Data Automation at Light Sources

Ben Blaiszik[1], Kyle Chard[1], Ryan Chard[1], Ian Foster[1,2,a)] and Logan Ward[1]

[1]*Data Science and Learning Division, Argonne National Laboratory, Argonne IL 60439, USA.*
[2]*Department of Computer Science, University of Chicago, Chicago IL 60637, USA.*

[a)]Corresponding author: foster@anl.gov

Abstract. Rapidly growing data volumes at light sources demand increasingly automated data collection, distribution, and analysis processes, in order to enable new scientific discoveries while not overwhelming finite human capabilities. We present here the case for automating and outsourcing light source science using cloud-hosted data automation and enrichment services, institutional computing resources, and high- performance computing facilities to provide cost-effective, scalable, and reliable implementations of such processes. We discuss three specific services that accomplish these goals for data distribution, automation, and transformation. In the first, Globus cloud-hosted data automation services are used to implement data capture, distribution, and analysis workflows for Advanced Photon Source and Advanced Light Source beamlines, leveraging institutional storage and computing. In the second, such services are combined with cloud-hosted data indexing and institutional storage to create a collaborative data publication, indexing, and discovery service, the Materials Data Facility (MDF), built to support a host of informatics applications in materials science. The third integrates components of the previous two projects with machine learning capabilities provided by the Data and Learning Hub for science (DLHub) to enable on-demand access to machine learning models from light source data capture and analysis workflows, and provides simplified interfaces to train new models on data from sources such as MDF on leadership scale computing resources. We draw conclusions about best practices for building next-generation data automation systems for future light sources.

INTRODUCTION

Light source scientists are facing a data crisis as the rate at which new instruments generate data volumes is rapidly exceeding Moore's Law [1]. Growing data volumes present new challenges as they overwhelm finite human capabilities, requiring new machine-based solutions to automate and outsource the acquisition, analysis, and distribution of light science data. Neither humans or computers can cope using current methods. Existing techniques to design experiments, manage and analyze data, and create and deliver software will not suffice for next generation data rates. However, these challenges also present new opportunities and offer the potential to create the new automation systems and enrichment services necessary to utilize instrument advancements. New approaches are required to unburden scientists from time consuming data munging, management, and dissemination practices, including reliably indexing and cataloging, pipelining, and transforming results.

Fourth-generation light sources such as the Advanced Photon Source Upgrade (APS-U) [2] will offer exciting new scientific capabilities to the thousands of scientists who use their many beamlines annually. They also pose major data and computation challenges, for at least four reasons. First, they will generate massive data. For example, x-ray photon correlation spectroscopy can already generate 2MB images at 3,000 Hz, for a data rate of 6 GB/s, comparable to that of the Large Hadron Collider [3]. With APS-U, this data rate is expected to increase by 2–3 orders of magnitude. Second, experiments will increasingly generate complex multi-modal data that needs advanced computing for interpretation, such as ptychography combined with elemental mapping and visual images as a function of reaction conditions. Third, it will become increasingly feasible to use advanced theory and modeling to fit and co-optimize model and experiment. Fourth, as synchrotron light sources mature as instruments, they increasingly serve more and different users, many with limited or no experience with light sources. Automation is important for such users [4, 5].

For light science to exploit these advancements the tools and services must transition from an artisanal/cottage industry to one that builds on automated solutions that are integrated into daily operation of a beamline. Tools to

automatically capture, apply advanced analyses, and autonomically drive experiments are necessary, such as those we have previously explored [6]. In addition, once data are acquired, scientists must outsource their ongoing management and distribution through dedicated, community-oriented storage hubs that make discoverable and accessible the results of experiments. Storage catalogs, such as the Materials Data Facility [7], provide rich environments to foster collaboration and increase the value of datasets by disseminating results and in turn, accelerating scientific discovery. Similarly, nexuses for applying transformations and analyses provide platforms that federate access to cutting edge analyses, optimized high performance modalities, and remove barriers of entry to leadership computing facilities.

Here we describe our efforts to outsource and automate data distribution, management, and transformation. We describe how each enables light source scientists to outsource tasks crucial for their science. First, we present data acquisition and distribution tools and services, describing specifically how the Globus data publication platform can enable scalable and secure data distribution using powerful data access and discovery capabilities. Second, we outline a new Globus cloud-hosted data automation service and describe how it can be used to manage sophisticated data lifecycles. Finally, we explore the data transformation capabilities provided by the Data and Learning Hub for science (DLHub) to enable simple, yet scalable, training of machine learning models and on-demand access to machine learning models from light source data capture and analysis workflows.

BACKGROUND/HISTORY

We first experimented with online analysis of APS data in the late 1990s, when we coupled computing microtomography [8, 9] and crystallographic [10] experiments to remote computers. In one memorable demonstration in 1998, we piped microtomography data from APS beamline 2-BM to a 96-node SGI Origin parallel computer at Argonne for incremental reconstruction via filtered backprojection as an experiment was proceeding, and then streamed visualization data to the Supercomputing conference (SC'98) in Orlando, Florida, for interactive analysis: see Figure 1. As we noted at the time, "the data rates and compute power required ... are prodigious, easily reaching one gigabit per second and a teraflop per second [respectively]" [11]—numbers that are dwarfed by today's requirements. These demands have since increased tremendously, as illustrated by recent work on near-real-time tomography [12, 13, 14].

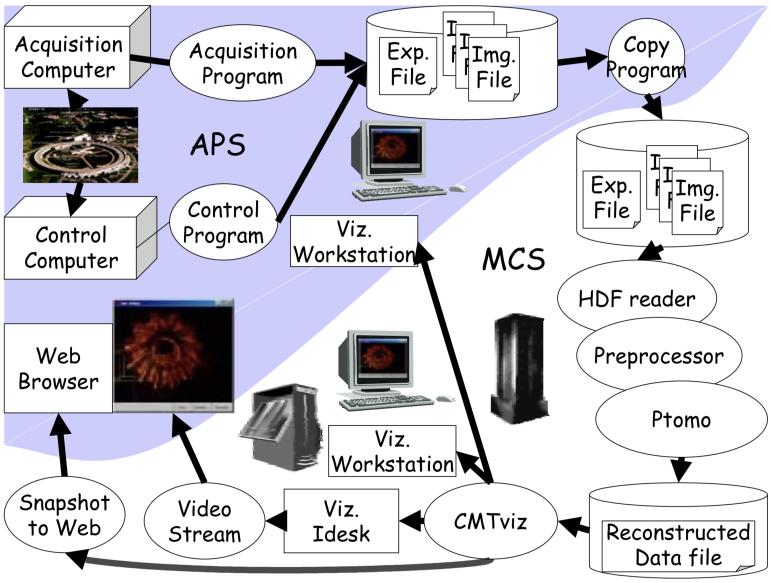


FIGURE 1. The processing pipeline used in the SC'98 demonstration [11]. Data collected at the APS were passed to a supercomputer in Argonne's Mathematics and Computer Science (MCS) division for incremental reconstruction and visualization, and results dispatched as a stereoscopic video stream to a remote virtual reality display at APS and in Orlando.

More recently, we have worked to combine experiment and simulations for diffuse scattering experiments: see Figure 2. The figure highlights the interactions between experiment, computation, and human expertise at various timescales. In particular, it shows how simulation and experimental workflows can be combined to enable experiment

steering and rapid validation of experiment results. On the experiment side, diffuse scattering data are obtained using a beamline. High performance computers are used to reconstruct and analyze the data, often in near real-time to provide rapid feedback to researchers. Here, data is collected as a sample is rotated 360 degrees in tenth-of-a-degree increments, yielding 3,600 images. Each image comprises 2048×2048 32- bit pixels, (i.e., 16 MB), a total of 56 GB. Images are produced at a rate of 10 per second and thus at a peak rate of 160 MB/s. Following the collection of a complete 3,600-image dataset, which takes about 10 minutes, either a new sample is inserted or the sample conditions are changed, and the process is repeated.

