline using Ripple, and discuss the lessons learned from this implementation that inform our design of an automation platform.

3.1 Ripple

Ripple is a distributed, rule-based automation system that enables the automation of research data management tasks through simple Trigger-Action-Programming (TAP). Ripple allows users to define TAP rules and apply them to specific storage systems. These rules are simple pairs: when a particular *event* occurs then Ripple will invoke a predefined *action*. Ripple supports events generated from file systems (e.g., file creation and deletion) and actions via Globus. Ripple can be used to define a wide range of rules. For example, it can automatically transfer data as it is generated, or invoke a script each time a particular file type is placed in a designated directory.

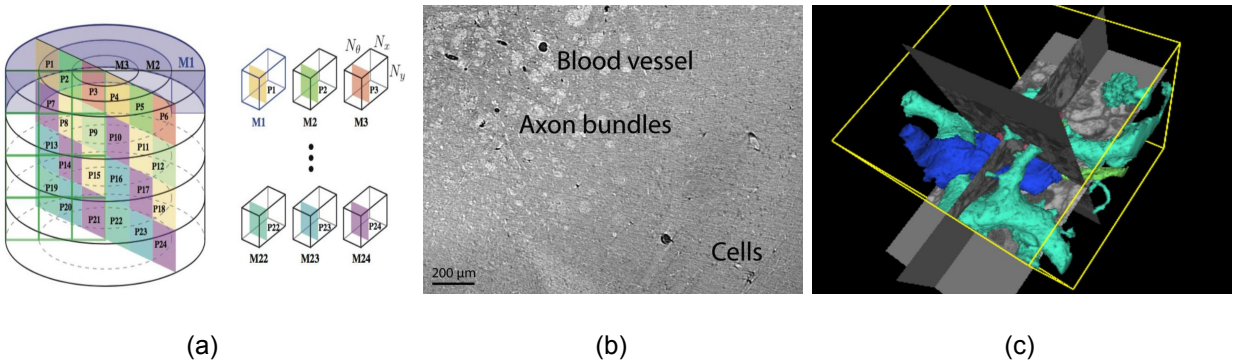

Figure 1: (a) Schematic diagram showing data acquisition process. (b) A magnified coronal slice. (c) A 3D rendering of the reconstructed brain volume using Neuroglancer.

Ripple relies on a lightweight local agent to monitor the file system and detect events. The agent includes a modular event monitoring interface that supports a suite of event-detection tools, such as `inotify` and kqueue. Once a relevant event is detected, the local agent reports it to a cloud-hosted service that then invokes the appropriate action. To apply Ripple to leadership storage systems, we have developed a high performance monitor to detect events across large (petabyte) Lustre and GPFS storage devices [24]. This monitor leverages the internal event monitoring capabilities of these file systems to report data events across the entire file system.

Ripple's action model is simple. It supports two varieties of actions: those performed locally by the Ripple agent (e.g., execute a script locally) and those performed by Ripple's cloud-hosted service (e.g., to invoke an external service). To support local actions we have developed a simple execution interface within the Ripple agent that can be used to run arbitrary commands (e.g., executing a script locally). To support remote actions, Ripple is able to call a collection of predefined services, such as Globus (perform transfers and set permissions) and Amazon services (send emails and invoke Lambda functions).

While a single TAP rule can automate a simple task, researchers usually want to implement more sophisticated automation pipelines that involve sequences of such tasks. We have used Ripple to encode such pipelines by daisy-chaining rules into a multi-step pipeline, with each trigger condition programmed to match the output of the previous action. However, this approach is cumbersome, is difficult to debug, and is error-prone.

3.2 Pipeline implementation

We expressed the neuroanatomy reconstruction pipeline in Ripple as a set of 10 TAP rules, described in Figure 3. These rules automate the end-to-end analysis of raw beamline data into visualizable datasets and publishes them for user consumption. Ripple agents have been deployed at both the APS acquisition machine and at the ALCF cluster, Cooley. Figure 2 shows how Ripple manages this pipeline.

