CellPress
OPEN ACCESS

Article

Linking scientific instruments and computation: Patterns, technologies, and experiences

Rafael Vescovi,[1] Ryan Chard,[1] Nickolaus D. Saint,[6] Ben Blaiszik,[1,6] Jim Pruyne,[1,6] Tekin Bicer,[1,3] Alex Lavens,[4] Zhengchun Liu,[1] Michael E. Papka,[2,7] Suresh Narayanan,[3] Nicholas Schwarz,[3] Kyle Chard,[1,5] and Ian T. Foster[1,5,*]
[1]Data Science and Learning Division, Argonne National Laboratory, 9700 S. Cass Ave., Lemont, IL 60439, USA
[2]Argonne Leadership Computing Facility, Argonne National Laboratory, 9700 S. Cass Ave., Lemont, IL 60439, USA
[3]X-ray Science Division, Argonne National Laboratory, 9700 S. Cass Ave., Lemont, IL 60439, USA
[4]Structural Biology Center, Argonne National Laboratory, 9700 S. Cass Ave., Lemont, IL 60439, USA
[5]Department of Computer Science, University of Chicago, 5730 S. Ellis Ave., Chicago, IL 60615, USA
[6]Globus, University of Chicago, 5730 S. Ellis Ave., Chicago, IL 60615, USA
[7]Department of Computer Science, University of Illinois Chicago, 1200 W. Harrison St., Chicago, IL 60607, USA
*Correspondence: foster@anl.gov
https://doi.org/10.1016/j.patter.2022.100606

THE BIGGER PICTURE The industrial revolution transformed society via large-scale automation of manufacturing. Today, AI- and robotics-driven automation of scientific research seems set to usher in a new era of accelerated discovery. But just as the industrial revolution depended on new replicable and scalable manufacturing processes and methods for delivering the copious mechanical power required by those processes, so the automated discovery revolution demands new methods for implementing research automation processes and for connecting those processes to computing and data power. We present here new methods that address these essential needs by allowing scientists to capture common automation patterns in reusable flows and to embed such flows in a global trust, data, and computing fabric that enables instant access to powerful AI, simulation, and other computational capabilities. We use examples from synchrotron light sources to show how these methods can be realized in software and applied at scale.

1 2 **3** 4 5 **Production:** Data science output is validated, understood, and regularly used for multiple domains/platforms

SUMMARY

Powerful detectors at modern experimental facilities routinely collect data at multiple GB/s. Online analysis methods are needed to enable the collection of only interesting subsets of such massive data streams, such as by explicitly discarding some data elements or by directing instruments to relevant areas of experimental space. Thus, methods are required for configuring and running distributed computing pipelines—what we call flows—that link instruments, computers (e.g., for analysis, simulation, artificial intelligence [AI] model training), edge computing (e.g., for analysis), data stores, metadata catalogs, and high-speed networks. We review common patterns associated with such flows and describe methods for instantiating these patterns. We present experiences with the application of these methods to the processing of data from five different scientific instruments, each of which engages powerful computers for data inversion,model training, or other purposes. We also discuss implications of such methods for operators and users of scientific facilities.

INTRODUCTION

Humphry Davy observed that “[n]othing tends so much to the advancement of knowledge as the application of a new instrument.”[1] Today, powerful new instruments such as upgraded synchrotron light sources,[2–5] free-electron lasers,[6] microscopes,[7,8] telescopes,[9] field laboratories,[10] and robotic laboratories[11–13] provide exciting new means to study phenomena in a broad range of scientific disciplines.

The power of these new instruments derives from their ability to probe reality rapidly and at fine spatial and temporal scales. In so doing, they can generate data at rates (multi-GB/s) and volumes (100 + TB/day[14,15]) that demand online computing, both to extract interesting information from data streams and to enable rapid configuration and steering of instruments to maximize information gained during scarce experimental time. Tight coupling with powerful computing resources, such as data center clusters, high-performance computing (HPC), or artificial intelligence (AI) accelerators, is often needed both to process this fire hose of data and to enable real-time feedback to experiments.

Such coupling requires flexible methods for coordinating actions and resources across diverse experimental and computing environments. We present common patterns for processing data from scientific instruments and describe tools that enable convenient specification of high-level flows linking diverse actions and the flexible mapping of such flows onto diverse physical resources to meet reliability, scalability, timeliness, and security goals as an experiment runs. (We use the term flow rather than the over-used workflow to emphasize our interest in capturing specialized data-processing patterns associated with scientific instrumentation.) Specifically, we (1) identify common patterns encountered when scientists develop and run online data processing flows; (2) show how Globus automation services[16,17] can be used to capture such patterns; (3) present experiences applying such methods in five different application scenarios; and (4) examine the implications of such flows for both computing and experimental facilities.

RESULTS

Patterns for integration of instruments and computing

Exponential growth in the rate at which instruments can perform measurements requires corresponding exponential improvements in the speed at which the resulting data are processed. This means increasing use of automation and computation at every stage in the experimental process, including steps that were previously not rate limiting and thus could be performed manually, such as recording and interpreting results and configuring the next experiment. New methods may be needed to capture data at high rates, extract interesting events in high rate streams, identify and filter out uninteresting phenomena, detect and/or correct errors, design further experiments, and perform simulations to choose between alternative experimental configurations—and to combine many such steps into automated experiment management flows.

As in other areas of design, the identification of recurring patterns[18,19] can contribute to cost reduction and performance improvement. A design pattern captures a solution to a problem or class of problems in a reusable form via documentation of its purpose/intent, applicability, solution structure, and sample implementations. In this section, we enumerate patterns we and others have observed when processing data from scientific instruments and review the nature of the resources required to implement the patterns.

What: Actions that are frequently included in flows

Data collection. Data collection captures data and associated metadata generated at high speeds, in unconventional formats, and on specialized devices and makes those data available to subsequent analyses.

Data reduction. Data reduction reduces the volume of data to be processed and/or stored in other steps by applying either general-purpose compression[20,21] or domain-specific feature detection (e.g., to find diffraction peaks in X-ray imaging[22,23]).

Data inversion. Sophisticated computations are often required to convert sensor data into useful forms: for example, to generate a three-dimensional (3D) or 4D representation from multiple 2D images[24,25] or a 2D image from diffraction patterns. This step may be performed incrementally while data are collected or after all data are available.

Machine-learning (ML) model training. In this increasingly popular approach to data reduction, previously collected data (from current or prior experiments and/or simulations) are used to train ML models to recognize interesting phenomena for data reduction or rapid response.[26–30]

Experiment steering. Even better than discarding uninteresting data is to collect only interesting data in the first place. Scientists may use analyses of results from current or prior experiments to determine what experiment or measurement to perform next. Steering can range from fine-grained control of apparatus, such as taking (more) data from one part of a sample, to coarse-grained (e.g., choosing the next sample). Experiment steering can use design of experiment methods or more sophisticated active learning,[31] Gaussian processes,[32] Bayesian optimization,[33] reinforcement learning,[34] or other methods.

Coupled simulation. Computational simulation can be used during experiment steering to eliminate (or prioritize) experimental configurations.

Data storage and publication. A flow may include steps to organize and store data and associated metadata (e.g., concerning experimental sample, configuration of apparatus, data processing steps) so as to make it findable, accessible, interoperable, and reusable (FAIR).[35]

Where: Alternative places to perform flow tasks

Analysis methods such as those just described can easily overwhelm instrument computers. Indeed, some analyses can consume tens or even hundreds of thousands of cores,[36,37] albeit typically in a bursty manner. Similarly, experiments can generate petabytes. The aggregate compute and storage demand across a research institution or multi-instrument research facility can be large, and shared (rather than per-instrument) computing facilities become attractive or even essential to exploit economies of scale in capital and operations costs.

A public cloud is a credible option for certain instrument workloads,[38] but data center systems can be more cost effective,[39] especially when high-capacity, low-latency networks can support high data rate instruments and experiment steering. Custom silicon may be required for certain data processing steps.[40,41] Specialized accelerators may be used for tasks such as ML model training and inference.[42–44]

When demand outstrips supply, adaptive methods may be used to direct compute and storage requests to different resources, prioritize certain tasks, and substitute alternative computational methods. In effect, computation may occur across a computing continuum[45–47] that extends from data acquisition computers co-located with experiments to powerful clusters in data centers. For a given flow, computation may occur at multiple points across this continuum. For example, rapid quality control may be executed near an instrument on

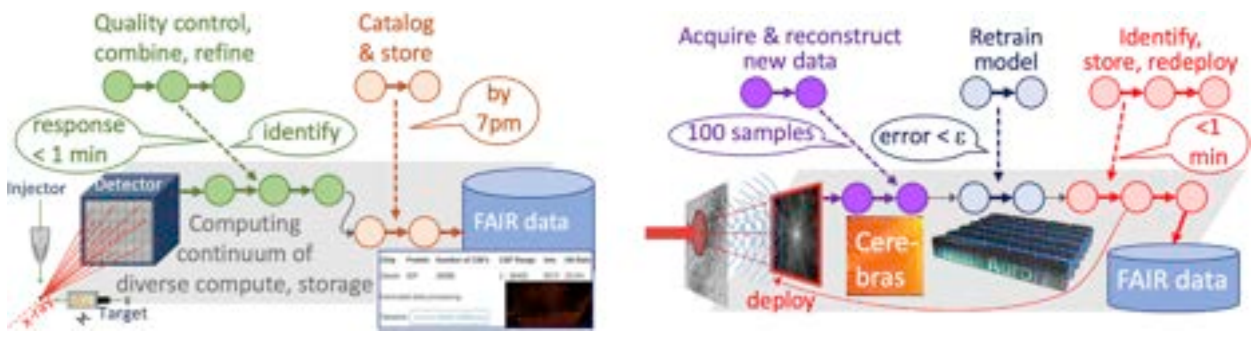


Figure 1. Two examples of instrumentation + computation applications, showing constituent flows
Left: serial synchrotron crystallography; right: high-energy diffraction microscopy. In each, a variety of computing systems (including, on the right, a Cerebras AI accelerator) are used to enable rapid collection and analysis of data from synchrotron light source experiments. In each subfigure, we show, as directed acyclic graphs linking distinct actions, both the distinct flows used to automate different functions (above) and their deployment in the context of the applications (below). The callouts indicate quality of service requirements.

a co-located device, machine-learning (ML) training on specialized AI hardware, and large-scale reconstruction on a data center cluster. The "best" location for a computation can be hard to determine and may change over time according to data location, resource availability, cost, and performance.

Example realizations of patterns

The two flows in Figure 1, to be described in more detail in application experiences, illustrate some of the elements just described. Serial synchrotron crystallography (SSX) experiments collect diffraction data from target crystals. Several flows combine to process batches of acquired images, identify "hits," refine crystal structures, and catalog results for later use. High-energy diffraction microscopy (HEDM) is used to characterize polycrystalline microstructures. This flow uses acquired data to train a neural network model for detecting peak positions in raw data. After training on a suitable AI accelerator, the flow transfers the trained model to the instrument for online use.

Implementing flows with the Globus platform

Having reviewed patterns for coupling experimental facilities with computation, we now examine how these patterns may be realized in practice, with the goal of providing actionable information that readers can apply to develop and execute their own flows.

We believe strongly that the widespread integration of scientific instruments into computational flows requires reusable flow specifications that can be easily adapted to different applications, instruments, and computational environments. Thus, our chosen approach to flow authoring and execution combines automation services for the specification and execution of flows with a research automation fabric to enable decoupling of abstract automation actions (e.g., move data, run program, publish records) from the specifics of individual data stores, computers, and catalogs—so that, for example, different compute and storage tasks can be directed to different resources (e.g., data center cluster, cloud, local accelerator) depending on needs and availability. In the following, we describe these two sets of capabilities in turn. For concreteness in presentation, we employ capabilities provided by the Globus platform,[48] which address both sets of needs.