On the simulation side, researchers explore a huge range of synthetic structures using various high performance software packages, and increasingly machine learning methods to create simulated output to be compared with, or to help guide, experiments. It is important to note, that these experiments and simulations need not be performed by the same researchers or at the same time. Rather, these simulations are often performed by collaborations that span institutions and even domains.

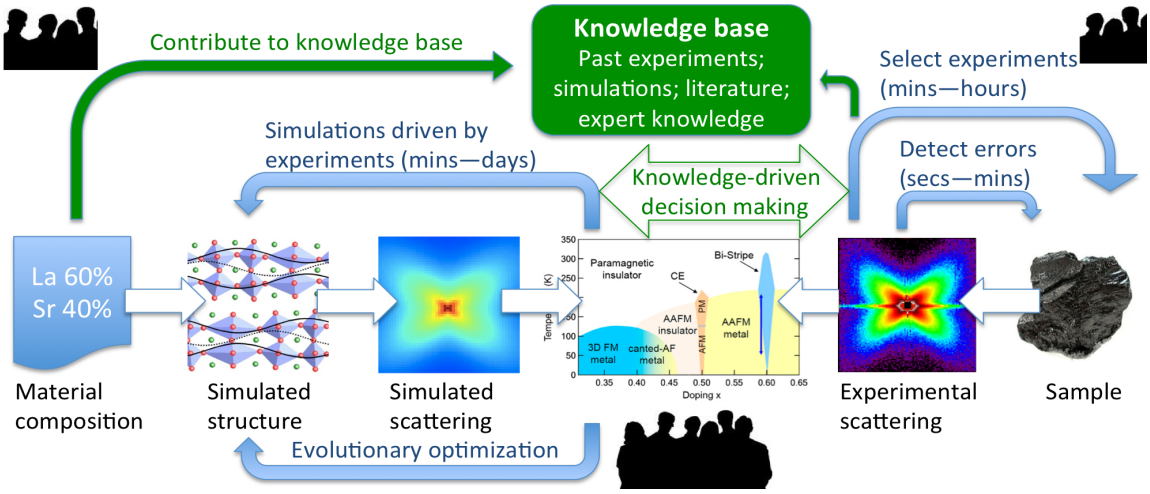


FIGURE 2. Activities involved in a diffuse scattering experiment. Data is acquired via simulation and experimentation, involves data storage in various locations from beamline computers through to data archives, requires analysis at various scales and coordination between a diverse group of collaborators. powerful knowledge base and simulation capabilities, can create a "discovery engine" for materials science research.

It is our long history working with light science researchers, experiments, and simulations that informs our belief that existing methods will not meet the requirements of next generation instruments. Indeed, new approaches are needed to manage increasing data volumes, design yet more complex experiments, drive acquisition at unprecedented rates, and apply and integrate advanced analyses. As we will describe, new methods that automate and outsource crucial data management and manipulation tasks are needed to exploit instrument advancements and advance light source science outcomes.

MOTIVATION: AUTOMATION AND OUTSOURCING

As the scale and complexity of light-source science continues to grow, so too does the burden on researchers to manage an increasingly complex ecosystem of data, software, and cyberinfrastructure. Unfortunately, these burdens are increasingly prohibitive as the workload consumes considerable research time. If we look to other fields, from the automotive industry to agriculture, the ability to automate common or mundane activities has underpinned massive advances in terms of productivity and efficiency. Similar advances are required to combine computational and data-driven science with experiment to speed discovery.

Over the past dozen years Globus [15] has demonstrated the value of outsourcing a myriad of common research data management activities, including data transfer, synchronization, and sharing. In these cases, researchers benefit from the ability to outsource activities to a reliable, cloud-hosted service that deals with the complexity of these tasks. For example, when transferring data, Globus manages credentials required to access storage systems, establishes

and maintains a secure connection between storage systems, optimizes transfer configuration to provide rapid data movement, validates data integrity, and recovers from errors.

Achieving similar gains in light source sciences will require the use of such services (e.g., those provided by Globus) as well as development of new services that are able to perform many other operations. We believe there is potential for widespread impact with such approaches and simultaneously other benefits such as reduced costs (via economies of scale) and increased reliability (due to professionally hosted services). We observe that many light source experiments apply a similar foundational processes to the acquisition, analysis, and distribution of data. We posit that components of the light source science data lifecycle can be reliably outsourced to specialized services, thus reducing the burden on researchers. Specialized services can be created to service many distributed sources concurrently, reliably, and efficiently. These services can reliably provide the tools necessary to perform a wide range of advanced analyses without requiring the scientist to have specialized skills in terms of building, running, and maintaining analytical tools, and removing the barriers to entry to apply analytical tools across leadership computing resources. In addition, specialized services to automatically extract metadata and enforce best-practice cataloging and replication can be deployed such that scientists can simply invoke such a service, rather than creating *ad hoc* data management solutions with little reproducibility, adding to the vanguard computing environment that litters many large-scale experimental science instruments.

Automation provides the second piece to this puzzle allowing for frequent operations to be applied without human involvement. by further raising the level of abstraction, such that light source scientists may define frequent operations (or series of operations) to be performed we further reduce the burden on humans while also increasing reliability and efficiency. Examples of automation include moving data when it is acquired to large storage or analysis resources and thereby eliminating the need for scientists to monitor storage space and orchestrate the movement of data or executing data quality control algorithms to automate the detection of errors in data acquisition and thereby eliminate costly experiment inefficiency. While full automation may be possible in some circumstances, partial automation via interactions between researchers and the automation processes may find early and more widespread adoption. Interfaces to facilitate such complex interactions will be critical at many points in the research lifecycle and at various scales: from direct steering of experiments to the choice of the next experiment and simulation. Human-in-the-loop interactions are critical to both the design of experiments, analysis of results, correction of errors, supplementation of training data, and optimization of experiments.

Developing such an ecosystem of services and existing software will support the synthesis of data from simulation and experiment into an evolving and accessible knowledge base. This programatically accessible knowledge base can then be used to automate analyses, perform machine learning tasks, evaluate data quality, and critically to guide current and future research direction. Automation of these processes holds the promise of simplifying the processes used by scientists, improving throughput, and increasing both the quality and accuracy of the results generated.

DATA ACQUISITION AND DISTRIBUTION

Data acquisition and distribution are fundamental to light science. As the rate at which both data are generated from new instruments and robotic processes are increasingly being used to rapidly manage the analysis of multiple samples, automated acquisition processes become increasingly necessary. New techniques to manage the entire data lifecycle in response to their generation are needed.

A range of tools have been developed to simplify the acquisition process. For example, the Data Management at the APS Imaging Group (DMagic) have worked to automate the collection process and facilitate reliable execution of data management tasks, such as sharing, indexing, and transfer, in response to data events. DMagic provides an event-driven system to detect the creation of files and enact actions, such as data movement, in response to samples being acquired. DMagic uses Globus APIs create a shared endpoint and deposit the data and set permissions. It also bridges the multiple data sources maintained at the APS, such that given an experiment's date and time, it can retrieve user information from the APS scheduler, associate metadata regarding the beamline with the experimental results, and email the user the location of the shared endpoint for data retrieval. Similarly, ISPyB is a Laboratory Information Management System that combines sample tracking and experiment reporting for synchrotrons, in particular for macromolecular crystallography. ISPyB has been in production for numerous years and is specifically designed to simplify development and maintenance [16].