The pipeline is initiated as new files with a *".h5"* extension are created in a *raw_data* directory on the APS acquisition machine. Ripple detects these new data files and initiates a Globus transfer to move the data to Cooley for processing. Once the transfer completes, the data are preprocessed with the AuTomo tool to restructure the directory into a standardized format. A batch submission file is then created in response to the data file being renamed. The batch file encodes the use of AuTomo to construct a preview from the data as well as execute the center finding tools. When the preview images are created they are returned to the user at the beamline to provide initial insight into the results and aid in instrument positioning. At the completion of the center finding process, denoted by the generation of a *center_pos.txt* file, the center slices and estimated center of rotation value are returned to the user. The automated flow then waits until the user creates a *real_center_pos.txt* file, containing the verified center value. This file is transferred to the ALCF for use in the reconstruction. When the transfer completes, the remainder of the pipeline is executed by creating and dispatching a batch submission script to perform the reconstruction of the image. Once reconstructed, the entire dataset, center slices, and reconstructed tiff stack are published to persistent storage to be shared with collaborators. Finally, metadata are extracted from the raw data file and associated with the resulting reconstruction tiff stack in a Globus Search index.

3.3 Reflecting on Ripple Automation

Ripple has proven successful as a platform for automating various tasks and pipelines. In particular, we have found that the TAP programming model provides a relatively user-friendly method for defining automation policies. However, our experience implementing multi-step pipelines has highlighted the challenges of creating reliable workflows comprised of cascading rules. Ripple will automatically retry an action if it fails. However, the lack of a rigorous workflow system composing multi-step TAP rules means that if an action simply fails to produce the required trigger of a subsequent rule (yet runs successfully), the pipeline is interrupted. When managing small pipelines, or pipelines with few actions, manually curating the pipeline and resolving problems is not difficult. However, when hundreds of pipelines are executing concurrently it is difficult to identify and resolve interrupted chains. A workflow system capable of managing the entire multi-step pipeline as a whole would improve the reliability of our automation system and

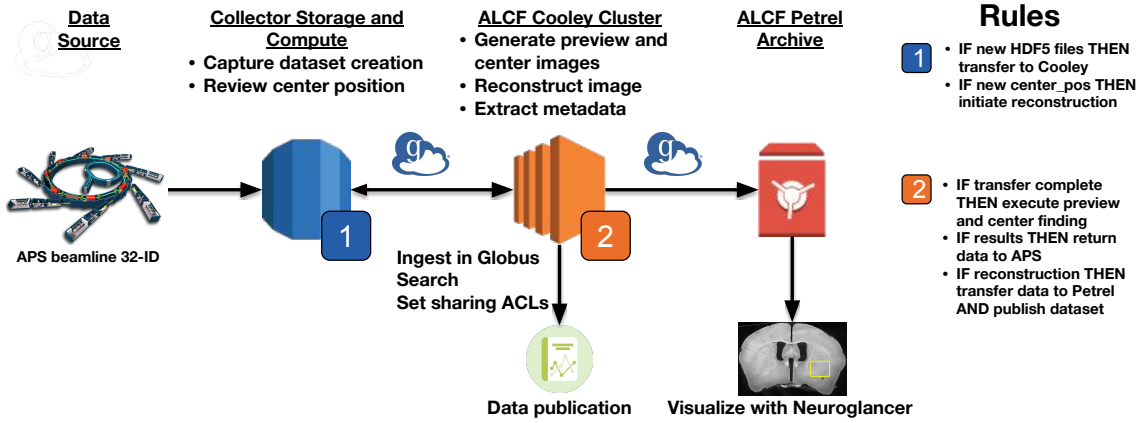

Figure 2: An overview of Ripple being used to automate the neuroanatomy pipeline.

Step 1: Transfer to ALCF
Trigger: *FileCreated*, `.*.h5`
Action: *GlobusTransfer*, `$FILE`, APS→ALCF

Step 2: Organize HDF5
Trigger: *TransferComplete*, `.h5`, APS→ALCF, User:Brainimaging
Action: *Bash*, `bash automo_organize.sh`

Step 3: Create batch file
Trigger: *FileCreated*, `data.h5`
Action: *Bash*, `echo ... >> $PATH/batch_job.qsub`

Step 4: Submit batch file
Trigger: *FileCreated*, `batch_job.qsub`
Action: *Cobalt*, `qsub $FILE`

Step 5: Return center
Trigger: *FileCreated*, `center_pos.txt`
Action: *GlobusTransfer*, `$PATH`, ALCF→APS

Step 6: Transfer verified center
Trigger: *FileCreated*, `real_center_pos.txt`
Action: *GlobusTransfer*, `$FILE`, APS→ALCF

Step 7: Create batch reconstruction file
Trigger: *TransferComplete*, `real_center_pos.txt`, APS→ALCF
Action: *Bash*, `echo ... >> $PATH/recon_job.qsub`