The Globus research automation fabric

The Globus platform comprises a set of cloud-hosted services to which users can make various requests: for example, to transfer data from one storage system to another, run a computation on a computer, and load or search data in a catalog. In each case, the appropriate cloud service handles details such as authentication, authorization, monitoring of progress, and retries on failure that would otherwise hinder a scientist's work. We leverage the following Globus services:

- IAM services (Auth, Groups) for single sign-on and management of identities and credentials and delegation.
- Data services (Transfer, HTTPS, Share) for access to, and managed movement of, files.
- Metadata management (Search, Identifiers) for indexing and generating persistent references to data.
- Compute services (funcX, OAuthSSH) for invocation and management of computational tasks.
- Automation services (Flows, Triggers, Queues) for execution of flows and related activities.

The Globus Transfer[49] and funcX[50] services interact with local proxy agents deployed on storage systems and computers, respectively: Globus collections (implemented by Globus Connect software) for data actions, and funcX endpoints (implemented by funcX software) for compute actions. These agents are deployed persistently at many experimental and computational facilities and can also be deployed as needed by scientists. The Globus cloud services plus the proxy agents implement a universal compute and data fabric that encompasses any and all resources on which agents are deployed—in aggregate, tens of thousands of resources at thousands of institutions worldwide, ranging from cloud providers to clusters, supercomputers, and AI accelerators. Searchable registries support the discovery of agents that a user has permission to access.

All Globus platform services leverage the Globus Auth security fabric[51] for management of user identities and credentials, generation of OAuth 2 access tokens[52] for programmatic invocation of services, and generation of delegation tokens that allow

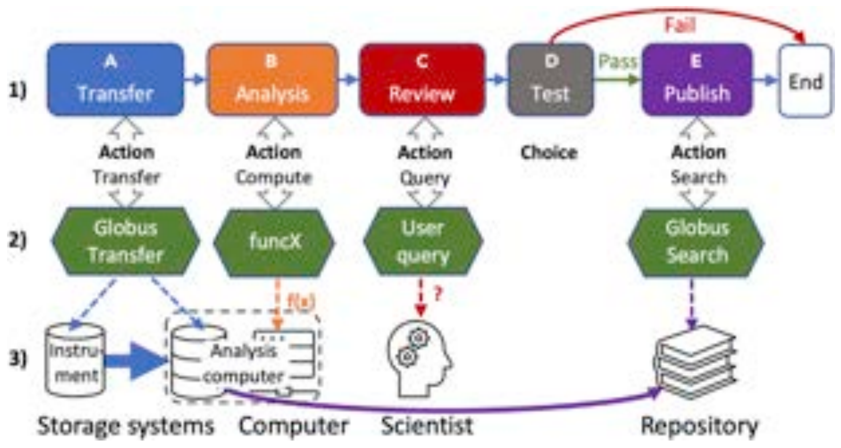


Figure 2. A simple example flow and its implementation

From top to bottom: (1) User perspective of a simple flow that, successively (shown left to right), (A) transfers data from an instrument to an analysis computer, (B) runs an analysis, (C) asks a user to review the analysis result, and, if (D) the user review is positive, (E) publishes the data to a repository. (2) The Globus platform services engaged by the transfer, compute, query, and search action providers. (3) The resources interacted with by those platform services: instrument storage system, co-located analysis storage system and storage computer, scientist, and data repository. Not shown are the Globus Auth service that handles identities and access tokens, and the Globus Flows service that coordinates flow execution.

services to act on a user's behalf. Crucially, data and computation remain at the edge: they never reach the cloud. Globus high-assurance service levels allow for management of protected (e.g., HIPAA) data.

Globus automation services

Globus automation services—Globus Flows, Triggers, and Queues[16]—build on the fabric provided by Globus platform services to allow scientists to specify and execute sequences of actions (or, sometimes, choices) called flows. A flow is specified as a JSON document—or, as described below, by using a Python toolkit, Gladier (for Globus Architecture for Data-Intensive Experimental Research). Flow execution may be invoked explicitly by the scientist or be triggered by an external event, such as generation of new data at an instrument. The Globus Flows service then manages flow execution. A web interface allows users to monitor the progress of a flow's execution and to detect and diagnose errors (see Figure S1).

Figure 2 shows an example flow and provides more details on how flows are implemented. Each type of action that may be invoked in a flow is handled by a persistent action provider service. Action providers can run programs (funcX,[50] OAuthSSH[53]), transfer files (Globus Transfer[49,54]), publish data to catalogs (Globus Search[55]), manage data permissions (Globus Share[56]), and generate persistent identifiers (Globus Identifiers[57]), among other tasks relevant to instrument data processing. In general, an action provider implements flow actions by requesting that the appropriate service (e.g., Globus Transfer, funcX) initiate the action and then polling periodically to see whether the action has completed. (As we discuss later, this polling can be a source of overhead.) All action provider services implement a consistent, asynchronous REST API,[58] facilitating the integration of new activities. Additional action providers may be deployed to support specific instruments, compute resources, or provide other customized needs by adhering to a well-defined interface.[16]

The implementation of Globus-operated action services, like those of other Globus platform services, leverages cloud services (e.g., Amazon Lambda, Step Functions, Simple Queue Service) for reliability and scalability. Cloud-based hosting enables delivery of research process automation capabilities to a wide user base without requiring users to download and install software. It also provides economies of scale, thereby reducing the costs associated with distributing software.

The Gladier toolkit

We have developed a Python toolkit, Gladier,[59] to assist in the authoring and management of flows for instrument science. This toolkit defines wrapper functions for registering funcX actions and flow definitions, invoking a new instance of a flow (a "run") with specified inputs, and monitoring a specified directory for file events. These functions allow for concise definitions of flows that integrate instrument and computation, as shown in Listing 1.

A Gladier user deploys client libraries on remote sources (e.g., on a computer co-located with an experiment) to detect events and invoke flows. A Gladier tool definition, implemented as a Python object, provides the information needed to populate a flow action. The Gladier toolkit provides implementations of common tools (e.g., transfer) as well as examples for experiment-specific tools (e.g., Stills processing with the Diffraction Integration for Advanced Light Sources [DIALS] package[60]); users may add other tools by implementing the Python class. To deploy and run a flow, users simply provide a list of tools to be used along with specific flow input arguments. Gladier uses this specification to register the necessary funcX functions and create and then register the flow definition.

We observe that flows for different experiments tend to follow similar patterns, independent of the experiment modality; the major area of customization concerns application-specific functions used to operate on data. Thus, scientists often can employ an existing flow unchanged, simply specifying different compute and data endpoint identifiers and storage paths; different processing function(s); and a different Globus Search catalog for publication. In other cases, they can adapt an existing flow by adding and deleting tools from the description and writing and deploying new funcX functions as required. Further, users can create, version, and share custom tools via GitHub, making them available for others to adopt within other flows.

The Gladier toolkit represents a relatively early attempt to provide a Pythonic interface to Globus Flows. Experiences thus far have been positive. Nevertheless, we imagine that future applications will motivate extensions—for example, to simplify specification of conditional execution and input schemas, both supported in Globus Flows but not handled well in the current

Listing 1. A simple SSX analysis flow, as defined with the Gladier toolkit. The flow comprises two tasks, one for the transfer from instrument to a compute resource, and one to run the DIALS Stills processing function on the transferred data. For brevity, we use *U1*, *U2*, and *U3* and *P1* and *P2* to represent UUIDs and paths, respectively.

```
from gladier import GladierBaseClient
@generate_flow_definition
class SSXFlow (GladierBaseClient):
  gladier_tools = [
  'gladier_tools.tools.Transfer',
  'gladier_ssx.tools.DialsStills'
]
flow_input = {
  'funcx_endpoint': U1,
  'transfer_source_endpoint_id': U2,
  'transfer_destination_endpoint_id': U3,
  'transfer_source_path': P1,
  'transfer_destination_path': P2,
}
ssx_flow_client = SSXFlow ()
run_id = ssx_flow_client.run ( flow_input )
```

toolkit. We expect to develop other interfaces (e.g., web) to support other communities.

Application experiences

We use five instrument + computation applications to illustrate how the patterns and technologies described in preceding sections can be realized and applied in practice. These applications link a number of scientific instruments and computing facilities, including Advanced Photon Source (APS) and Stanford Synchrotron Radiation Lightsource (SSRL) beamlines and the Argonne Leadership Computing Facility (ALCF).[61] Each example is implemented by using the Gladier toolkit to define, configure, and manage one or more flows. For each, we provide pointers in the supplemental information to the source code for both the full application and a simplified implementation that can be run on a personal computer, plus a sample dataset.

In each of the cases presented here, scientists had previously employed manual and ad hoc methods to implement similar, although typically simpler, behaviors: for example, by capturing data locally, transferring data via portable media to a cluster, and manually running analysis codes. After being introduced to Gladier tools, they implemented, with varying degrees of assistance from Gladier developers, the flows described in the following.

X-ray photon correlation spectroscopy (XPCS)

This experimental technique is used at synchrotron light sources to study materials dynamics at the mesoscale/nanoscale by identifying correlations in time series o ~ f area detector images.[62,63] Current detectors acquire megapixel frames at up to 2 kHz at 16-bit depth and 50 kHz at 2-bit depth (4 GB/s); next-generation detectors are expected to generate tens of GB/s or more.[64,65] Computing correlations at these rates requires powerful computing, both to process large quantities of data and to enable rapid response for experiment feedback.

We describe a flow developed to automate the collection, reduction, and publication of XPCS data at the APS 8-ID beamline. Each experiment can produce hundreds of thousands of images, with precise rate and image size controlled by the scientist. During image acquisition, the instrument's experiment management system typically creates a data file for every 20,000 images, with a size of ~2.4 GB; to enable use of the automation services described in this paper, it is configured to trigger a flow each time such a file is created.

The flow, illustrated in Figure 3, comprises 10 steps: (1) copy the experiment data file to a compute facility (transfer); (2) extract metadata, such as data acquisition parameters and processing instructions, from the experiment data file (compute); (3) copy these metadata to persistent storage (transfer); (4) load metadata into a Globus Search catalog, providing visibility into the data that are being processed and the software version and input arguments to be used during subsequent processing steps (search); (5) run the XPCS Boost correlation analysis function, a matrix-heavy operation that is best run on a GPU (compute); (6) run a plotting function to create correlation plots and compact images for display in the portal (compute); (7) extract metadata from correlation plots (compute); (8) aggregate the correlation plots, new metadata, execution logs, and compact images for publication (compute); (9) transfer the aggregated data and metadata to persistent storage (transfer); and (10) add the aggregated metadata and associated data references to the catalog entry created in step 4, thus allowing the scientist to verify quality and also making data available for future uses (search).

Before using this flow, it must be defined and registered with the Globus Flows service, and any tools and infrastructure used by the flow must be installed and configured if not already in place. We describe these steps in some detail in the supplemental information so as to illustrate the process by which a new flow is configured, deployed, and operated. A similar process is required for each of the other applications described in this section.

We note that while all computational steps (2, 5–9) can run on general-purpose CPUs, step 6, XPCS Boost analysis, benefits from the use of GPUs, and thus the flow, is typically configured to access a funcX endpoint associated with a GPU resource.