In order to enable easy, efficient, secure, reliable distribution of data from beamlines to many locations, platforms are required to store and distribute data with high performance and accessibility. At Argonne National Laboratory we have created a purpose built platform for sharing research outcomes underpinned by high performance networks and

the Globus data management platform. This service, called Petrel, provides researchers access to a large, 1.7PB, store positioned within Argonne's network with high speed access to the APS. Petrel leverages Globus' identity management to provide researchers access to their data without requiring Argonne-specific credentials. Instead, users can authenticate with Petrel and access and distribute data world-wide using their own institutes credentials. Petrel provides cost-effective storage for researchers. Because Petrel is underpinned by Globus services it presents a sustainable storage model which researchers can rely on to exist and distribute data for prolonged periods of time.

DATA PUBLICATION AND DISCOVERY

The Globus data publication platform [17] provides a collection of loosely coupled services from which arbitrary publication pipelines can be constructed. These services—data management, persistent identifier management, and metadata indexing and discovery—can be used individually or collectively to address a wide range of publication needs. They allow for data to be made discoverable (via metadata search), verifiable (via checksums), and accessible (via Globus data access interfaces).

Globus data management and access capabilities allow for the "publication" of data in any Globus-accessible storage system: from local PCs, through to cloud object storage (e.g., Amazon S3), high performance file systems, and even archival tape storage. Globus data access model allows for user- and group-based access permissions to be applied to accessible data and for high performance (GridFTP) or web-based (HTTP) access to data. Users may therefore choose to publish data by moving it to a permanent location for immutable (and perhaps redundant) storage, or choose to publish data "in place" (where it currently resides).

An important step in data publication is the need to uniquely reference data irrespective of its location. A persistent identifier provides a long-lasting reference which can be resolved (using a resolution service) to locate the current location of that data. There are a range of persistent identifier schemes commonly used for digital artifacts including Digital Object Identifiers (DOIs), Archival Resource Keys (ARKs), and Handles. In each case, an identifier provider is responsible for minting unique identifiers and maintaining minimal metadata about that identifier (e.g., by whom it was created, when, and where the referenced object may be accessed). The Globus identifiers service provides a standard way to create and manage persistent identifiers (using various identifier providers) and to associate them with arbitrary data. Importantly, it also allows users to associate a checksum with identified data such that other users may subsequently validate the integrity of published data.

Having made data accessible and created a unique reference the final component of the data publication platform aims to make data discoverable. Globus Search provides a schemaless indexing and discovery model with fine grain access control. Thus, arbitrary metadata, adhering to any publication or domain- specific schema may be associated with a publication. Each metadata entry is assigned to a subject, a unique URI or identifier for a particular dataset. Each entity may also specify a visibility policy which restricts what users are able to view or query that metadata. Globus Search provides a powerful free-text query model that supports a expression of various query types (substring, range, wildcard, etc.) to discover matching metadata. Further, it can generate categorical facets and associated frequencies to intuitively summarize and query data.

THE MATERIALS DATA FACILITY (MDF): DATA PUBLICATION AND INDEXING

The MDF Data Publication service builds upon the Globus data publication service [18] and publication platform to enable users to publish data through a web user interface or a programmatically accessible API and to group similar publications into collections [7]. Key features of the deployed service include the capability to publish large datasets (our largest dataset to date is 1.85 TB); the ability to publish datasets with millions of files (our largest dataset by file count contains more than one million files); and the ability to publish data on distributed data stores. Each of these capabilities is coupled with features that allow users to add high-level descriptive metadata to each dataset (e.g., title, authors, institution, contact), materials-specific metadata following the NIST Materials Resource Vocabulary; and the ability to associate a permanent identifier with the dataset (e.g., a DOI or Handle) for scholarly citation.

The MDF Data Discovery service enables researchers to discover, query, browse and aggregate data that have been indexed by MDF. Entries in the MDF search index are comprised of descriptive information, materials-specific metadata (e.g., composition, crystal structure), and links to data harvested from a number of sources. These sources include the MDF Data Publication service, and data harvested from databases, services, and other sources from across the materials community. To date, we indexed 117 sources representing over 3.4M individually discoverable entries.

In order to facilitate usage of the data indexed by MDF, we have released the Forge Python client [19]. Forge enables users to search for data in MDF using materials-specific facets (e.g., elements, space group number) or general

metadata (e.g., authors, title) and aggregate search result data with only a few lines of code. Forge contains a host of helper functions that abstract from the user the need to understand the infrastructure in place behind MDF and instead allow them to focus on their task at hand. For example, users can perform a query and fetch the resultant files locally or to an analysis cluster with a single function call. These functions support data transfer via both HTTPS and Globus.

PIPELINES AND AUTOMATION

As discussed there is a growing need to automate a range of activities that are frequent, require rapid response, and require little human intervention. Light source science provides many such examples, consider a common workflow applied when using the APS: after data acquisition there is a need to apply apply quality control, assign identifiers, preprocess, move to data to a compute platform and perform analysis, postprocess, extract features, and eventually publish data to a repository for distribution. While these steps each exhibit need for automation, each presents opportunities to apply custom tools and locations depending on the beamline, users, and experiment being performed. Thus, it impractical to create a custom pipeline for every situation. Instead, higher level frameworks are needed that allow users to define and then outsource the management of such automated pipelines.

To empower scientists to automate their pipelines, it is important that user-friendly interfaces and models are provided. We believe that Trigger-Action Programming (TAP) provides a suitable intuitive and easy to use model for defining such pipelines. TAP is increasingly common in areas such as home automation and through online services such as if-this-then-that (IFTTT) that allow for the connection of online services (e.g., Twitter and Facebook). To experiment with such approaches we developed a prototype system called Ripple [20]. Ripple is comprised of two components: an end user agent that monitors a particular file system to capture events, and a cloud-hosted service that allows for the encoding of TAP rules and then execution of these rules based on events generated by the monitor. We used Ripple to develop a collection of TAP rules, primarily triggered by the creation of files on various machines, to perform actions like "move data from APS beamline to compute resources" or "analyze newly created files with program X." While this experience demonstrated the promise of such approaches, it also highlighted several shortcomings, including the need to construct complex pipelines of automated actions and to integrate a wide range of external event sources and action options.

Globus Automate

We have developed a prototype cloud-hosted automation service called Globus Automate that allows for the construction of robust automation pipelines (called "flows") that include both machine and human-in-the-loop operations. It is designed following an extensible, modular architecture via which external *event* sources and *actions* can be used in pipelines. Thus, developers may select a particular event source and type of event for which a pipeline should be invoked and assemble a series of one or more actions that should then be executed as a result of that event. Globus Automate is comprised of two core components: a web service that allows for the construction, execution, and sharing of automation flows; and a user agent that is used to capture data events on remote systems (e.g., file created, deleted, modified).

Globus Automate exposes a simple, declarative, JSON-based, state machine language for defining flows, based on the Amazon States Language [21]. This language 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.

An automation flow consists of one or more steps that invoke pre-defined actions. We define a REST API by which external services can be integrated with Globus Automate to perform actions. Services implementing these interfaces may be registered and then integrated into flows by any user, subject to service access rights. This API is similar to the popular if-this-then-that (IFTTT) action service API, with enhancements to accommodate asynchronous activities and the end-to-end security model provided by Globus Auth. When a flow is executing, Globus Automate will use the registered action API to invoke the action.