Step 8: Submit batch reconstruction file
Trigger: *FileCreated*, `recon_job.qsub`
Action: *Cobalt*, `qsub $FILE`

Step 9: Publish results
Trigger: *FileCreated*, `recon_0000.tiff`
Action: *GlobusTransfer*, `$PATH`, ALCF→Petrel

Step 10: Catalog results
Trigger: *TransferComplete*, `recon_0000.tiff`, ALCF→Petrel
Action: *Bash*, `python catalog_data.py $PATH`

Figure 3: Ripple TAP rules used to automate APS image reconstruction.

greatly simplify the debugging of pipelines. In addition, our experience has shown that existing pipelines do not necessarily map well to TAP combinations and actions are occasionally required to be customized to produce unnecessary outputs simply to trigger subsequent rules, for example, one may need to alter an execution script to generate a done.txt file on completion.

While creating TAP rules is simple, creating *correct* TAP rules is non-trivial [17]. File creation events provide an example of the problems that can arise. Most Ripple rules are triggered by file events, which provide an easily understandable trigger condition for users of any technical ability. However, if a program or user creates a file with the same name as an existing file, a file modification event, rather than a creation event, will be raised—a potentially confusing result. Such problems are compounded when creating multi-step pipelines from individual trigger-action rules. Therefore, methods to ensure correctness of automation pipelines during both their creation and execution must be investigated.

The Ripple architecture is not easily extensible and does not currently allow users to register their own event sources or actions. For example, one cannot raise custom events without altering the source code to support such events. This restricts users from developing custom tools and enabling user-friendly human-in-the-loop interfaces for arbitrary tasks. For example, in the neuroanatomy pipeline, we encode human-in-the-loop actions by requiring that a specific file be created (raising a file trigger event) to continue the pipeline. Ideally, one would simply create a website where users could perform an action (e.g., indicating the center value) and return control to the pipeline.

4 TOWARDS SERVICE-BASED AUTOMATION

Our experiences using Ripple to automate various data management use cases, including not only neuroanatomy but also examples in cosmology and materials science, indicate a need for a flexible, workflow-oriented, research automation system that simplifies the application of trigger-action programming to common research tasks. We outline the design of such a system and describe our initial prototype.

We envision a research automation platform that offers the simplicity of Ripple's trigger-action programming methodology and distributed event capture model, adds an extensible interface via which arbitrary event sources and actions can be integrated, and is built around a cloud-hosted, automation engine that is capable of orchestrating and reliably executing pipelines comprised of many actions. Our goals in developing such a platform are to (1) make it easy, even for non-experts, to create robust, secure distributed research automation flows, comprising both automated and human-in-the-loop actions, that can be automatically triggered by events; (2) provide scalable, secure, and fault-tolerant execution of flows comprising a collection of distributed actions; and (2) enable the system to be extended by integrating arbitrary events and actions generated/performed by external services exposing a simple REST API.

Our goals, in some ways, are similar to home automation services like IFTTT [5], except that we aim to support multi-step automation flows, rather than just simple trigger-action behaviors; asynchronous, long-lived actions, in addition to synchronous tasks; and robust end-to-end security that integrates with research cyberinfrastructure. Our approach is also differentiated by our focus on research automation examples, and the associated integration with common research event sources and actions such as parallel file systems, Globus, and analysis execution capabilities, rather than weather services, light bulbs, and garage doors.

To ensure that our platform can indeed be used by non-expert users, we aim to develop user-friendly graphical interfaces to support the definition and execution of flows as well as the specification and integration of event sources and actions, with relative ease. Furthermore, tight integration with a research oriented authentication and authorization model, such as that provided by Globus Auth [8], will simplify technical challenges associated with securely performing actions on behalf of users by automatically acquiring and passing credentials to remote services.

In the remainder of this section we describe the current implementation of the automation platform and describe how we can apply it to the neuroanatomy use case.

4.1 A Reliable Automation Engine

At the core of our platform is an automation engine that is responsible for reliably executing automation flows. This engine requires many of the capabilities (e.g., scalability, reliability, flexibility) of a well-tested workflow engine. After review of workflow models, such as Amazon Simple Workflow Service [2], Netflix's Conductor [4], and Apache Airflow [1], we identified Amazon Step Functions [3] as the most well-suited workflow engine for managing flows. Step Functions is provided by AWS as a managed service for reliably and scalably executing state machines (i.e., automation flows); its declarative and intuitive JSON-based state machine language allows non-experts to define flows and its flexible execution model allows our automation platform to execute arbitrary actions. This language also allows for concise definition of flows comprising multiple actions, with control logic ranging from simple sequential actions to complex branching and iteration, with simple but powerful timeout, retry, and recovery capabilities.