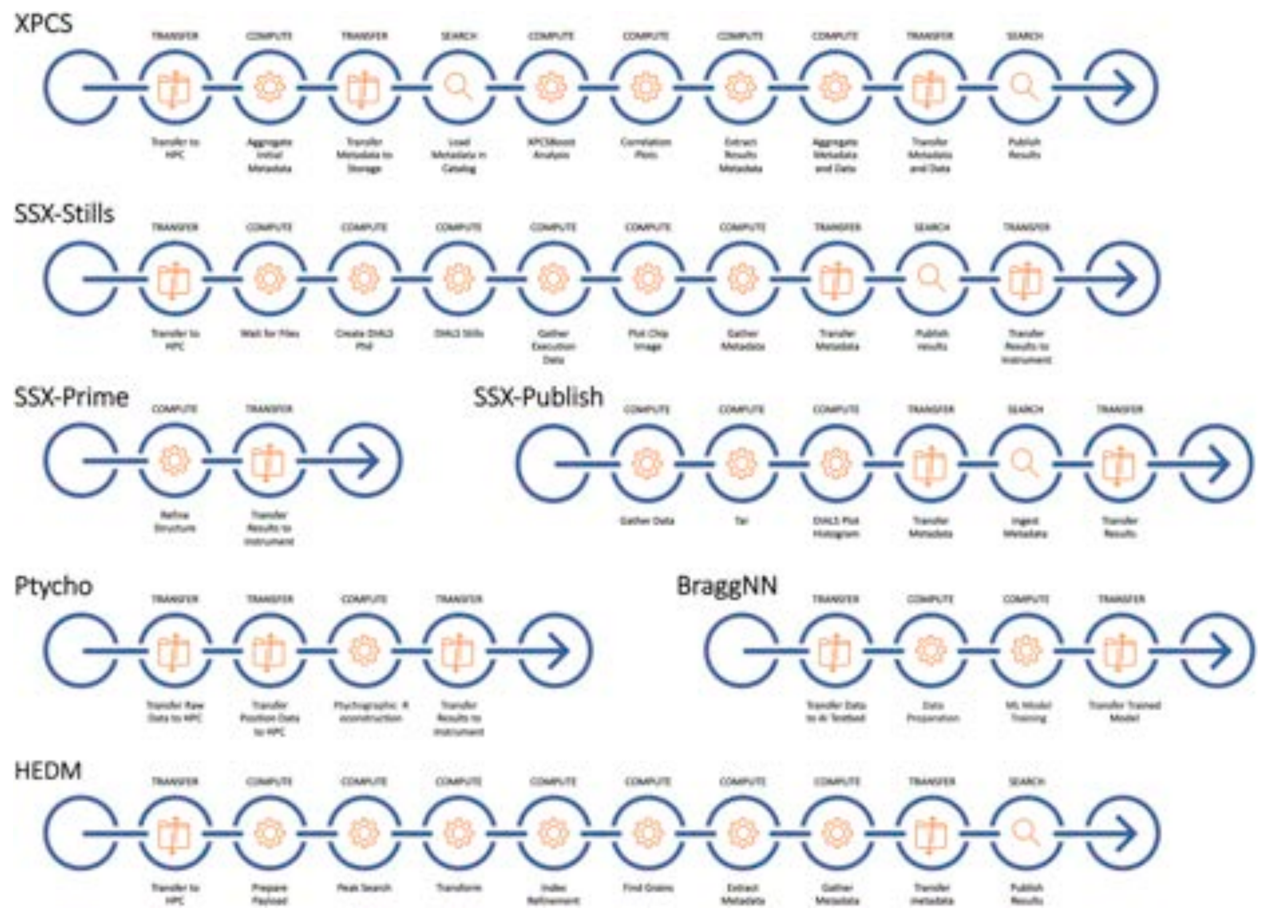


Figure 3. Depictions of the flows presented in the paper

An x-ray photon correlation spectroscropy processing flow, XPCS; three serial synchrotron crystallography flows, SSX-Stills, SSX-Prime, and SSX-Publish; a ptychography image reconstruction flow, Ptycho; a training flow for a neural network function approximator, BraggNN; and a high-energy diffraction microscopy far-field reconstruction flow, HEDM. Text above each circle names the action; text below describes its application in the flow.

Using GPUs, the flow can process a dataset and produce visualizations to the scientist in about 240 s (see Table 1) and in around 50 s with dedicated resources.

SSX

SSX is a technique in which a bright synchrotron beam and specialized apparatus are used to collect diffraction data from many crystals, at rates of tens of thousands of images per hour.[66] It can collect diffraction data from samples at room temperature and produce higher quality data than conventional crystallography due to reduced radiation damage.[67]

At APS Sector 19, a typical SSX experiment generates around 40,000 1,475 × 1,255 16-bit pixel images per sample, with tens of samples processed during a beamtime. While the detector is capable of operating at 100 Hz, for a data rate of 370 MB/s, the experiment is flux limited and is typically performed at roughly 10 Hz, or 37 MB/s. As images are produced, they are processed (in batches) with the DIALS package to identify crystal lattices, or hits, in each image. As hits are accumulated, they are processed with the post-refinement and merging (PRIME) package[68] to solve the crystal structure. DIALS and PRIME outputs are published to an SSX repository and cataloged for subsequent use.

These activities are implemented by three distinct flows. The first, SSX-Stills, transfers a batch of acquired images to a computing facility and uses the DIALS Stills package to perform quality analysis on each image and identify those that contain a good quality diffraction (a hit). It comprises 10 steps: (1) transfer image data from the beamline to a computing facility (transfer); (2) confirm necessary input files are present (compute); (3) create configuration files for analysis (compute); (4) perform DIALS Stills processing on each raw image (compute); (5) extract metadata from files regarding hits (compute); (6) generate visualizations showing the sample and hit location (compute); (7) gather metadata and visualizations for publication (compute); (8) transfer metadata and visualizations for publication (transfer); (9) ingest results, metadata, and visualizations to an SSX Globus Search catalog (search); and (10) transfer the results back to the beamline (transfer).

The SSX-Prime flow uses diffractions from SSX-Stills to solve the crystal structure. This flow is run first when at least 1,000 hits have been identified, and then again to refine the structure as additional hits become available. It (1) performs PRIME analysis

Table 1. For the instance of each flow with median runtime, the times taken by its constituent transfer, compute, and search action(s), in seconds, and the aggregate overhead, both in seconds (OH) and as a percentage of total runtime (%OH)

Experiment	Runtime	Transfer	Compute	Search	OH	%OH
BraggNN	259.5	64	162.1	0	33.4	12.9
HEDM	498.2	16	405.9	1	75.3	15.1
Ptycho	2,283.3	11	2,259.4	0	13.0	0.6
SSX-Publish	355.2	3	306.2	1	44.9	12.7
SSX-Prime	332.6	152	53.7	0	126.9	38.2
SSX-Stills	1,041.4	97	860.0	1	83.4	8.0
XPCS	240.0	12	177.9	2	48.1	20.0

to solve the structure (compute) and (2) copies the structure back to the beamline (transfer).

The SSX-Publish flow publishes results obtained to date, plus derived data such as histograms, to a repository and catalog. Its six steps are as follows: (1) gather results, metadata, and visualizations (compute); (2) create an archive file containing processed data (compute); (3) create histograms of the analysis (compute); (4) transfer metadata and results for publication (transfer); (5) publish results to the SSX repository and catalog (search); and (6) transfer results back to the beamline (transfer).

These three flows are initiated by a local agent deployed at the instrument that monitors the creation of files. In the experiments reported here, an SSX-Stills flow is triggered for each of the 512 images and an SSX-Publish flow for each of the 4,096 images; an SSX-Prime flow is triggered initially when at least 1,000 hits have been identified, and then again after each SSX-Stills flow completion. This flexibility allows each activity to proceed at an appropriate pace and permits new flows to be triggered given the result of previous flows, further advancing the automation of the scientific process.

The result is an indexed, searchable collection of processed images and associated statistics that is updated continuously while an experiment is running. Scientists use this catalog to determine whether sufficient data have been collected for a sample, a second sample is needed to produce suitable statistics, or a sample is not producing sufficient data to warrant continued processing.[69]

Ptychography

This coherent diffraction imaging technique can image samples with sub-20 nm resolutions.[70] A sample is scanned with overlapping beam positions while corresponding far-field diffraction patterns, 2D small-angle scattering patterns containing frequency information about the object, are collected with a pixelated photon counting detector. Current detectors routinely generate 1,030 × 514 12-bit pixel frames at 3 kHz for ~20 Gbps[71] and TBs per experiment. Next-generation detectors will have readout speeds of more than 100 kHz and increased pixel counts, resulting in multi-PB datasets.

Phase retrieval is applied to ptychography data to recover phase information in reciprocal space. Typical phase retrieval algorithms are iterative and hence computationally expensive. ML-based methods that perform phase retrieval in a non-iterative manner[72–74] can achieve speedups of tens[73] to thousands[74] of times, opening the door to real-time imaging and thus automated steering of experiments. However, phase retrieval is highly sensitive to material properties, and hence the ML model must be retrained for each new material.

The Ptycho flow performs 2D inversion and phase retrieval on diffraction patterns. It comprises three steps:[75] (1) transfer data from experimental facility to computing facility (transfer), (2) process each diffraction pattern to obtain a full image (compute), and (3) transfer intermediate results back to experimental facility (transfer). During a ptychography experiment, hundreds of instances of this flow can be initiated concurrently. Further, this flow can be extended with 3D reconstruction steps and science-specific AI/ML methods: for example, feature segmentation and event or phenomena detection to enable feedback loops for experimental steering.

HEDM

This non-destructive technique combines imaging and crystallography algorithms to characterize polycrystalline material microstructure in three dimensions (3D) and under various *in situ* thermo-mechanical conditions.[22,76] The technique uses a synchrotron beam to map grains in a polycrystalline aggregate by considering diffraction patterns as a function of rotation angle. It thus requires identification of diffraction "spots" for each grain. Far-field (~10 μm) HEDM, near-field (~1 μm) HEDM, and tomography may be combined when studying a material,[76] with, for example, far-field data used to guide near-field measurements.

We present two distinct HEDM applications that implement different approaches to HEDM data analysis. The first, HEDM, involves flows for collection, analysis, and storage of far-field and near-field data and for coordination of those activities. We show in Figure 3 the first of these flows, which involves eight steps: (1) transfer data from experimental facility to computing facility (transfer); (2) process each raw image using MIDAS[77] (compute); (3) extract metadata from files regarding hits (identified crystal diffractions) and generate visualizations showing the sample and hit locations (compute); (4) process each set of processed images (from step 2) to refine structure (compute); (5) gather metadata (compute); (6) transfer metadata to storage facility (transfer); (7) publish raw data, metadata, and visualizations (search); and (8) transfer the results back to the experimental facility (transfer). A single flow typically moves ~11.5 GB and consumes ~400 s of compute time in steps 2 and 4.

The MIDAS package used by the HEDM application determines peak positions and shapes by fitting the observed intensities in area detector data to a theoretical peak shape such as pseudo-Voigt. While the HEDM flow presented allows scientists to harness powerful computing for these computations, the higher data rates at new experimental facilities greatly increase overall computational costs.[29] A promising alternative, explored in our second HEDM application, BraggNN, is to train and deploy a neural network approximator to the conventional curve fitting function. The neural network training can be performed on a powerful data center computer (e.g., conventional cluster or AI accelerator), after which the trained network can be deployed on a lightweight "edge" device at the instrument for real-time diffraction peak analysis to power applications such as experiment steering and anomaly detection.

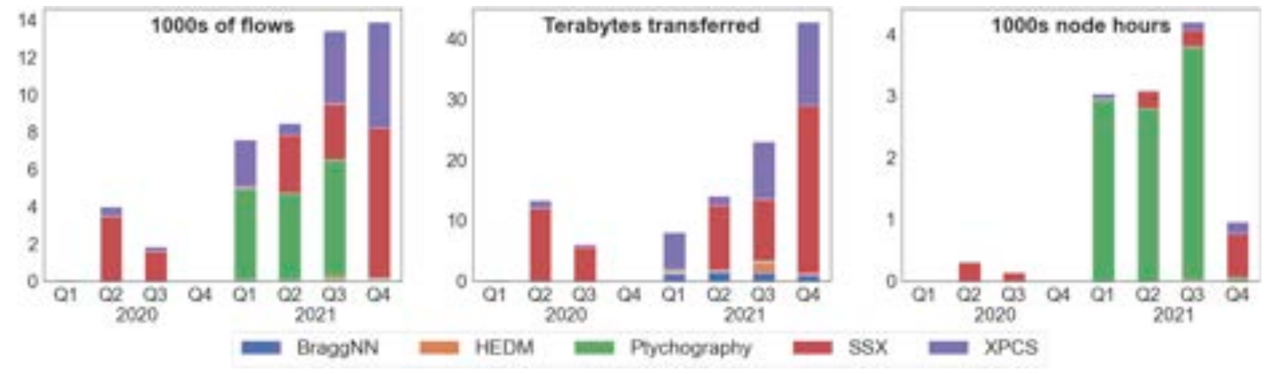


Figure 4. Resource usage over time by five experiments
Total flows, data transferred, and compute time used (on 64-core ALCF Theta nodes), per quarter, for the five experiments described in application experiences.