An automation flow can be triggered by a variety of events including data events, external service events, and periodic timers. We define a REST API that allows any service to produce events and to submit them to Globus Automate. When creating a flow, users define the type of event to be monitored and simple criteria for filtering events. In our prototype system we have created a modular event monitoring system that can be deployed on arbitrary storage systems and which leverages native storage system notification mechanisms to capture data events [20]. We have

demonstrated monitoring via the Linux inotify API [22], and achieved, via a hierarchical approach, ~10,000 events per second on a 1PB Lustre file system [23].

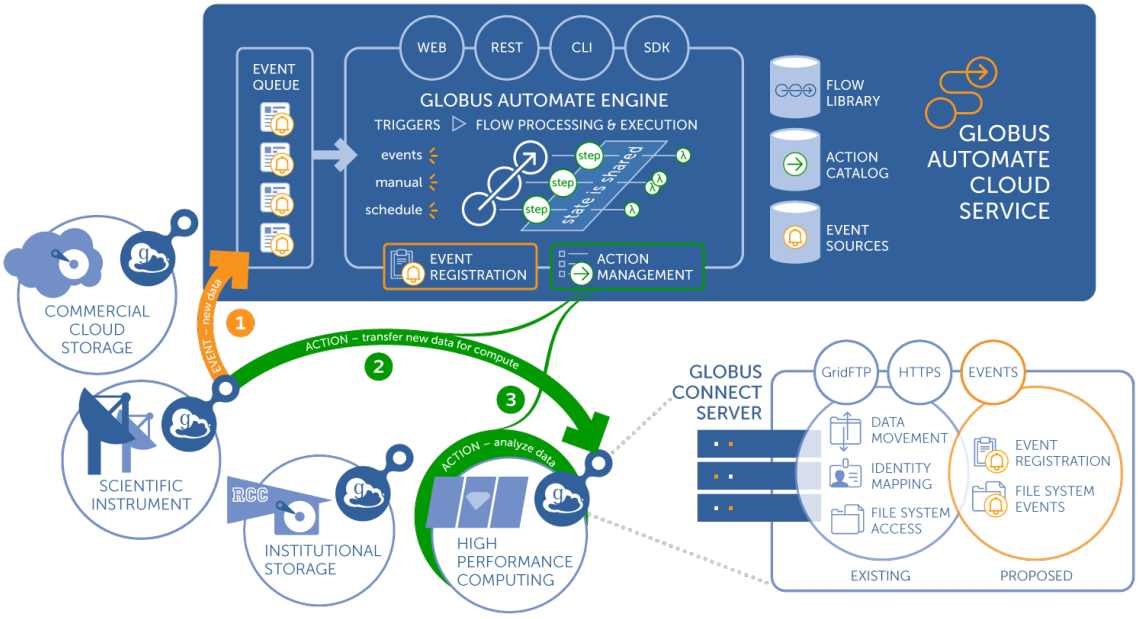


FIGURE 3. Globus Automate architecture, showing the cloud service at top, services that it manages below, and the extended Globus Connect internals at bottom right. Also, a flow comprising (1) an event (data created at instrument), (2) a transfer action, and (3) an analysis action.

Capturing and responding to events

Globus Automate is reliant on the ability to monitor and receive events from external services. We define a `globus_event` REST API to be implemented by any service that wants to produce events for consumption by Globus Automate and a simple queuing mechanism to reliably deliver events. Our goal is to make it simple for a service to produce and consume events, while also enabling performant and robust event delivery. The `globus_event` API, allows for a client (e.g., Globus Automate) to register with a service (e.g., Globus Connect) to receive events (e.g., file created) that match event-specific criteria (e.g., all files matching a path expression).

Globus Automate (or any other client) can use the `/introspect` method on a service to determine what event types are available from that service—and, for each event type, what parameters can be use to match events. For example, Globus Connect endpoints could offer events such as "file created" and "file deleted," each accepting a path matching expression as a parameter, so that Globus Automate can request notifications when files in a certain directory are created or deleted. Globus Automate will `/register` for events matching particular parameters, providing a URL to which the service should POST the events, along with a globally unique `<registration_id>` UUID. In the event of a communication error, Globus Automate can idempotently retry the same `/register` call, such that if the service already knows the `<registration_id>` it can respond as such. When events are no longer needed from a service, Globus Automate will `/cancel` the registration using the same `<registration_id>`.

Invoking and managing actions

The Globus Automate engine invokes an action in an external service to perform a step of an automation flow. Each external service must implement a simple REST API for introspection, execution, and management. When registering an external service, the service must define the actions that it supports, for each with a unique name, description, input and output parameters, and whether it runs synchronously (quickly, blocking) or asynchronously (slowly, non-

blocking). The action API focuses only on the information needed by Globus Automate. It may be implemented alongside any other REST API offered by a service.

The action model is simple: JSON document in, JSON document out, with a simple protocol to ensure reliable, once-and-only-once execution. When Globus Automate needs to run an action as part of a flow, it calls the service's action `/run` method with an input JSON document with the appropriate parameters, and a globally unique `<action_id>`. If Globus Automate does not receive a response from `/run`, for example due to a network error, it can repeat the `/run` call with the same `<action_id>`, allowing the service to recognize duplicate requests. In the case of asynchronous actions, Globus Automate will then periodically call `/status` to check if the action has completed. If the action has completed, `/status` will return the output JSON document with the appropriate parameters.

Once Globus Automate sees that an action has completed, it calls `/release` so that the service can forget about the `<action_id>`. If an action takes longer than a flow step's configured timeout, Globus Automate may call `/cancel`. If a service does not recognize the `<action_id>` in a `/cancel` or `/release`, it should assume it was previously canceled or released, and return an appropriate error. After some extended period (e.g., 30 days after completion), a service can forget any `<action_id>` that is not canceled or released.

Some actions may require human interaction. For example, as part of the Globus data publication platform we aim to implement a web-based form entry service, with configurable JSON schema and form configuration. This service will be used for such purposes as simple user prompts to complex metadata entry and curation. Such services will be integrated with Globus Automate as asynchronous actions.

The Globus Automate engine.

Globus Automate is responsible for executing end-to-end automation flows *reliably*, from event registration and processing for triggers to flow (i.e., state machine) execution and action invocation. After review of several workflow models, such as Amazon Simple Workflow Service [24], Netflix's Conductor [25], and Apache Airflow [26], we choose to leverage Amazon Step Functions (ASF) [27] for implementing and managing flows. ASF 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 Globus Automate to execute arbitrary actions.

Globus Automate uses parameterized Lambda functions [28] to execute actions. When creating a flow, Globus Automate calls `/introspect` to determine the input and output parameters for an action. It then creates parametrized Lambda functions for that action. For asynchronous actions multiple Lambda functions are created (i.e., start, monitor). When a user deploys an automation flow, Globus Automate translates the user's state machine with user-friendly action and event names into a state machine that invokes the appropriate Lambda functions and manages the flow of state between actions. Its deployment requires that the user specify both how its execution may be triggered and the source of the JSON document to be provided as input to the first step of the state machine. (The input JSON document may contain both static content and dynamic content provided by the trigger.)

Globus Automate wraps all logic for registering and interacting with external event- and action-providing services. Currently these capabilities are exposed as a REST API. In the future we intend to enhance the Globus web interface to make it easy for users to register action and event services, and to deploy, execute, and manage automation flows.

Automation Use Cases

We present two use cases in which Globus Automate is currently being applied. These use cases highlight the benefits of automation for analysis and publication activities conducted at light sources.