Step functions allow us to represent arbitrary state machines, however they have only limited support for performing operations. In order to support arbitrary external actions we have developed an execution model based on Amazon Lambda functions. For each registered action we create a corresponding Lambda function that is parametrized to match the action execution API. We internally map the Lambda function's identifier to a user-friendly action name, such as *Transfer*, which can then be used by users to incorporate the action into a flow. When a flow executes this action, the Lambda function is invoked which in turn executes the action using parameters obtained from the flow's state.

4.2 Events and Actions

To achieve our goal of developing an ecosystem of event sources and actions that can be used in automation flows, we define standardized APIs for integrating both event sources and action services.

We define an event REST API to be implemented by any service that wants to produce events for consumption. Our goal is to make it simple for a service to produce and consume events, while also enabling performant and robust event delivery. Our event API enables the automation platform to register interest in an event type and to then periodically poll the service for new events. As a proof-of-concept, we have extended the Ripple cloud-hosted service with this API, enabling it to aggregate and publish events from Ripple agents deployed on arbitrary file systems. Other event sources, such as Globus Transfer, can also expose this API and be integrated into the automation ecosystem in the future.

We have defined an action API for users to register services with the automation platform. Any user can define custom actions by developing a Web service (or augmenting an existing service), exposing the required API, and registering the service. Once registered, the action is accessible to the automation platform and can be used in a flow. A key requirement of our work is to support both human-in-the-loop and asynchronous actions. Thus, it is necessary for flows to execute an action that, for example, waits for a user to complete a metadata form. To achieve this we have created a polling action that can be used to halt a flow until an action completes. The polling action is implemented as a Step Function *Activity Worker*. Users can include the polling action into a flow by passing it the identifier of a task to monitor. The polling worker then periodically requests status updates from the action until it receives a completed status. Once the polling worker returns the flow will resume and proceed to the next step.

4.3 An Ecosystem for Automation

Before being able to apply the automation platform to our neuroanatomy reconstruction pipeline we require a number of services to facilitate the necessary actions. Below we discuss several such services and describe how they can be integrated in our platform to fulfill these needs.

In order to transfer data between locations we use Globus to provide high performance and reliable data transfer. To integrate Globus, we will need to implement the automation platform action API. As a proof of concept, we have developed a thin service, using the Globus Python SDK, that exposes the necessary action interface

to initiate transfers and associated status interface to obtain transfer status on request.

We then require a service that is capable of executing computational tasks on remote computers. To do so, we have created a simple execution service using Parsl [9]. This service is able to facilitate remote execution through a secure SSH connection using a pilot job model to execute work directly on the specified resource. To explore this approach, we have deployed our prototype and enabled it to perform executions on the ALCF's Cooley cluster.

Finally, the reconstruction pipeline must accommodate human-in-the-loop tasks (e.g., to specify the optimal center of rotation from which the reconstructions will be based). To address this need we have developed a user-centric service to perform the center-verification process. This service allows users to associate an integer, representing the center position, with a flow, which is then used to perform the reconstruction.

4.4 Autonomous Automation

Our prototype automation platform, along with the services described above, is now capable of automating the neuroanatomy pipeline. Our experiences developing the Ripple-based pipeline as well as prototyping the automation platform has identified several areas in which autonomous computing concepts could be applied.

The remote execution service could be extended to dynamically guide execution decisions based on historical traces. For example, execution traces and knowledge of the data used in a flow could be used to forecast execution durations (and cost), predict data transfer costs, and dynamically determine optimal execution locations.

The center-verification service could be extended to aid the user in selecting optimal center values, in turn improving the quality of the reconstructed images. For example, the center images could be moved to this service and displayed to the user. The result of the center-finding algorithms could guide what the user is shown and estimated errors could be associated with each slice.

Finally, the entire automation system could use autonomic computing approaches to further reduce the burden on users by identifying common actions and flows associated with trigger event types. For example, a common flow may be to extract metadata from CSV files and, given permission, the automation platform could silently add value to these datasets by extracting and aggregating metadata on the file.