The BraggNN flow, as shown in Figure 3, explores the feasibility of this approach and, in particular, the relative costs of data transfer, network training, and network deployment. It comprises just four steps[28,78]: (1) copy data from beamline to computing facility (transfer); (2) prepare the data for training (compute); (3) train the BraggNN model (compute); and (4) copy the trained model back to the beamline (transfer). In the experiments described below, data are collected at SSRL and transferred to ALCF for training on AI accelerators such as the Cerebras wafer-scale engine.[79] The ease with which Gladier permits retargeting of compute tasks proved invaluable when selecting an appropriate platform for different neural network architectures.

Application usage

Scientists have employed the methods and tools described above at APS and ALCF since early 2020 at a cadence that has varied with instrument availability and research priorities but that is generally increasing. Usage across the five experiments described in this article, summarized in Figure 4, encompass 49,367 distinct flow runs that consumed over 11,700 node hours of compute and transferred roughly 108 TB. The variation in usage across experiments and over time is primarily due to the sporadic nature of experiments at large-scale facilities. There are periods of downtime in which few, or no, experiments are run. We see a general increase over time in the number of flows run and the amount of data transferred. The decrease in compute time in Q4 2021 is due to the fact that the compute-intensive ptychography experiment was not running during this period. Several experiments are deploying more ambitious and expensive computational methods now that the feasibility of on-demand computing has been established.

We explore in Figure 5 the ability for flows to keep pace with data acquisition rates. Specifically, we show a 12 h period in which XPCS flows are executed during an experiment session. During a preparatory period of roughly 4 h, the scientists run occasional bursts of flows to calibrate equipment and ensure that the analysis pipeline is operational. Here, we see up to 39 instances of the XPCS flow executing concurrently, each with the 11 steps shown in Figure 3. The subsequent 8 h of the experiment represents steady-state processing in which flows are executed as the result of data acquisition. We see here that approximately 10 flows execute concurrently throughout the experiment, showing that the flows meet the required data acquisition rate of one file per minute. The additional flows represent out-of-band reprocessing tasks executed by the scientists.

We compare the runtime of each flow in Figure 6. Here, we see mean and quartiles for the more than 2,600 flow runs. We see that the Ptycho flow has significantly longer execution times and also higher variance in execution time (25th to 75th percentile is approximately 2,000 s) than other flows. This variance is primarily due to unpredictable compute cluster queue delays, as these flows were run without dedicated reservations. Importantly, flows complete reliably despite such delays.

We show in Figure 7 a breakdown of action execution time for a single instance of each flow. We select the instance of that flow with median total runtime and show the time spent executing each action as measured by the respective action provider. We illustrate overhead as the difference between the time measured by the action provider to perform the task and the time recorded by the Globus Flows service to complete a step. Overheads include costs incurred as Globus Flows transitions between steps, invokes action providers to submit a task, and, most significantly, polls for action status (see next paragraph). Flow durations ranged from a mean of 31 s for XPCS to 3,527 s for Ptycho. All except SSX-Prime are compute bound. For SSX-Prime and some other flows, the overheads (see Table 1) reveal opportunities for optimization (e.g., by improved polling strategies), but none are so high as to hinder experiments.

Figure 8 drills down on the runtime and overhead of individual steps within the XPCS flow. The histograms in the top row are of runtimes for each of the flow's 11 steps, over 2,197 flow executions; those in the bottom row are the associated per-step overheads. The varied performance seen in the runtime graphs for transfer and compute actions is expected, as these actions involve functions that may run for minutes and transfers that move gigabytes and that are subject to compute cluster queue and Globus Transfer limits, respectively. The similar distributions seen in the runtime and overhead graphs for the same action are due to the exponential backoff polling interval (starting at 1 s) used by Globus Flows: the longer an action takes to execute, the less frequently Globus Flows polls the action to check completion. (The backoff maximum of 10 min is reflected in the maximum overhead of roughly 500 s.) The two search actions show more consistent performance (within 20 s), although still

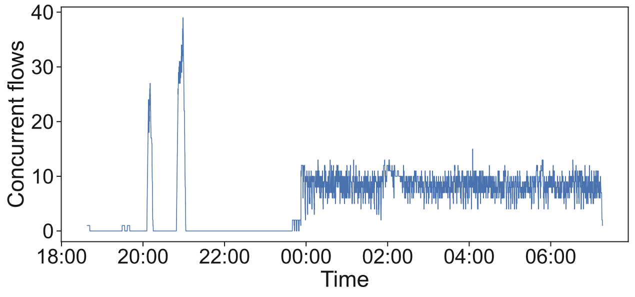


Figure 5. The number of concurrent XPCS flows over a roughly 12 h period, March 10–11, 2022
The initial peaks are burst tests before beginning the experiment; by 00:00, a constant stream of data from the beamline is processed.

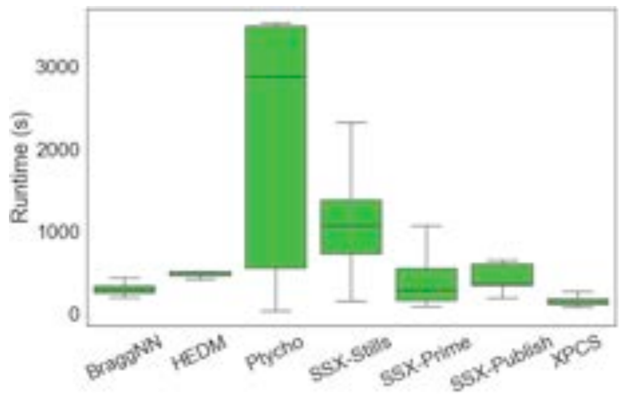


Figure 6. Distribution of runtimes for the seven flows discussed in the text
Box plots show upper and lower quartiles, with whiskers to 1.5× the interquartile range.

with outliers. Roundtrip times to cloud services are not a significant source of overhead for any action. These results show, again, overheads that are acceptable for these applications but with opportunities for optimization.

DISCUSSION

We discuss implications of the patterns and technologies described here for various stakeholders. We base this discussion on our experiences working with the five example applications described in application experiences, each of which use the patterns and technologies outlined in this article to meet their science needs.

Adopting patterns

The patterns presented in this article can be used to design and implement instrument-linking applications using technologies different than those presented here (i.e., they could be implemented using other components). The patterns illustrate common steps that are necessary for such use cases and outline requirements for related systems. Implementations of these patterns present concrete examples that can be reused and adapted to other use cases.

Adopting Globus and Gladier

The Gladier toolkit and Globus platform are publicly available and accessible to the research community. Thus, users can define new flows or adapt published flow templates that implement common patterns, including those described here. Our platform-based approach means that a user need only ensure that Globus and funcX endpoints are in place before running a flow in a new environment. At many scientific facilities, required endpoints are already deployed, in which case users need only modify a template to specify endpoints, data locations, and compute functions. In environments where endpoints are not already available, users must first deploy the endpoint software to make their resources accessible—a relatively straightforward task as Globus Transfer endpoint software is distributed in native Linux packages and for MacOS and Windows PCs, while funcX endpoint software can be installed via Python pip (Package Installer for Python). A happy consequence of these low deployment costs and our use of Python has been considerable diversity in our early adopter community. For example, the flows described in application experiences were authored by both computer scientists and domain scientists, with little support from our team.

Use of a cloud platform

Our use of Globus platform services for IAM, data, flow automation, and computation simplified the realization of the patterns described here. Because Globus operates on a public cloud with publicly accessible APIs and web interfaces, users can readily start, monitor, and manage flows irrespective of where they and where their flows are located. They also benefit from the heightened reliability that results from outsourcing the management of multi-step flows spanning distributed resources to a reliable cloud platform with replicated state. The cloud-hosted services architecture also makes it easy for users to compose flows in different ways to meet different needs, without the need to apply monolithic software stacks.

The Globus platform's use of web authentication and authorization standards (e.g., OAuth 2[52]) provides a rich IAM ecosystem for managing the security of complex flows. This approach allows users and resource owners to manage what actions are performed and by whom and also supports the complexities of real-world use cases. For example, Globus Auth allows for secure integration with external tools (e.g., facility data management systems) by using various OAuth 2 grant types (e.g., for public clients), group-based community accounts for shared computing access, and delegated authorizations for flows to securely invoke external services.

The ease with which the platform can be extended to edge resources by deploying data and compute agents (Globus collections and funcX endpoints, respectively) is important for use cases that require edge computing. These lightweight and easily installed agents offer crucial capabilities that allow execution of actions on remote and diverse resources. They may be operated by resource owners to support any authorized users or, alternatively, deployed by an individual user to process their own requests only.

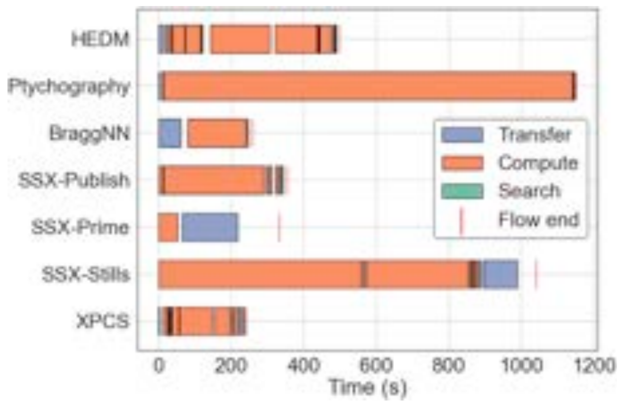


Figure 7. For the instance of each flow with median runtime, a timeline for its constituent actions

The empty spaces between steps correspond to flow orchestration overheads. Note that the Ptycho analysis times are scaled to 50% (from 2,261 to 1,130 s total) so as to better show details in the other flows.

While the Globus platform provides capabilities needed to implement a broad range of flows, it does not (and cannot) offer every capability desired by users. Thus, another advantage of the platform model is that we are able to prescribe a common asynchronous REST API and flexible OAuth-based IAM model such that others can implement and integrate external actions with the platform. This API and IAM model could be used to integrate capabilities provided by other cloud-hosted research platforms, such as Tapis[28] and CILogon/CoManage.[80] Integrating other platforms is dependent on the need for platforms to "trust" one another so that authorization decisions can be routed to different authorization servers. Adoption of common token formats (e.g., SciTokens[81]) would further enable consuming services and agents to validate assertions from different authorization domains.

A potential disadvantage of cloud-based platforms such as Globus is the need for continuous connectivity between research facility and cloud, which introduces a new failure mode and may not be permitted by cybersecurity policies. We see such concerns declining due to the high availability, reachability, and security of modern clouds but note that a possible compromise is to use local computers for initial data capture while leveraging the cloud platform for more advanced capabilities.

Implications for computing facilities

Rapidly advancing and evolving experimental apparatus and associated computational methods result in growing demands for computing and storage. The appropriate combination of custom silicon, edge computing, and data center computing likely will evolve over the next decade and beyond; however, it remains natural to turn to large computing facilities (e.g., data centers, clouds) for both capacity and hardware specialization (e.g., accelerators). Such facilities are natural rallying points for data storage and organization coupled with close access to compute resources. These needs are particularly important given the adoption of new computing modalities, such as AI and digital twins.[82,83]

The experiences reported here show the benefits of a platform that permits easy redirection of tasks to different destinations so that choices can be made based on user preferences and/or institutional policies. However, enabling such redirection relies on facilities exposing interfaces for remote access to data and computing, IAM infrastructure to enable seamless, yet secure, access to such resources, and methods for enabling access (e.g., to service accounts) without prior direct trust relationships.