Neuroanatomy analysis: We applied a prototype version of Globus Automate to support neuroanatomy experiments that rely on X-ray microtomography at the Advanced Photon Source (APS) 32-ID beamline to characterize the neuroanatomical structure of large (cm) unsectioned brain volumes [29], as shown in FIGURE 4. Datasets, generated at >20GB per minute, are processed with a complex reconstruction pipeline comprising many machines, tools, and services. Globus Automate is used to move data from beamline to remote computers, execute Automo [30] and TomoPy [31] to generate preview images. Simultaneously, machine learning models are applied (automatically) to estimate the center of rotation. This value and the resulting images are returned to beamline scientists to guide instrument positioning. Once the center value has been verified the flow continues by performing the full reconstruction. Results are moved to persistent storage for distribution, cataloged with provenance metadata, and precomputed to be visualized with Neuroglancer [32].

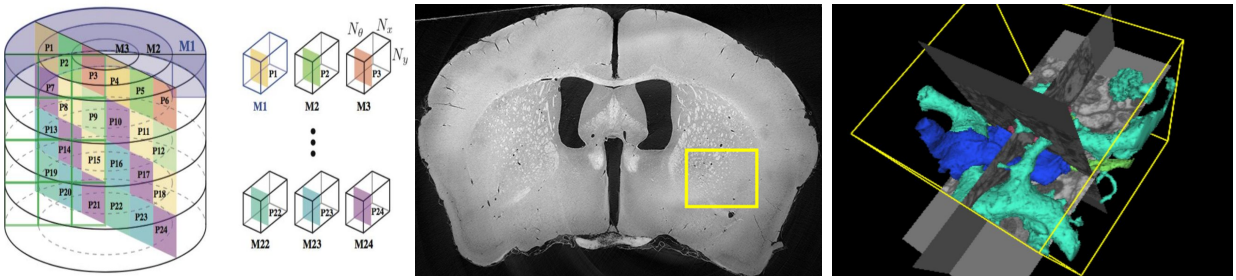


FIGURE 4. From left to right: Schematic diagram showing data acquisition process; coronal cross section of a reconstructed brain; and 3D rendering of a reconstructed brain volume using Neuroglancer.

Data publication: We use the prototype system to develop flexible data publication pipelines for MDF that combine machine and human steps. Our aim here is to provide an automated deposit mechanism in which users define a dataset to be published (via a reference to a Globus endpoint) and the automation flow manages the steps required to make that data accessible in MDF. In one example, the workflow will transfer data to an intermediary location for processing and also to a permanent storage location for publication. That dataset will be processed with a community code to extract materials-specific metadata from the dataset. This metadata will be used to register the dataset in in the NIST Materials Resource Registry, Citrination, or other community registries, and also to index the dataset in the MDF discovery index. Before the data is to be made accessible the flow will notify MDF curators of its arrival and wait until they have approved the dataset for publication before progressing. A unique identifier will be created for the dataset, using user-defined and automatically extracted metadata. Finally, access permissions will be modified such that the metadata is discoverable and the dataset is accessible.

TRANSFORMATION, ANALYSIS, AND VISUALIZATION

The requirement to transform and analyze results is common across light source science. Yet, the specific analyses and cyberinfrastructure required to support analyses are often developed by beamline scientists and are infrequently shared amongst researchers working in similar fields. This is clearly inefficient as it requires beamline scientists to obtain expertise in several unnecessary areas (e.g., developing cyberinfrastructure to support real-time machine learning analyses) and leads to significant duplication. As scientists increasingly look towards new machine learning approaches for various tasks (e.g., center finding, error detection, image segmentation), these challenges are likely to become more significant. To address these challenges we are developing the Data and Learning Hub for Science (DLHub) which aims to support the deposit and sharing of machine learning models, on-demand inference using these models, and a high performance computing environment for training models.

DATA AND LEARNING HUB FOR SCIENCE (DLHUB)

DLHub is a service to promote, simplify, and speed the widespread adoption and usage of machine learning and deep learning techniques by researchers in disciplines ranging from materials science to chemistry, physics, cosmology, and biology. The co-emergence of large amounts of available datasets, movement towards cohesive data services, and new machine learning capabilities, creates a unique opportunity to leverage and integrate these data streams to allow for machine learning techniques to guide and, indeed, lead discovery efforts.

Thus, we are developing DLHub a self-service platform for publishing, applying, and creating new machine learning models. DLHub will provide: 1) publication capabilities to make models more discoverable, citable, and reusable; 2) the ability to easily run or test existing models; and 3) links to the data and computing infrastructure to re-train models for new applications. Users will benefit from DLHub in many ways. Data scientists can publish their models (i.e., architectures and weights) and methods. Materials scientists can apply existing models to new data with ease (e.g., by querying a prediction API for a deployed mode) and create new models with state-of-the-art techniques. Together, these capabilities will lower barriers to employing machine learning, making it easier for researchers to discover and benefit from the most recent advances in machine learning.

DLHUB ARCHITECTURE

DLHub is operated as a cloud-hosted service that offers a REST API to publish, search, and invoke models. DLHub serves models and transformation codes across remote *execution sites*. The REST API is secured with Globus Auth [33] and offers endpoints for publishing, searching, and invoking models. We provide a Python SDK to simplify use of the service. In future work, we aim to develop a user interface for many of these tasks. We have defined a simple metadata schema for describing models. Users who wish to deposit a model must provide this metadata and a container with their trained model.

When invoking a model, users must first select the specific model to use (discovered via searching the list of published models) and provide a set of data to be passed to the model. At present, all input data is described in JSON. DLHub then executes the model for each item in the input JSON and returns JSON-based results to the user.

DLHub relies on the Kubernetes container orchestration system to manage the deployment of containerized codes and models on an execution site. Models and transformation logic are wrapped in Docker containers for three primary reasons: 1) to standardize their execution interface, regardless of implementation or language; 2) to enable the encapsulation of their vastly different software requirements; and 3) to allow them to be scaled by deploying multiple instances of the containers. In DLHub, a containerized model is referred to as a *servable*. When a code is published in DLHub, it is automatically packaged with a custom DLHub Python library to provide a standard execution interface to facilitate remote invocation. DLHub Servables can be deployed at execution sites and invoked directly through DLHub. A single servable may serve results for many inference jobs and the same servable may be deployed several times to meet user demand.

Each execution site is connected to the DLHub service through ZeroMQ (ZMQ) channels, providing a high-performance and reliable mechanism to transmit jobs. We use the Parsl parallel scripting library [34] to manage the execution infrastructure by deploying and managing servables. Each site includes a Parsl *foreman* and a ZMQ receiver. The ZMQ receiver is used to receive inference jobs from the DLHub service. The Parsl foreman is then responsible for taking these inference jobs and executing them on one or more servables. This task includes submitting work to existing servables or deploying new ones if one is not available. Parsl uses IPyParallel and a pilot job model to perform remote execution within a container. Each container is deployed with an *ipengine* that connects back to the Parsl foreman to receive jobs. This two-tier design enables multi-level caching, with both Parsl and cloud-hosted caches enhancing response time. Requests can also be batched at the cloud level before being transmitted to execution sites to improve throughput. While the primary mode for execution is via the DLHub service, for low latency jobs we also allow users to establish a direct connection to the execution site. This model allows for DLHub to be easily integrated in light source workflows by streaming data from the acquisition machine to DLHub for transformation or inference.

DLHub USE CASES

We present two separate uses cases of the DLHub service each highlighting key capabilities. DLHub analysis pipelines were created for these use cases to simplify invocation on PetrelKube and within Amazon Web Services.