5 RELATED WORK

Lustre's Robinhood [19] and the Integrated Rule-Oriented Data System (iRODS) [25] allow system administrators to automate infrastructure-wide tasks, such as purging data and creating regular backups. While similar to our goals, we instead aim to automate entire research workflows that include a variety of data management and analytics tasks. Our prior work to develop Software Defined Cyberinfrastructure [15] aims to empower end users to automate such a wide variety of processes via programmable infrastructure. Here we extend this model to provide robust automation workflows, rather than loosely coupled rules. Similar work by AbdelBaky et al. [7] presents a programmable infrastructure service composition approach to facilitate the control and composition of services, leading to the federation of diverse services.

There is currently an unmet need for secure and reliable automation of sequences of data management tasks that may span locations and timescales, and integrate both mechanical and human actions. This is a different problem to that of programming parallel workflows, as handled by HTCondor [20], Kepler [21], Parsl [9], Pegasus [13], Swift [28], and Taverna [23], or building integrated data management systems, as handled by iRODS [25] and business rules engines [22]. IFTTT [5] is the most similar service to our proposed automation platform, being simple, SaaS, and extensible to new events and actions. However, it does not integrate with cyberinfrastructure security or resources, or handle sequences of actions.

6 CONCLUSION

The trend towards computational and data-oriented research is rapidly overwhelming traditional data management techniques, leaving researchers with poorly cataloged and managed data, reducing data reusability and value, and placing significant burden on researchers. Here we have presented two approaches for automating research data management tasks in the context of a data-intensive neuroanatomy use case. We have described how our SDCI implementation, called Ripple, has been used to automate this particular use case and discussed its shortcomings. We then presented a new approach to research automation that leverages an extensible and reliable cloud-hosted service to support the definition and execution of workflows comprised of external actions. Finally, we described how we can use this platform to meet the complex requirements of our use case.

In future work we aim to first apply the prototype platform to the neuroanatomy use case during the next available beam-time for this group. We then aim to productionize the platform such that it can be applied to other use cases. We will continue to develop the event and action interfaces required to integrate external services with the platform and work with others to extend existing services to implement these interfaces.

ACKNOWLEDGMENTS

We acknowledge support from DOE contract DE-AC02-06CH11357, RAMSES, and research credits provided by Amazon Web Services.

REFERENCES

[1] [n. d.]. Airflow. ([n. d.]). https://airflow.apache.org/. Accessed April 1, 2018.
[2] [n. d.]. Amazon Simple Workflow Service. ([n. d.]). https://aws.amazon.com/swf/. Accessed April 1, 2018.
[3] [n. d.]. Amazon Step Functions. ([n. d.]). https://aws.amazon.com/step-functions. Accessed April 1, 2018.
[4] [n. d.]. Conductor. ([n. d.]). https://netflix.github.io/conductor/. Accessed April 1, 2018.
[5] [n. d.]. If This Then That. ([n. d.]). https://platform.ifttt.com/docs. Accessed April 1, 2018.
[6] [n. d.]. Neuroglancer. ([n. d.]). https://github.com/google/neuroglancer. Accessed April 1, 2018.
[7] Moustafa AbdelBaky, Javier Diaz-Montes, and Manish Parashar. 2017. Software-defined environments for science and engineering. *The International Journal of High Performance Computing Applications* (2017), 1094342017710706.
[8] Rachana Anathankrishnan, Kyle Chard, Ian Foster, Mattias Lidman, Brendan McCollam, Stephen Rosen, and Steven Tuecke. 2016. Globus Auth: A Research Identity and Access Management Platform.
[9] Yadu Babuji, Alison Brizius, Kyle Chard, Ian Foster, Daniel S. Katz, Michael Wilde, and Justin Wozniak. 2017. Introducing Parsl: A Python Parallel Scripting Library. (Aug. 2017). https://doi.org/10.5281/zenodo.891533

[10] R. Chard, K. Chard, J. Alt, D. Y. Parkinson, S. Tuecke, and I. Foster. 2017. Ripple: Home Automation for Research Data Management. In *37th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW)*. 389–394. https://doi.org/10.1109/ICDCSW.2017.30

[11] R. Chard, K. Chard, S. Tuecke, and I. Foster. 2017. Software Defined Cyberinfrastructure for Data Management. In *13th IEEE International Conference on e-Science (e-Science)*. 456–457. https://doi.org/10.1109/eScience.2017.69

[12] Francesco De Carlo, Doga Gürsoy, Federica Marone, Mark Rivers, Dilworth Y Parkinson, Faisal Khan, Nicholas Schwarz, David J Vine, Stefan Vogt, S-C Gleber, et al. 2014. Scientific data exchange: a schema for HDF5-based storage of raw and analyzed data. *Journal of synchrotron radiation* 21, 6 (2014), 1224–1230.