Even simple mechanisms can drive innovation. For example, ScienceDMZs[84] have enabled unobstructed data flows to/from scientific computing facilities, deployment of user-managed and Globus-accessible storage has allowed scientists to rapidly collaborate using shared data, and support for container technologies has reduced barriers for porting applications between systems.[85] These mechanisms should all be universally adopted by computing facilities to enable instrument + computation flows.

Our work has highlighted other capabilities that could reduce barriers for linking instruments and advanced computing.[86] Flexible, on-demand access to computing capacity is needed to support bursty online workloads. The modest computing demands associated with our five experiments were satisfied at ALCF by a mix of backfill queue, standard queues, and reservations, but such capabilities may no longer suffice as demands increase. Some sites operate both specialized queues and dedicated and on-demand clusters,[87–89] but more flexible scheduling mechanisms are likely needed. In high-demand situations, the ability either to transition automatically (through standardized and exposed IAM infrastructure) to other computing facilities, including to the commercial cloud (funcX supports provisioning of cloud instances), without direct intervention from experimental scientists could allow the scientists to stay focused on real-time needs. New facility evaluation metrics are needed that encompass not only utilization but also responsiveness for real-time workloads.

Planning for future computing-enhanced experimental science suffers from inadequate knowledge of future demand and the cost-performance tradeoffs associated with meeting demand in different ways. It will be important to establish systematic tracking of resource demand and availability at both experimental and computing facilities. Also needed is a cohort of staff with expertise in both experimental science and computing to assist with the development, deployment, and executing of flows such as those described here.

Implications for experimental facilities

Effective coupling of experimental and computational facilities requires both modern computing infrastructure at experiments and high-quality internal and external network connections; many facilities still have deficiencies in these areas. Adoption of the ScienceDMZ architecture[84,90] is important so as to eliminate bottlenecks in network paths. Experimental facilities must support deployment of the Globus and funcX software needed to integrate with the cloud-based compute and data fabric described here. This is both a social and technical challenge. Administrators must allow for policies that permit deployment and provide for external connectivity, both to computing facilities

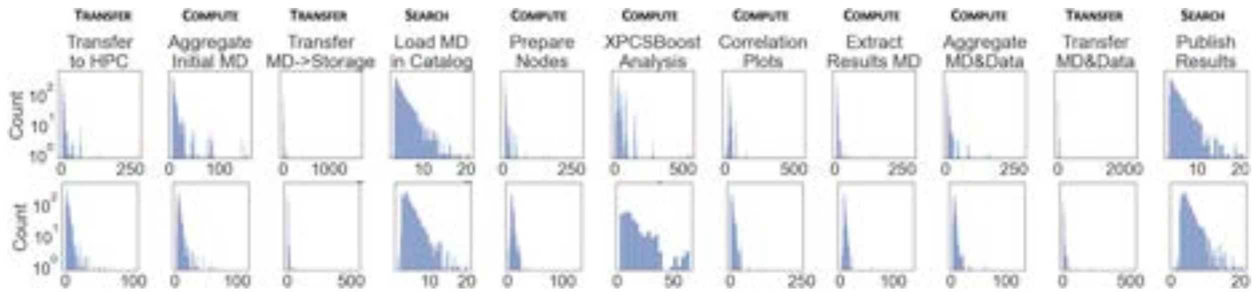


Figure 8. Run time distributions in seconds
Distributions of run time (first row) and overhead (second row), in seconds, for each of the 11 steps in the XPCS flow.
MD, metadata.

and to cloud-hosted platform services. Facilities must provision hardware near instruments so that agents can be deployed close to data sources. Work is also needed to integrate IAM ecosystems. Many facility users are locked within a single IAM domain. Adoption of federated IAM, such as that provided by Globus Auth and adopted by a growing number of scientific computing facilities, can integrate diverse IAM domains. By adopting standard mechanisms, facilities can make their identities accessible to modern cloud platforms.

There are opportunities for yet more sophisticated integration. For example, direct integration of the methods described here with the software tools employed by scientists reduces barriers for use by providing familiar interfaces to automation capabilities. Flows can also be used to control experiments, a practice that will require implementation of common APIs, perhaps aligning with the action provider API, for instruments and other devices.

Full automation (without human intervention) will require that experiments generate meaningful events that can be used to trigger flow executions.[91] In the applications reported here, flows are triggered by mechanisms that monitor co-located file systems to integrate with beamline software. Other integrations are possible, such as connecting with instrument control systems like EPICS,[92] Bluesky,[93] LabView,[94] and ROS[95] that allow for generation of events.

Implications for scientists

Higher data acquisition rates, larger datasets, and more complex processing flows mean that scientists must increasingly embrace automation to remain competitive. The outsourcing of automation tasks to cloud-hosted platforms, as described here, can simplify this transition by avoiding the need for larger local hardware and software deployments. However, scientists must be willing to trust external providers to handle mission-critical functionality. The growing reliance on cloud-hosted services in our daily lives, coupled with their extreme availability and reliability, helps to expedite this transition.

Adopting the patterns and methods proposed here requires that scientists decouple traditionally monolithic workflows into a series of discrete steps that may be executed separately. This approach can improve understandability and make it easier to substitute implementations for individual steps (e.g., to update an analysis routine) and to execute steps in more preferable locations (e.g., in terms of cost, availability, and performance).

We see increasing use of ML techniques for data analysis and for selecting experiment configurations, samples, and processes, with an increasing focus on completing the feedback loop to enable automated steering of experiments. These developments make it yet more important to automate data capture and cataloging so as to provide a clear provenance path when data are used for ML model training.

Facilitating FAIR science

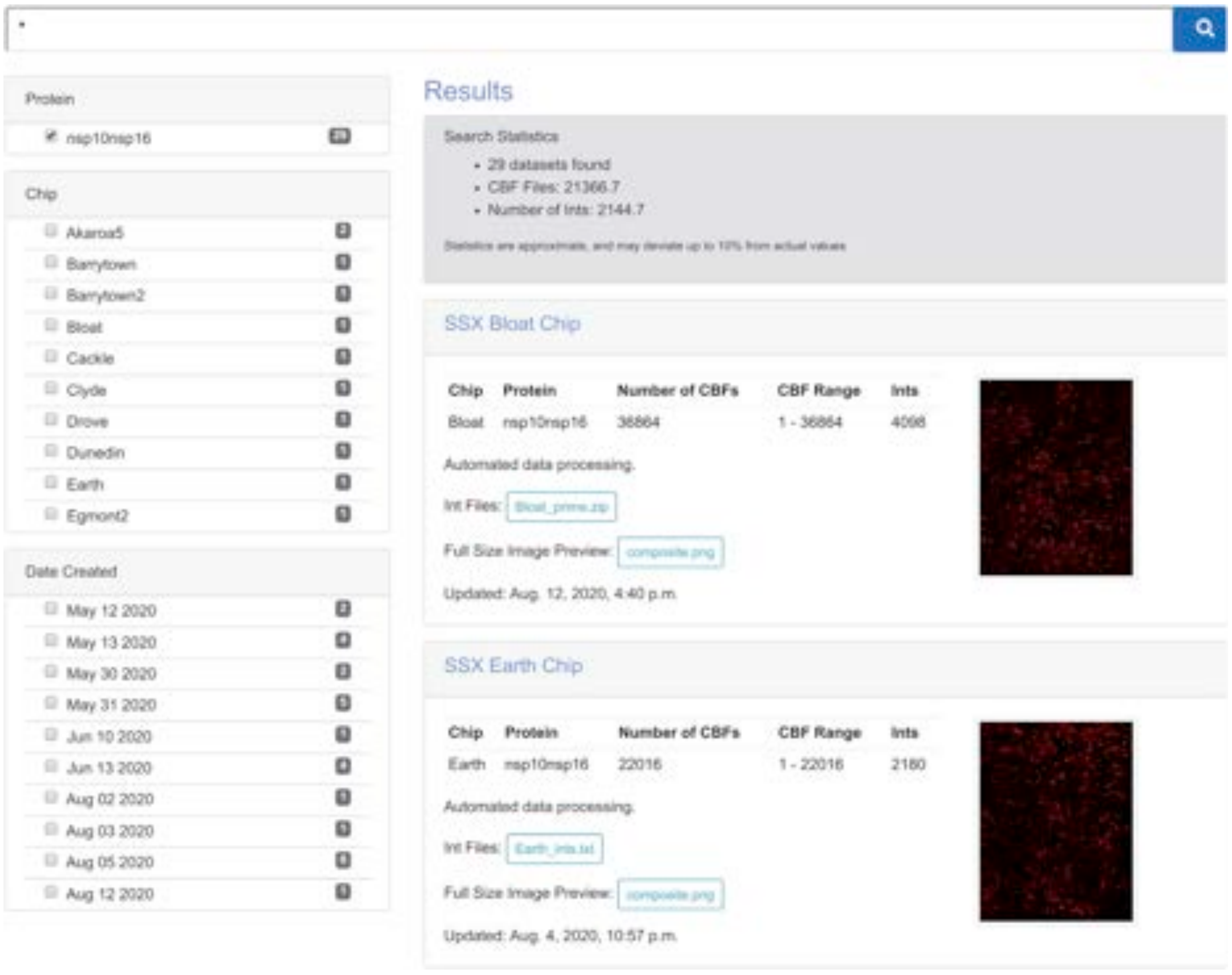
The methods described in this article can contribute to making experimental data FAIR[35,96] by making it easy to integrate data publication into data acquisition and analysis flows. In the SSX, HEDM, and XPCS examples presented here, data plus descriptive metadata (expressed in an extensible schema based on that of DataCite[97]) are published automatically to a Globus Search catalog, with an auto-generated interactive portal (e.g., see Figure 9). These catalogs have been used to index collections containing many terabytes in thousands of files. Trained models can also be published.[98]

Related work

Specialized data processing systems have been developed in fields such as high energy physics[40] and very long baseline interferometry.[99] At the Large Hadron Collider, ∼1 PB/s data streams are reduced by custom electronics and then processed on a distributed computing grid with hundreds of thousands of cores.[40] More routine linking of instruments with computers[100–103] predates the Internet.[104] Automation has involved both experiment-specific code[105,106] and orchestration and analysis solutions targeted at specific communities.[107–110] However, none enable specification and reuse of end-to-end flows as is done here.

Experimental facilities use control systems such as EPICS[92] to drive instruments and monitor experiments. Bluesky[93] provides Python interfaces for experiment control and data collection.[111] These systems can be combined with analysis tools and workflow systems to process data as they are captured. Streaming protocols can be used to expedite data movement.[112,113]

The Globus data fabric on which we build here is widely deployed in the US and other countries.[56] Other data sharing approaches, varying in scope, maturity, and adoption, include



Protein

- [✓] nsp10nsp16

Chip

- Akaroa5
- Barrytown
- Barrytown2
- Bloat
- Cackle
- Clyde
- Drove
- Dunedin
- Earth
- Egmont2

Date Created

- May 12 2020
- May 13 2020
- May 30 2020
- May 31 2020
- Jun 10 2020
- Jun 13 2020
- Aug 02 2020
- Aug 03 2020
- Aug 05 2020
- Aug 12 2020

Results

Search Statistics

- 29 datasets found
- CBF Files: 21366.7
- Number of Ints: 2144.7

Statistics are approximate, and may deviate up to 10% from actual values

SSX Bloat Chip

Chip	Protein	Number of CBFs	CBF Range	Ints
Bloat	nsp10nsp16	36864	1 - 36864	4098

Automated data processing.