Prediction of Metallic Glass Forming Compositions. Metallic glasses are an important class of materials that promise improved durability, corrosion resistance, and mechanical behavior in harsh environments. However, discovery of metallic glass forming materials is particularly challenging due to the lack of theories that can predict which alloys can be formed into metallic glasses. Ren et al. trained a machine learning model to predict glass-forming ability using data from the materials literature, and used it to identify new Co-V-Zr metallic glasses via high-throughput experiments at Stanford Linear Accelerator Laboratory (SLAC). By adding the SLAC data to the training set of the model, they found its accuracy improved for many other types of alloys.

The associated machine learning model was contributed to DLHub as a serialized scikit-learn Random Forest model. Containers were prepared to serve the model, which take a list of elements as inputs return the predicted glass-forming ability for that alloy system. An example Jupyter notebook was then prepared to use the model in DLHub to generate ternary plots showing areas of highest metallic glass forming likelihood 5.

Batch Classification of Beamline X-Ray Scattering Data. Classifying streaming data or archived data from beamlines at national user facilities promises to aide future data discovery, promote data reuse, speed analyses, and to allow users to receive near real-time feedback on the state of their experiments. For example, if a model is able to automatically determine that a beam is misaligned, an experimental session may be saved from waste by user intervention.

(a)

```
dl = DLHub()
servable_id = dl.get_id_by_name("glass_forming")
elems = ["Zr","Co","V"]

res = dl.infer(servable_id, {"data":elems})
```

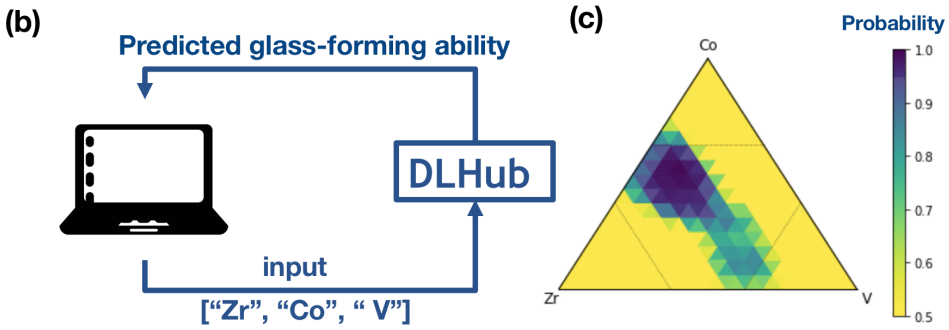
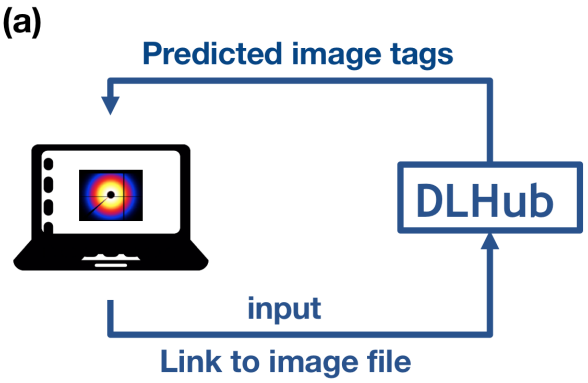


FIGURE 5. Predicting metallic glass forming ability with DLHub. (a) Example code showing instantiation of DLHub Python client, selection of model, and inference against that model. (b) A user submits a set of three elements and sends to DLHub service for prediction. User receives the predicted glass-forming ability as an output. (c) Displaying the results of the model for Zr, Co, V inputs as a ternary diagram.

The classification model here enables the multi-label classification of X-Ray scattering data with 17 potential labels (e.g., "beam off image," "FCC," "BCC," "polycrystalline," "high background," "strong scattering") [35]. The classification model is a Tensorflow 1.4 [36] implementation in Python 2.7 of a convolutional neural network following the ResNet architecture. Original training data comprised simulated data and experimental data tagged by experts collected at National Synchrotron Light Source (NSLS) at Brookhaven National Laboratory as described in Wang et al. This pretrained model and dataset were contributed to DLHub by Wang and Yager et al.; the source code is available on Github[37]. Containers were created to serve the model and to transform input image files to the required input format of 256×256 NumPy arrays.



(b)

	probability	tags
1	0.999780	Beam Off Image
6	0.745707	Halo
13	0.364863	Strong scattering
7	0.082574	High background
12	0.027826	Ring
8	0.012137	Higher orders
15	0.008106	Weak scattering
9	0.001964	Linear beamstop
16	0.001744	Wedge beamstop
2	0.000982	Circular Beamstop

FIGURE 6. Tagging X-Ray scattering images with DLHub. (a) User sends pointers to X-Ray scattering images to the DLHub service. (b) DLHub returns a list of most likely tags for the image.

RELATED WORK

The requirement for automation in science has been well studied studied [38, 39, 20] and many automation systems have been developed. Coles et al. [40, 41] developed an automated small molecule crystallography service to manage the end-to-end-flow from sample receipt to results dissemination. The SPOT framework [39] powers similar pipelines for Lawrence Berkeley National's Advanced Light Source. The Rapid Experimental Analysis (REXAN) system [14] developed at Pacific Northwest National Laboratory, pipelines developed at Argonne for high-speed tomography [12, 13] and high energy diffraction microscopy [42], and the high data rate processing and analysis initiative of the Helmholtz Association [43] are other examples.

Scientific workflow engines, such as Galaxy [44], Parsl [34], Pegasus [45], Swift [46], and Taverna [47], enable the fault-tolerant and reproducible execution of pipelines comprising various tools and services. While some, such as Taverna, support Web service composition, these tools are generally designed to facilitate dataflow modeling and the execution of sequences of tools. In contrast, Automate aims to orchestrate a mix of custom software and human-in-the-loop tasks.

The need to reliably catalog and publish data has led to the development of several tools [18, 48, 49]. Indeed, many of these platforms are in widespread use (e.g., Dataverse [50], DuraCloud [51], and figshare [52]). However, current approaches have limitations with needs of light source science. First, these systems are primarily designed to support small scale data publication and do not scale to the large data sizes typical in light source science. Second, these systems are primarily software services and thus they cannot be easily built upon or integrated into existing software. Finally, each system is designed for a specific community or data publication scenario and is therefore difficult to apply to different scenarios (e.g., domain-specific metadata, various standards, data types, and curation requirements). MDF, in contrast, provides a generalizable platform that has been shown able to accommodate diverse communities, datasets, and publication scenarios by outsourcing the tools required to mint identifiers, catalog metadata, and search over distributed datasets to specialized services.

As machine learning becomes increasingly pervasive there is a growing requirement for services to publish, share, and serve models [53]. High performance, low latency model serving is necessary to facilitate the integration of machine learning solutions into everyday tasks. To this end, various model serving platforms have been developed [54, 55, 56, 57]. Clipper [55] and Tensorflow Serving [56] use RPC-based invocations solutions to minimize the overhead when invoking models, providing extremely low latency serving solutions. In addition, Tensorflow serving facilitates extreme-scale deployment of models, enabling integration in large-scale platforms, such as shopping carts. One limitation of these systems is their strict enforcement of the types of models and transformation logic that can be served. For example, Tensorflow Serving restrict servable types to be those that can be exported as Tensorflow graphs and provides minimal support for transformation codes.

Similarly, model publication services, such as Kipoi [58] and ModelHub [59], have been established to publish models along with their trained weights and metadata describing their inputs and outputs, graph design, training and validation datasets, accuracy, as well as information relating to citation and authorship. Such tools are necessary to disseminate machine learning models and make them available to the community.