[13] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P.J. Maechling, R. Mayani, W. Chen, R.F. da Silva, M. Livny, et al. 2015. Pegasus, a workflow management system for science automation. *Future Generation Computer Systems* 46 (2015), 17–35.

[14] Ming Du, Rafael Vescovi, Ryan Chard, Narayanan Kasthuri, Chris Jacobsen, Eva Dyer, and Doga Gursoy. 2018. An Automated Pipeline for the Collection, Transfer, and Processing of Large-scale Tomography Data. In *Biophotonics Congress: Biomedical Optics Congress 2018 (Microscopy/Translational/Brain/OTS)*. Optical Society of America, BF4C.2. https://doi.org/10.1364/BRAIN.2018.BF4C.2

[15] I. Foster, B. Blaiszik, K. Chard, and R. Chard. 2017. Software Defined Cyberinfrastructure. In *37th IEEE International Conference on Distributed Computing Systems (ICDCS)*. 1808–1814. https://doi.org/10.1109/ICDCS.2017.333

[16] Doga Gürsoy, Francesco De Carlo, Xianghui Xiao, and Chris Jacobsen. 2014. TomoPy: A framework for the analysis of synchrotron tomographic data. *Journal of Synchrotron Radiation* 21, 5 (2014), 1188–1193.

[17] Justin Huang and Maya Cakmak. 2015. Supporting Mental Model Accuracy in Trigger-action Programming. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15)*. ACM, New York, NY, USA, 215–225. https://doi.org/10.1145/2750858.2805830

[18] Narayanan Kasthuri, Kenneth Jeffrey Hayworth, Daniel Raimund Berger, Richard Lee Schalek, José Angel Conchello, Seymour Knowles-Barley, Dongil Lee, Amelio Vázquez-Reina, Verena Kaynig, Thouis Raymond Jones, et al. 2015. Saturated reconstruction of a volume of neocortex. *Cell* 162, 3 (2015), 648–661.

[19] Thomas Leibovici. 2015. Taking back control of HPC file systems with Robinhood Policy Engine. *arXiv preprint arXiv:1505.01448* (2015).

[20] Michael J Litzkow, Miron Livny, and Matt W Mutka. 1988. Condor–a hunter of idle workstations. In *8th International Conference on Distributed Computing Systems*. IEEE, 104–111.

[21] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A Lee, Jing Tao, and Yang Zhao. 2006. Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice and Experience* 18, 10 (2006), 1039–1065.

[22] Ayman Meidan, Julián Alberto García-García, MJ Escalona, and I Ramos. 2017. A survey on business processes management suites. *Computer Standards & Interfaces* 51 (2017), 71–86.

[23] Tom Oinn, Matthew Addis, Justin Ferris, Darren Marvin, Martin Senger, Mark Greenwood, Tim Carver, Kevin Glover, Matthew R Pocock, Anil Wipat, et al. 2004. Taverna: A tool for the composition and enactment of bioinformatics workflows. *Bioinformatics* 20, 17 (2004), 3045–3054.

[24] Arnab K. Paul, Steven Tuecke, Ryan Chard, Ali R. Butt, Kyle Chard, and Ian Foster. 2017. Toward Scalable Monitoring on Large-scale Storage for Software Defined Cyberinfrastructure. In *2nd Joint International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems (PDSW-DISCS '17)*. ACM, New York, NY, USA, 49–54. https://doi.org/10.1145/3149393.3149402

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

[26] Mark L Rivers. 2012. tomoRecon: High-speed tomography reconstruction on workstations using multi-threading. In *Developments in X-Ray Tomography VIII*, Vol. 8506. International Society for Optics and Photonics, 85060U.

[27] RFC Vescovi, MB Cardoso, and EX Miqueles. 2017. Radiography registration for mosaic tomography. *Journal of synchrotron radiation* 24, 3 (2017).

[28] M. Wilde, M. Hategan, J.M. Wozniak, B. Clifford, D.S. Katz, and I. Foster. 2011. Swift: A language for distributed parallel scripting. *Parallel Comput.* 37, 9 (2011), 633–652.