Int Files: Bloat_prime.zip

Full Size Image Preview: composite.png

Updated: Aug. 12, 2020, 4:40 p.m.

SSX Earth Chip

Chip	Protein	Number of CBFs	CBF Range	Ints
Earth	nsp10nsp16	22016	1 - 22016	2180

Automated data processing.

Int Files: Earth_ints.txt

Full Size Image Preview: composite.png

Updated: Aug. 4, 2020, 10:57 p.m.

Figure 9. SSX data analysis portal
Facets on the left allow for selection of different proteins (*nsp10nsp16* is selected here), chips, and creation dates. Search results, shown on the right, provide researchers with a quick summary of the experiment and visual representation of the analysis results.

logistical networking,[114] Rucio[115] and StashCache[116] in high energy physics, ELIXIR[117] for the life sciences, PANdata[118,119] and EXPANDS[120] for photon and neutron science, iRODS,[121] and the European Open Science Cloud.[122]

The term scientific workflow encompasses many technologies.[123–125] Scientific workflow systems are commonly used to orchestrate many-task computational campaigns[126–128] that may execute local programs or submit jobs to data center computers. Research on workflow scheduling, execution, and related problems has enabled impressive scale and performance within individual systems or across multiple computers under coordinated control.[129,130] In contrast, the patterns that are our focus engage many concerns besides orchestration of compute jobs.[131] We require methods for linking diverse activities and resource types, from computations on computers to experiments on scientific instruments; integrating different resource types; bridging authentication domains; managing flows that may run for days or even weeks; and organizing and arbitrating among collections of flows. These concerns motivate our decision to build on the cloud-hosted Globus platform,[48] which provides for robust orchestration of diverse activities managed by purpose-specific agents that are already widely deployed. (The Taverna Web services orchestration platform, while not cloud hosted, had similarities.[132]) The extensibility of the Globus platform allows for the introduction of new non-compute elements into flows and thus into the patterns realized by these flows.

Bridging instruments and distributed computation requires capabilities for reliable and secure remote task submission. This challenge motivated Grid computing[133,134] and the superfacility concept.[135] Facilities have developed specialized interfaces for remote job submission[136,137] and for managing workloads on and across systems.[129,138] Remote execution has been integrated with Jupyter notebooks.[139–141] The ability to compute anywhere enables users to leverage specialized computing resources designed for low-cost, distributed, and edge computing.[142] AI systems deployed at experimental facilities support rapid data filtering at the edge.[28]

Domain-specific data repositories can play a pivotal role in fostering collaboration.[143–145] Science gateways[146,147] address data and compute challenges by abstracting underlying resources and providing intuitive analysis interfaces.

The value of federated identity and single sign on as means of streamlining access to scientific resources is broadly recognized,[148–151] although not yet universally adopted. Globus Auth complements such initiatives by using OAuth tokens[52] to delegate to third parties (e.g., a funcX server) the right to perform certain tasks, such as transferring data and running functions, on a user's behalf. Delegation methods have been developed previously.[152–154]

Summary

Maximizing the value obtained from new instruments requires tight coupling with heterogeneous and large-scale computing facilities and new online computing methods to automate data collection, processing, and dissemination. We have reported on our experiences working with five groups of instrument scientists, first to understand their current and future computing challenges and second to automate various of their research flows. We described an automation approach that leverages Globus platform services to enable construction of flows by composing modular components that execute programs, transfer files, publish data to catalogs, manage data permissions, and generate persistent identifiers, among other tasks. Importantly, given dynamic resource availability, our approach achieves a separation of concerns between what actions are applied in each flow and where those actions are performed. We also described Gladier, a Python toolkit that abstracts registration of funcX functions, flow authoring, and flow execution with specific input arguments and simplifies the coupling of such flows to experiments.

The five experiments discussed here vary significantly in their data rates, flow and action runtimes, use of heterogeneous resources, and geographically distributed execution. We provide quantitative evaluations of those differences and demonstrate that our methods can, in each case, support the robust, scalable, and performant execution required for production use, with overheads that are acceptable even for complex and long-running flows.

This work represents a first step towards identifying, and capturing in reusable forms, a broad collection of patterns for processing data from scientific instruments—patterns that range from online data processing to ML training and data cataloging. We believe that understanding these patterns and the methods and resources required to support their execution will have important implications for a range of stakeholders, from individual scientists to compute facilities, experimental facilities, and cloud-based research platforms.

EXPERIMENTAL PROCEDURES

Resource availability

Lead contact

The lead contact is Ian Foster (foster@anl.gov).

Materials availability

This study did not generate new unique materials.

Data and code availability

In an effort to make this work accessible, reproducible, and reusable, we have published, via Zenodo, archival snapshots of the following software referenced in this article, at the time of publication: the Gladier toolkit[155] and associated tools;[156] the five simplified Gladier applications described in the supplemental information ("Gladier Examples");[157] and full versions of the five Gladier applications described in the article, Gladier XPCS,[158] Gladier HEDM,[159] Gladier Kanzus (SSX),[160] Gladier BraggNN,[161] and Gladier Ptychography.[162] Each of the provided Zenodo DOIs also provides a GitHub repository URL that may contain more recent versions of the code. We also provide access to example datasets[163] for each simplified application through the Materials Data Facility.[164,165]

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.patter.2022.100606.

ACKNOWLEDGMENTS

This work was supported in part by NSF grants OAC-1835890 and OAC-2004894; award 70NANB14H012 from the US Department of Commerce, National Institute of Standards and Technology, as part of the Center for Hierarchical Material Design (CHiMaD); and by the US Department of Energy under contract DE-AC02-06CH11357, including by the Office of Advanced Scientific Computing Research's Braid project. We are grateful to staff at the APS, ALCF, University of Chicago Globus group, and SSRL for assistance with this work.

AUTHOR CONTRIBUTIONS

Software, R.V., R.C., N.D.S., and J.P.; investigation, R.V., R.C., N.D.S., T.B., A.L., Z.L., and S.N.; writing – review & editing, R.V., R.C., B.B., J.P., M.E.P., K.C., and I.T.F.; writing – original draft, K.C. and I.T.F.; conceptualization, R.C., B.B., M.E.P., N.S., K.C., and I.T.F.; project administration, B.B., N.S., and I.T.F.; methodology, B.B. and I.T.F.

DECLARATION OF INTERESTS

The authors declare no competing interests.

Received: June 16, 2022
Revised: August 7, 2022
Accepted: September 14, 2022
Published: October 14, 2022

REFERENCES

1. Davy, H. (1812). Elements of Chemical Philosophy, Part I, Volume 1 (Bradford and Inskeep).
2. White, A., Goldberg, K., Kevan, S., Leitner, D., Robin, D., Steier, C., and Yarris, L. (2019). A new light for berkeley lab–the advanced light source upgrade. Synchrotron Radiat. News *32*, 32–36. https://doi.org/10.1080/08940886.2019.1559608.
3. APS Upgrade. Visited May 1, 2022. https://www.aps.anl.gov/APS-Upgrade.
4. Daukantas, P. (2021). Synchrotron light sources for the 21st century. Opt. Photonics News *32*, 32–39.
5. Chenevier, D., and Joly, A. (2018). ESRF: inside the extremely Brilliant source upgrade. Synchrotron Radiat. News *31*, 32–35. https://doi.org/10.1080/08940886.2018.1409562.
6. Bostedt, C., Boutet, S., Fritz, D.M., Huang, Z., Lee, H.J., Lemke, H.T., Robert, A., Schlotter, W.F., Turner, J.J., and Williams, G.J. (2016). Linac coherent light source: the first five years. Rev. Mod. Phys. *88*, 015007. https://doi.org/10.1103/RevModPhys.88.015007.
7. Eberle, A.L., and Zeidler, D. (2018). Multi-beam scanning electron microscopy for high-throughput imaging in connectomics research. Front. Neuroanat. *12*, 112. https://doi.org/10.3389/fnana.2018.00112.