DLHub differs from these solutions by combining publishing capabilities, model serving functionality, and the resources required to train and invoke models on-demand. DLHub is designed to accommodate arbitrary models and servables through containerization, removing the restrictions of serving systems such as Tensorflow Serving. Leveraging descriptive metadata, inspired by Kipoi [58], and the Globus Search service, DLHub enables the publication of models such that users can search and find models, retrain them on specific datasets, and invoke them against data on-demand.

ACKNOWLEDGMENTS

This work was supported in part by DOE contract DE-AC02-06CH11357 and by award 70NANB14H012 from the U.S. Department of Commerce, National Institute of Standards and Technology as part of the Center for Hierarchical Materials Design (CHiMaD). We are grateful to colleagues at the Argonne Photon Source and elsewhere for many helpful discussions, in particular Jon Almer, Tekin Bicer, Francesco De Carlo, Doğa Gürsoy, Ray Osborn, Dula Parkinson, Nicholas Schwarz, Hemant Sharma, Stephen Streiffer, Brian Toby, Stefan Vogt, and Justin Wozniak.

REFERENCES

[1] B. H. Toby, D. Gürsoy, F. De Carlo, N. Schwarz, H. Sharma, and C. J. Jacobsen, "Practices and standards for data and processing at the APS," Synchrotron Radiation News **28**, 15–21 (2015).

[2] S. Streiffer, S. Vogt, P. Evans, *et al.*, "Early science at the upgraded Advanced Photon Source," Tech. Rep. (Argonne National Laboratory, 2015) `https://bit.ly/2x4Vb2i`.

[3] C. Pralavorio, LHC Season 2: CERN computing ready for data torrent, May 2015, `https://home.cern/about/updates/2015/06/lhc-season-2-cern-computing-ready-data-torrent`.

[4] M. Hiraki, S. Watanabe, N. Phonda, Y. Yamada, N. Matsugaki, N. Igarashi, Y. Gaponov, and S. Wakatsuki, "High-throughput operation of sample-exchange robots with double tongs at the Photon Factory beamlines," Journal of Synchrotron Radiation **15**, 300–303 (2008).

[5] B. H. Toby, Y. Huang, D. Dohan, D. Carroll, X. Jiao, L. Ribaud, J. A. Doebbler, M. R. Suchomel, J. Wang, C. Preissner, *et al.*, "Management of metadata and automation for mail-in measurements with the APS 11-BM high-throughput, high-resolution synchrotron powder diffractometer," Journal of Applied Crystallography **42**, 990–993 (2009).

[6] J. M. Wozniak, K. Chard, B. Blaiszik, R. Osborn, M. Wilde, and I. Foster, "Big data remote access interfaces for light source science," in *2nd International Symposium on Big Data Computing* (IEEE, 2015), pp. 51–60.

[7] B. Blaiszik, K. Chard, J. Pruyne, R. Ananthakrishnan, S. Tuecke, and I. Foster, "The Materials Data Facility: Data services to advance materials science research," Journal of Materials **68**, 2045–2052 (2016).

[8] Y. Wang, F. De Carlo, I. Foster, J. Insley, C. Kesselman, P. Lane, G. von Laszewski, D. C. Mancini, I. McNulty, M.-H. Su, *et al.*, "Quasi-real-time x-ray microtomography system at the Advanced Photon Source," in *Developments in X-Ray Tomography II*, Vol. 3772 (International Society for Optics and Photonics, 1999), pp. 318–328.

[9] Y. Wang, F. De Carlo, D. C. Mancini, I. McNulty, B. Tieman, J. Bresnahan, I. Foster, J. Insley, P. Lane, G. von Laszewski, *et al.*, "A high-throughput x-ray microtomography system at the Advanced Photon Source," Review of Scientific Instruments **72**, 2062–2068 (2001).

[10] G. Von Laszewski, M. L. Westbrook, C. Barnes, I. Foster, and E. M. Westbrook, "Using computational grid capabilities to enhance the capability of an X-ray source for structural biology," Cluster Computing **3**, 187–199 (2000).

[11] G. Von Laszewski, J. A. Insley, I. Foster, J. Bresnahan, C. Kesselman, M. Su, M. Thiebaux, M. L. Rivers, S. Wang, B. Tieman, and I. McNulty, "Real-time analysis, visualization, and steering of microtomography experiments at photon sources," in *9th SIAM Conference on Parallel Processing* (1999).

[12] T. Bicer, D. Gürsoy, R. Kettimuthu, F. D. Carlo, G. Agrawal, and I. T. Foster, "Rapid tomographic image reconstruction via large-scale parallelization," in *Euro-Par 2015: 21st International Conference on Parallel and Distributed Computing* (2015), pp. 289–302.

[13] T. Bicer, D. Gursoy, R. Kettimuthu, I. T. Foster, B. Ren, V. De Andrede, and F. De Carlo, "Real-time data analysis and autonomous steering of synchrotron light source experiments," in *13th International Conference on e-Science* (IEEE, 2017), pp. 59–68.

[14] M. Thomas, K. Kleese-van Dam, M. J. Marshall, A. Kuprat, J. Carson, C. Lansing, Z. Guillen, E. Miller, I. Lanekoff, and J. Laskin, "Towards adaptive, streaming analysis of x-ray tomography data," Synchrotron Radiation News **28**, 10–14 (2015).

[15] K. Chard, S. Tuecke, and I. Foster, "Efficient and secure transfer, synchronization, and sharing of big data," IEEE Cloud Computing **1**, 46–55Sept (2014).

[16] S. Delagenière, P. Brenchereau, L. Launer, A. W. Ashton, R. Leal, S. Veyrier, J. Gabadinho, E. J. Gordon, S. D. Jones, K. E. Levik, *et al.*, "ISPyB: An information management system for synchrotron macromolecular crystallography," Bioinformatics **27**, 3186–3192 (2011).

[17] R. Ananthakrishnan, B. Blaiszik, K. Chard, R. Chard, B. McCollam, J. Pruyne, S. Rosen, S. Tuecke, and I. Foster, "Globus platform services for data publication," in *Practice and Experience in Advanced Research Computing (PEARC)* (IEEE, 2018).

[18] K. Chard, J. Pruyne, B. Blaiszik, R. Ananthakrishnan, S. Tuecke, and I. Foster, "Globus data publication as a service: Lowering barriers to reproducible science," in *11th International Conference on e-Science* (IEEE, 2015), pp. 401–410.

[19] J. Gaff, B. Blaiszik, and L. Ward, MDF Forge Python package, `https://github.com/materials-data-facility/forge`. Accessed June 15, 2018.

[20] R. Chard, K. Chard, J. Alt, D. Y. Parkinson, S. Tuecke, and I. Foster, "Ripple: Home automation for research data management," in *37th IEEE International Conference on Distributed Computing Systems Workshops* (2017), pp. 389–394.

[21] Amazon States Language, , `https://states-language.net/spec.html`. Accessed April 1, 2018.

[22] Linux programmers manual: inotify API, `http://man7.org/linux/man-pages/man7/inotify.7.html`. Accessed August 1, 2016 (August 2016).

[23] A. K. Paul, S. Tuecke, R. Chard, A. R. Butt, K. Chard, and I. Foster, "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* (ACM, 2017), pp. 49–54.

[24] Amazon Simple Workflow Service, , `https://aws.amazon.com/swf/`. Accessed April 1, 2018.

[25] Conductor, `https://netflix.github.io/conductor/`. Accessed April 1, 2018.

[26] Airflow, `https://airflow.apache.org/`. Accessed April 1, 2018.