8. Bai, X.C., McMullan, G., and Scheres, S.H.W. (2015). How cryo-EM is revolutionizing structural biology. Trends Biochem. Sci. *40*, 49–57. https://doi.org/10.1016/j.tibs.2014.10.005.
9. Andreoni, I., and Cooke, J. (2017). The deeper wider faster programme: Chasing the fastest bursts in the universe. Proc. Int. Astron. Union *14*, 135–138. https://doi.org/10.1017/S1743921318002399.
10. Catlett, C., Beckman, P., Ferrier, N., Papka, M.E., Sankaran, R., Solin, J., Taylor, V., Pancoast, D., and Reed, D. (2022). Hands-on computer science: the array of things experimental urban instrument. Comput. Sci. Eng. *24*, 57–63. https://doi.org/10.1109/MCSE.2021.3139405.
11. Flores-Leonar, M.M., Mejía-Mendoza, L.M., Aguilar-Granda, A., Sanchez-Lengeling, B., Tribukait, H., Amador-Bedolla, C., and Aspuru-Guzik, A. (2020). Materials acceleration platforms: on the way to autonomous experimentation. Curr. Opin. Green Sustain. Chem. *25*, 100370. https://doi.org/10.1016/j.cogsc.2020.100370.
12. Steiner, S., Wolf, J., Glatzel, S., Andreou, A., Granda, J.M., Keenan, G., Hinkley, T., Aragon-Camarasa, G., Kitson, P.J., Angelone, D., and Cronin, L. (2019). Organic synthesis in a modular robotic system driven by a chemical programming language. Science *363*, eaav2211. https://doi.org/10.1126/science.aav2211.
13. Burger, B., Maffettone, P.M., Gusev, V.V., Aitchison, C.M., Bai, Y., Wang, X., Li, X., Alston, B.M., Li, B., Clowes, R., et al. (2020). A mobile robotic chemist. Nature *583*, 237–241. https://doi.org/10.1038/s41586-020-2442-2.
14. Wang, C., Steiner, U., and Sepe, A. (2018). Synchrotron big data science. Small *14*, 1802291. https://doi.org/10.1002/smll.201802291.
15. Rao, R. (2020). Synchrotrons face a data deluge. Phys. Today. https://doi.org/10.1063/PT.6.2.20200925a.
16. Chard, R., Pruyne, J., McKee, K., Bryan, J., Raumann, B., Ananthakrishnan, R., Chard, K., and Foster, I. (2022). Research Process Automation across the Space-Time Continuum. https://doi.org/10.48550/arXiv.2208.09513.
17. Globus for Scientific Instruments. https://www.globus.org/instruments.
18. Alexander, C. (1977). A Pattern Language: Towns, Buildings, Construction (Oxford University Press).
19. Gamma, E., Helm, R., Johnson, R.E., and Vlissides, J. (1995). Design Patterns: Elements of Reusable Object-Oriented Software (Addison-Wesley).
20. Cappello, F., Di, S., Li, S., Liang, X., Gok, A.M., Tao, D., Yoon, C.H., Wu, X.C., Alexeev, Y., and Chong, F.T. (2019). Use cases of lossy compression for floating-point data in scientific data sets. Int. J. High Perform. Comput. Appl. *33*, 1201–1220. https://doi.org/10.1177/1094342019853336.
21. Vohl, D., Pritchard, T., Andreoni, I., Cooke, J., and Meade, B. (2017). Enabling Near Real-Time Remote Search for Fast Transient Events with Lossy Data Compression*34* (Publications of the Astronomical Society of Australia). https://doi.org/10.1017/pasa.2017.34.
22. Pokharel, R. (2018). Overview of high-energy x-ray diffraction microscopy (HEDM) for mesoscale material characterization in three-dimensions. In Materials Discovery and Design (Springer International Publishing), pp. 167–201. https://doi.org/10.1007/978-3-319-99465-9_7.
23. Liu, Z., Sharma, H., Park, J.S., Kenesei, P., Miceli, A., Almer, J., Kettimuthu, R., and Foster, I. (2022). BraggNN: fast X-ray Bragg peak analysis using deep learning. IUCrJ *9*, 104–113. https://doi.org/10.1107/S2052252521011258.
24. Clackdoyle, R., and Defrise, M. (2010). Tomographic reconstruction in the 21st century. IEEE Signal Process. Mag. *27*, 60–80. https://doi.org/10.1109/MSP.2010.936743.
25. Nashed, Y.S.G., Vine, D.J., Peterka, T., Deng, J., Ross, R., and Jacobsen, C. (2014). Parallel ptychographic reconstruction. Opt Express *22*, 32082–32097. https://doi.org/10.1364/OE.22.032082.
26. Pelt, D., Batenburg, K., and Sethian, J. (2018). Improving tomographic reconstruction from limited data using mixed-scale dense convolutional neural networks. J. Imaging *4*, 128. https://doi.org/10.3390/jimaging4110128.
27. Wasmer, K., Le-Quang, T., Meylan, B., Vakili-Farahani, F., Olbinado, M., Rack, A., and Shevchik, S. (2018). Laser processing quality monitoring by combining acoustic emission and machine learning: a high-speed X-ray imaging approach. Procedia CIRP *74*, 654–658. https://doi.org/10.1016/j.procir.2018.08.054.
28. Liu, Z., Ali, A., Kenesei, P., Miceli, A., Sharma, H., Schwarz, N., Trujillo, D., Yoo, H., Coffee, R., Layad, N., et al. (2021a). Bridging data center AI systems with edge computing for actionable information retrieval. In 3rd IEEE/ACM Annual Workshop on Extreme-scale Experiment-in-the-Loop Computing (IEEE), pp. 15–23. https://doi.org/10.1109/XLOOP54565.2021.00008.
29. Li, J., Huang, X., Pianetta, P., and Liu, Y. (2021). Machine-and-data intelligence for synchrotron science. Nat. Rev. Phys. *3*, 766–768. https://doi.org/10.1038/s42254-021-00397-0.
30. Konstantinova, T., Maffettone, P.M., Ravel, B., Campbell, S.I., Barbour, A.M., and Olds, D. (2022). Machine learning enabling high-throughput and remote operations at large-scale user facilities. Digital Discovery *1*, 413–426. https://doi.org/10.1039/D2DD00014H.
31. Kusne, A.G., Yu, H., Wu, C., Zhang, H., Hattrick-Simpers, J., DeCost, B., Sarker, S., Oses, C., Toher, C., Curtarolo, S., et al. (2020). On-the-fly closed-loop materials discovery via Bayesian active learning. Nat. Commun. *11*, 5966–6011.
32. Noack, M.M., Zwart, P.H., Ushizima, D.M., Fukuto, M., Yager, K.G., Elbert, K.C., Murray, C.B., Stein, A., Doerk, G.S., Tsai, E.H.R., et al. (2021). Gaussian processes for autonomous data acquisition at large-scale synchrotron and neutron facilities. Nat. Rev. Phys. *3*, 685–697. https://doi.org/10.1038/s42254-021-00345-y.
33. Zhang, Y., Xie, R., and Zhang, H. (2022). Autonomous Atomic Hamiltonian Construction and Active Sampling of X-Ray Absorption Spectroscopy by Adversarial Bayesian Optimization. https://doi.org/10.48550/arxiv.2203.07892.
34. Maffettone, P.M., Lynch, J.K., Caswell, T.A., Cook, C.E., Campbell, S.I., and Olds, D. (2021). Gaming the beamlines—employing reinforcement learning to maximize scientific outcomes at large-scale user facilities. Mach. Learn, Sci. Technol. *2*, 025025. https://doi.org/10.1088/2632-2153/abc9fc.
35. Wilkinson, M.D., Dumontier, M., Aalbersberg, I.J.J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.W., da Silva Santos, L.B., Bourne, P.E., et al. (2016). The FAIR guiding principles for scientific data management and stewardship. Sci. Data *3*, 160018. https://doi.org/10.1038/sdata.2016.18.
36. Hidayetoglu, M., Bicer, T., de Gonzalo, S.G., Ren, B., Gursoy, D., Kettimuthu, R., Foster, I.T., and Hwu, W.M.W. (2022). MemXCT: design, optimization, scaling, and reproducibility of x-ray tomography imaging. IEEE Trans. Parallel Distrib. Syst. *33*, 2014–2031. https://doi.org/10.1109/TPDS.2021.3128032.
37. McClure, J.E., Yin, J., Armstrong, R.T., Maheshwari, K.C., Wilkinson, S., Vlcek, L., Da Wang, Y., Berrill, M.A., and Rivers, M. (2020). Toward real-time analysis of synchrotron micro-tomography data: accelerating experimental workflows with AI and HPC. In Smoky Mountains Computational Sciences and Engineering Conference (Springer), pp. 226–239. https://doi.org/10.1007/978-3-030-63393-6_15.
38. Chard, R., Madduri, R., Karonis, N.T., Chard, K., Duffin, K.L., Ordoñez, C.E., Uram, T.D., Fleischauer, J., Foster, I.T., Papka, M.E., and Winans, J. (2018a). Scalable pCT image reconstruction delivered as a cloud service. IEEE Trans. Cloud Comput. *6*, 182–195. https://doi.org/10.1109/TCC.2015.2457423.
39. Wang, S., and Casado, M. (2021). The Cost of Cloud, a Trillion Dollar Paradox. https://a16z.com/2021/05/27/cost-of-cloud-paradox-market-cap-cloud-lifecycle-scale-growth-repatriation-optimization.
40. Bird, I. (2011). Computing for the large Hadron Collider. Annu. Rev. Nucl. Part Sci. *61*, 99–118. https://doi.org/10.1146/annurev-nucl-102010-130059.

41. Hammer, M., Yoshii, K., and Miceli, A. (2021). Strategies for on-chip digital data compression for x-ray pixel detectors. J. Instrum. *16*, P01025. https://doi.org/10.1088/1748-0221/16/01/p01025.

42. Abeykoon, V., Liu, Z., Kettimuthu, R., Fox, G., and Foster, I. (2019). Scientific image restoration anywhere. In 1st IEEE/ACM Annual Workshop on Large-scale Experiment-in-the-Loop Computing (IEEE), pp. 8–13. https://doi.org/10.1016/j.eng.2020.01.007.

43. Chen, Y., Xie, Y., Song, L., Chen, F., and Tang, T. (2020). A survey of accelerator architectures for deep neural networks. Engineering *6*, 264–274. https://doi.org/10.1016/j.eng.2020.01.007.

44. Deiana, A.M., Tran, N., Agar, J., Blott, M., Di Guglielmo, G., Duarte, J., Harris, P., Hauck, S., Liu, M., Neubauer, M.S., et al. (2022). Applications and techniques for fast machine learning in science. Front. Big Data *5*, 787421. https://doi.org/10.3389/fdata.2022.787421.

45. Beckman, P., Dongarra, J., Ferrier, N., Fox, G., Moore, T., Reed, D., and Beck, M. (2020). Harnessing the computing continuum for programming our world. In Fog Computing: Theory and Practice (Wiley Online Library), pp. 215–230. https://doi.org/10.1002/9781119551713.ch7.

46. Balouek-Thomert, D., Renart, E.G., Zamani, A.R., Simonet, A., and Parashar, M. (2019). Towards a computing continuum: enabling edge-to-cloud integration for data-driven workflows. Int. J. High Perform. Comput. Appl. *33*, 1159–1174. https://doi.org/10.1177/1094342019877383.

47. Kumar, R., Baughman, M., Chard, R., Li, Z., Babuji, Y., Foster, I., and Chard, K. (2021). Coding the computing continuum: Fluid function execution in heterogeneous computing environments. In IEEE International Parallel and Distributed Processing Symposium Workshops (IEEE), pp. 66–75. https://doi.org/10.1109/IPDPSW52791.2021.00018.

48. Ananthakrishnan, R., Chard, K., Foster, I., and Tuecke, S. (2015). Globus platform-as-a-service for collaborative science applications. Concurr. Comput. *27*, 290–305. https://doi.org/10.1002/cpe.3262.

49. Allen, B., Bresnahan, J., Childers, L., Foster, I., Kandaswamy, G., Kettimuthu, R., Kordas, J., Link, M., Martin, S., Pickett, K., and Tuecke, S. (2012). Software as a service for data scientists. Commun. ACM *55*, 81–88. https://doi.org/10.1145/2076450.2076468.

50. Chard, R., Babuji, Y., Li, Z., Skluzacek, T., Woodard, A., Blaiszik, B., Foster, I., and Chard, K. (2020). FuncX: a federated function serving fabric for science. In 29th International Symposium on High-Performance Parallel and Distributed Computing (ACM), pp. 65–76. ISBN 9781450370523. https://doi.org/10.1145/3369583.3392683.

51. Tuecke, S., Ananthakrishnan, R., Chard, K., Lidman, M., McCollam, B., Rosen, S., and Foster, I. (2016). Globus Auth: a research identity and access management platform. In IEEE 12th International Conference on e-Science, pp. 203–212. https://doi.org/10.1109/eScience.2016.7870901.

52. Hardt, D. (2012). OAuth 2.0 Authorization Framework Specification. http://tools.ietf.org/html/rfc6749.

53. Alt, J., Ananthakrishnan, R., Chard, K., Chard, R., Foster, I., Liming, L., and Tuecke, S. (2020). OAuth SSH with globus Auth. In Practice and Experience in Advanced Research Computing (ACM), pp. 34–40. ISBN 9781450366892. https://doi.org/10.1145/3311790.3396658.

54. Liu, Z., Kettimuthu, R., Chung, J., Ananthakrishnan, R., Link, M., and Foster, I. (2021b). Design and evaluation of a simple data interface for efficient data transfer across diverse storage. ACM Trans. Model. Perform. Eval. Comput. Syst. *6*, 1–25. https://doi.org/10.1145/3452007.

55. Ananthakrishnan, R., Blaiszik, B., Chard, K., Chard, R., McCollam, B., Pruyne, J., Rosen, S., Tuecke, S., and Foster, I. (2018). Globus platform services for data publication. In Practice and Experience on Advanced Research Computing (ACM), pp. 1–7. ISBN 9781450364461. https://doi.org/10.1145/3219104.3219127.

56. Chard, K., Tuecke, S., and Foster, I. (2014). Efficient and secure transfer, synchronization, and sharing of big data. IEEE Cloud Comput. *1*, 46–55. https://doi.org/10.1109/MCC.2014.52.

57. Ananthakrishnan, R., Chard, K., D'Arcy, M., Foster, I., Kesselman, C., McCollam, B., Pruyne, J., Rocca-Serra, P., Schuler, R., and Wagner, R. (2020). An open ecosystem for pervasive use of persistent identifiers. In Practice and Experience in Advanced Research Computing (ACM), pp. 99–105. ISBN 9781450366892. https://doi.org/10.1145/3311790.3396660.

58. Fielding, R.T. (2000). Architectural Styles and the Design of Network-Based Software Architectures (Irvine: Ph.D. thesis. University of California).

59. Gladier team. Gladier Documentation. https://gladier.readthedocs.io.

60. Winter, G., Waterman, D.G., Parkhurst, J.M., Brewster, A.S., Gildea, R.J., Gerstel, M., Fuentes-Montero, L., Vollmar, M., Michels-Clark, T., Young, I.D., et al. (2018). DIALS: implementation and evaluation of a new integration package. Acta Crystallogr. D Struct. Biol. *74*, 85–97. https://doi.org/10.1107/S2059798317017235.

61. Riley, K., Papka, M.E., Collins, J., Heinonen, N., Cerny, B., Kim, H., and Wolf, L. (2019). Argonne Leadership Computing Facility Science Report (Tech. Rep. Argonne National Laboratory).

62. Shpyrko, O.G. (2014). X-ray photon correlation spectroscopy. J. Synchrotron Radiat. *21*, 1057–1064. https://doi.org/10.1364/JOSAA.375595.