[27] Amazon Step Functions, , `https://aws.amazon.com/step-functions`. Accessed April 1, 2018.

[28] Amazon Lambda, , `https://aws.amazon.com/lambda`. Accessed April 1, 2018.

[29] N. Kasthuri, K. J. Hayworth, D. R. Berger, R. L. Schalek, J. A. Conchello, S. Knowles-Barley, D. Lee, A. Vázquez-Reina, V. Kaynig, T. R. Jones, *et al.*, "Saturated reconstruction of a volume of neocortex," Cell **162**, 648–661 (2015).

[30] F. De Carlo, Automo, `https://automo.readthedocs.io`. Accessed June 1, 2018.

[31] D. Gürsoy, F. De Carlo, X. Xiao, and C. Jacobsen, "TomoPy: A framework for the analysis of synchrotron tomographic data," Journal of Synchrotron Radiation **21**, 1188–1193 (2014).

[32] Neuroglancer, `https://github.com/google/neuroglancer`. Accessed June 1, 2018.

[33] S. Tuecke, R. Ananthakrishnan, K. Chard, M. Lidman, B. McCollam, S. Rosen, and I. Foster, "Globus Auth: A research identity and access management platform," in *12th International Conference on e-Science* (IEEE, 2016), pp. 203–212.

[34] Y. Babuji, A. Brizius, K. Chard, I. Foster, D. S. Katz, M. Wilde, and J. Wozniak, Introducing Parsl: A Python Parallel Scripting Library, August 2017, `https://doi.org/10.5281/zenodo.891533`.

[35] B. Wang, K. Yager, D. Yu, and M. Hoai, "X-ray scattering image classification using deep learning," in *Winter Conference on Applications of Computer Vision* (IEEE, 2017), pp. 697–704.

[36] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "TensorFlow: A system for large-scale machine learning." in *OSDI*, Vol. 16 (2016), pp. 265–283.

[37] B. Wang, X-ray scattering classifier Resnet implementation, `https://github.com/Boyu-Wang/Xray_Scattering_Resnet`. Accessed June 15, 2018.

[38] R. D. King, J. Rowland, S. G. Oliver, M. Young, W. Aubrey, E. Byrne, M. Liakata, M. Markham, P. Pir, L. N. Soldatova, *et al.*, "The automation of science," Science **324**, 85–89 (2009).

[39] J. Deslippe, A. Essiari, S. J. Patton, T. Samak, C. E. Tull, A. Hexemer, D. Kumar, D. Parkinson, and P. Stewart, "Workflow management for real-time analysis of lightsource experiments," in *9th Workshop on Workflows in Support of Large-Scale Science* (IEEE, 2014), pp. 31–40.

[40] S. J. Coles, J. G. Frey, M. B. Hursthouse, M. E. Light, K. E. Meacham, D. J. Marvin, and M. Surridge, "ECSES–Examining crystal structures using e-science: A demonstrator employing web and grid services to enhance user participation in crystallographic experiments," Journal of Applied Crystallography **38**, 819–826 (2005).

[41] S. J. Coles, J. G. Frey, M. B. Hursthouse, M. E. Light, A. J. Milsted, L. A. Carr, D. DeRoure, C. J. Gutteridge, H. R. Mills, K. E. Meacham, *et al.*, "An e-science environment for service crystallography from submission to dissemination," Journal of Chemical Information and Modeling **46**, 1006–1016 (2006).

[42] J.-S. Park, X. Zhang, H. Sharma, P. Kenesei, D. Hoelzer, M. Li, and J. Almer, "High-energy synchrotron x-ray techniques for studying irradiated materials," Journal of Materials Research **30**, 1380–1391 (2015).

[43] R. Gehrke, A. Kopmann, E. Wintersberger, and F. Beckmann, "The high data rate processing and analysis initiative of the Helmholtz Association in Germany," Synchrotron Radiation News **28**, 36–42 (2015).

[44] J. Goecks, A. Nekrutenko, and J. Taylor, "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences," Genome biology **11**, p. R86 (2010).

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

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

[47] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, *et al.*, "Taverna: a tool for the composition and enactment of bioinformatics workflows," Bioinformatics **20**, 3045–3054 (2004).

[48] M. Wilkinson, M. Dumontier, I. Aalbersberg, G. Appleton, M. Axton, A. Baak, *et al.*, "The FAIR guiding principles for scientific data management and stewardship," Scientific Data **3**, p. 160018 (2016).

[49] M. Costello, "Motivating online publication of data," BioScience **59**, 418–427 (2009).

[50] M. Crosas, "The Dataverse Network: An open-source application for sharing, discovering and preserving data," D-Lib Magazine **17** (2011).

[51] DuraCloud, `http://duracloud.org/`. Accessed April 1, 2018.

[52] figshare, `https://figshare.com/`. Accessed March 22, 2018.

[53] A. Kumar, M. Boehm, and J. Yang, "Data management in machine learning: Challenges, techniques, and systems," in *International Conference on Management of Data* (ACM, 2017), pp. 1717–1722.

[54] D. Crankshaw, P. Bailis, J. E. Gonzalez, H. Li, Z. Zhang, M. J. Franklin, A. Ghodsi, and M. I. Jordan, "The missing piece in complex analytics: Low latency, scalable model management and serving with Velox," arXiv preprint arXiv:1409.3809 (2014).

[55] D. Crankshaw, X. Wang, G. Zhou, M. J. Franklin, J. E. Gonzalez, and I. Stoica, "Clipper: A low-latency online prediction serving system." in *NSDI* (2017), pp. 613–627.

[56] Tensorflow serving, `https://github.com/tensorflow/serving`. Accessed June 1, 2018.

[57] Amazon sagemaker: Developer guide, 2017, `http://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-dg.pdf`. Accessed June 13, 2018.

[58] Kipoi: Model zoo for genomics, .

[59] H. Miao, A. Li, L. S. Davis, and A. Deshpande, "Towards unified data and lifecycle management for deep learning," in *33rd International Conference on Data Engineering* (IEEE, 2017), pp. 571–582.

[60] J. Brase, "Datacite – a global registration agency for research data," in *4th International Conference on Cooperation and Promotion of Information Resources in Science and Technology* (2009), pp. 257–261.

[61] I. Foster, R. Ananthakrishnan, B. Blaiszik, K. Chard, R. Osborn, S. Tuecke, M. Wilde, and J. Wozniak, "Networking materials data: Accelerating discovery at an experimental facility," in *Big Data and High Performance Computing* (2015).

[62] I. Foster, B. Blaiszik, K. Chard, and R. Chard, "Software defined cyberinfrastructure," in *37th International Conference on Distributed Computing Systems* (IEEE, 2017), pp. 1808–1814.

[63] R. Chard, K. Chard, S. Tuecke, and I. Foster, "Software defined cyberinfrastructure for data management," in *13th International Conference on e-Science* (IEEE, 2017), pp. 456–457.

[64] F. Ren, Discovering metallic glasses with HiTp experiment and machine learning, `https://github.com/fang-ren/Discover_MG_CoVZr`. Accessed June 1, 2018.

[65] L. Ward, A. Agrawal, A. Choudhary, and C. Wolverton, "A general-purpose machine learning framework for predicting properties of inorganic materials," npj Computational Materials **2**, p. 16028 (2016).

[66] F. Ren, L. Ward, T. Williams, K. J. Laws, C. Wolverton, J. Hattrick-Simpers, and A. Mehta, "Accelerated discovery of metallic glasses through iteration of machine learning and high-throughput experiments," Science advances **4**, p. eaaq1566 (2018).