63. Lehmkühler, F., Roseker, W., and Grübel, G. (2021). From femtoseconds to hours–measuring dynamics over 18 orders of magnitude with coherent x-rays. Appl. Sci. *11*, 6179. https://doi.org/10.3390/app11136179.

64. Perakis, F., and Gutt, C. (2020). Towards molecular movies with x-ray photon correlation spectroscopy. Phys. Chem. Chem. Phys. *22*, 19443–19453. https://doi.org/10.1039/D0CP03551C.

65. Zhang, Q., Dufresne, E.M., Nakaye, Y., Jemian, P.R., Sakumura, T., Sakuma, Y., Ferrara, J.D., Maj, P., Hassan, A., Bahadur, D., et al. (2021). 20 μs-resolved high-throughput x-ray photon correlation spectroscopy on a 500k pixel detector enabled by data-management workflow. J. Synchrotron Radiat. *28*, 259–265. https://doi.org/10.1107/S1600577520014319.

66. Diederichs, K., and Wang, M. (2017). Serial synchrotron X-ray crystallography (SSX). In Protein Crystallography (Springer), pp. 239–272. https://doi.org/10.1007/978-1-4939-7000-1_10.

67. Nam, K.H. (2022). Serial x-ray crystallography. Crystals *12*, 99. https://doi.org/10.3390/cryst12010099.

68. Uervirojnangkoorn, M., Zeldin, O.B., Lyubimov, A.Y., Hattne, J., Brewster, A.S., Sauter, N.K., Brunger, A.T., and Weis, W.I. (2015). Enabling x-ray free electron laser crystallography for challenging biological systems from a limited number of crystals. Elife *4*, e05421. https://doi.org/10.7554/eLife.05421.

69. Wilamowski, M., Sherrell, D.A., Minasov, G., Kim, Y., Shuvalova, L., Lavens, A., Chard, R., Maltseva, N., Jedrzejczak, R., Rosas-Lemus, M., et al. (2021). 2'-O methylation of RNA cap in SARS-CoV-2 captured by serial crystallography. Proc. Natl. Acad. Sci. USA e2100170118. https://doi.org/10.1073/pnas.2100170118.

70. Maiden, A.M., Humphry, M.J., Zhang, F., and Rodenburg, J.M. (2011). Superresolution imaging via ptychography. J. Opt. Soc. Am. Opt Image Sci. Vis. *28* (4), 604–612. https://doi.org/10.1364/JOSAA.28.000604.

71. Deng, J., Preissner, C., Klug, J.A., Mashrafi, S., Roehrig, C., Jiang, Y., Yao, Y., Wojcik, M., Wyman, M.D., Vine, D., et al. (2019). The Velociprobe: an ultrafast hard x-ray nanoprobe for high-resolution ptychographic imaging. Rev. Sci. Instrum. *90*, 083701. https://doi.org/10.1063/1.5103173.

72. Guan, Z., Tsai, E.H., Huang, X., Yager, K.G., and Qin, H. (2019). PtychoNet: fast and high quality phase retrieval for ptychography. In British Machine Vision Conference, p. 1172. https://doi.org/10.2172/1599580.

73. Nguyen, T., Xue, Y., Li, Y., Tian, L., and Nehmetallah, G. (2018). Deep learning approach for fourier ptychography microscopy. Opt Express *26*, 26470–26484. https://doi.org/10.1364/OE.26.026470.

74. Cherukara, M.J., Nashed, Y.S.G., and Harder, R.J. (2018). Real-time coherent diffraction inversion using deep generative networks. Sci. Rep. *8*, 16520. https://doi.org/10.1038/s41598-018-34525-1.

75. Bicer, T., Yu, X., Ching, D.J., Chard, R., Cherukara, M.J., Nicolae, B., Kettimuthu, R., and Foster, I.T. (2021). High-performance Ptychographic Reconstruction with Federated Facilities. https://doi.org/10.48550/arxiv.2111.11330.

76. Bernier, J.V., Barton, N.R., Lienert, U., and Miller, M.P. (2011). Far-field high-energy diffraction microscopy: a tool for intergranular orientation and strain analysis. J. Strain Anal. Eng. Des. *46*, 527–547. https://doi.org/10.1177/0309324711405761.

77. MIDAS. Microstructural Imaging Using Diffraction Analysis Software. https://www.aps.anl.gov/Science/Scientific-Software/MIDAS. (Accessed 28 March 2022). Accessed.

78. Liu, Z.. Demo of workflows for rapid NN training using remote data center AI systems. https://github.com/lzhengchun/nnTrainFlow. (Accessed 4 July 2022). Accessed.

79. Lauterbach, G. (2021). The path to successful wafer-scale integration: the Cerebras story. IEEE Micro *41*, 52–57. https://doi.org/10.1109/MM.2021.3112025.

80. Basney, J., Fleury, T., and Gaynor, J. (2014). CILogon: A federated X.509 certification authority for cyberinfrastructure logon. Concurrency Computat, Pract. Exper. *26*, 2225–2239. https://doi.org/10.1002/cpe.3265.

81. Withers, A., Bockelman, B., Weitzel, D., Brown, D., Gaynor, J., Basney, J., Tannenbaum, T., and Miller, Z. (2018). SciTokens: capability-based secure access to remote scientific data. In Practice and Experience on Advanced Research Computing, pp. 1–8. https://doi.org/10.1145/3219104.3219135.

82. Saracco, R. (2019). Digital twins: bridging physical space and cyberspace. Computer *52*, 58–64. https://doi.org/10.1109/MC.2019.2942803.

83. Niederer, S.A., Sacks, M.S., Girolami, M., and Willcox, K. (2021). Scaling digital twins from the artisanal to the industrial. Nat. Comput. Sci. *1*, 313–320. https://doi.org/10.1038/s43588-021-00072-5.

84. Dart, E., Rotman, L., Tierney, B., Hester, M., and Zurawski, J. (2014). The Science DMZ: a network design pattern for data-intensive science. Sci. Program. *22*, 173–185. https://doi.org/10.1145/2503210.2503245.

85. Gerhardt, L., Bhimji, W., Canon, S., Fasel, M., Jacobsen, D., Mustafa, M., Porter, J., and Tsulaia, V. (2017). Shifter: Containers for HPC. Journal of Physics: Conference Series, *898* (IOP Publishing), p. 082021. https://doi.org/10.1088/1742-6596/898/8/082021.

86. Uram, T.D., and Papka, M.E. (2016). Expanding the scope of high-performance computing facilities. Comput. Sci. Eng. *18*, 84–87. https://doi.org/10.1109/MCSE.2016.53.

87. Salim, M., Uram, T., Childers, J.T., Vishwanath, V., and Papka, M.E. (2019). Balsam: near real-time experimental data analysis on supercomputers. In 1st IEEE/ACM Annual Workshop on Large-scale Experiment-in-the-Loop Computing (IEEE), pp. 26–31. https://doi.org/10.1109/XLOOP49562.2019.00010.

88. Hightower, K., Burns, B., and Beda, J. (2017). Kubernetes: Up and Running Dive into the Future of Infrastructure, 1st ed. (O'Reilly Media, Inc.). ISBN 1491935677, 9781491935675.

89. Giannakou, A., Blaschke, J.P., Bard, D., and Ramakrishnan, L. (2021). Experiences with cross-facility real-time light source data analysis workflows. In IEEE/ACM HPC for Urgent Decision Making (IEEE), pp. 45–53. https://doi.org/10.1109/UrgentHPC54802.2021.00011.

90. Chard, K., Dart, E., Foster, I., Shifflett, D., Tuecke, S., and Williams, J. (2018b). The Modern Research Data Portal: a design pattern for networked, data-intensive science. PeerJ. Comput. Sci. *4*, e144. https://doi.org/10.7717/peerj-cs.144.

91. Chard, R., Vescovi, R., Du, M., Li, H., Chard, K., Tuecke, S., et al. (2018c). High-throughput neuroanatomy and trigger-action programming: a case study in research automation. In 1st International Workshop on Autonomous Infrastructure for Science, pp. 1–7. https://doi.org/10.1145/3217197.3217206.

92. Experimental Physics and Industrial Control System (EPICS). https://epics.anl.gov.

93. Allan, D., Caswell, T., Campbell, S., and Rakitin, M. (2019). Bluesky's ahead: a multi-facility collaboration for an a la carte software project for data acquisition and management. Synchrotron Radiat. News *32*, 19–22. https://doi.org/10.1080/08940886.2019.1608121.

94. Kodosky, J. (2020). LabVIEW. Proc. ACM Program. Lang. *4*, 1–54. https://doi.org/10.1145/3386328.

95. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A.Y. (2009). ROS: an open-source Robot operating system. In International Conference on Robotics and Automation, Workshop on Open Source Software, *3*, p. 5.

96. Brinson, L.C., Bartolo, L.M., Blaiszik, B., Elbert, D., Foster, I., Strachan, A., and Voorhees, P.W. (2022). FAIR Data Will Fuel a Revolution in Materials Research. https://doi.org/10.48550/arxiv.2204.02881.

97. DataCite Metadata Working Group (2021). DataCite Metadata Schema for the Publication and Citation of Research Data and Other Research Outputs. version 4.4. https://doi.org/10.14454/fxws-0523.

98. Ravi, N., Chaturvedi, P., Huerta, E., Liu, Z., Chard, R., Scourtas, A., Schmidt, K.J., Chard, K., Blaiszik, B., and Foster, I. (2022). FAIR Principles for AI Models, with a Practical Application for Accelerated High Energy Diffraction Microscopy. https://doi.org/10.48550/arXiv.2207.00611.

99. Schuh, H., and Behrend, D. (2012). A fascinating technique for geodesy and astrometry. J. Geodyn. *61*, 68–80. https://doi.org/10.1016/j.jog.2012.07.007.

100. Johnston, W.E., Greiman, W., Hoo, G., Lee, J., Tierney, B., Tull, C., and Olson, D. (1997). High-speed distributed data handling for on-line instrumentation systems. In ACM/IEEE Conference on Supercomputing (IEEE), p. 55. https://doi.org/10.1145/509593.509648.

101. von Laszewski, G., Su, M.H., Insley, J.A., Foster, I., Bresnahan, J., Kesselman, C., Thiebaux, M., Rivers, M.L., Wang, S., Tieman, B., et al. (1999). Real-time analysis, visualization, and steering of microtomography experiments at photon sources. In 9th SIAM Conference on Parallel Processing for Scientific Computing https://www.osti.gov/servlets/purl/752879.

102. Goscinski, W.J., McIntosh, P., Felzmann, U., Maksimenko, A., Hall, C.J., Gureyev, T., Thompson, D., Janke, A., Galloway, G., Killeen, N.E.B., et al. (2014). The multi-modal Australian ScienceS Imaging and Visualization Environment (MASSIVE) high performance computing infrastructure: applications in neuroscience and neuroinformatics research. Front. Neuroinform. *8*, 30. https://doi.org/10.3389/fninf.2014.00030.

103. Toby, B.H., Gürsoy, D., De Carlo, F., Schwarz, N., Sharma, H., and Jacobsen, C.J. (2015). Practices and standards for data and processing at the APS. Synchrotron Radiat. News *28*, 15–21. https://doi.org/10.1080/08940886.2015.1013415.

104. Dessy, R.E. (1977). Computer networking: a rational approach to lab automation. Anal. Chem. *49*, 1100A–1108A. https://doi.org/10.1021/ac50021a713.

105. Basu, S., Kaminski, J.W., Panepucci, E., Huang, C.Y., Warshamanage, R., Wang, M., and Wojdyla, J.A. (2019). Automated data collection and real-time data analysis suite for serial synchrotron crystallography. J. Synchrotron Radiat. *26*, 244–252. https://doi.org/10.1107/S1600577518016570.

106. Khan, F., Narayanan, S., Sersted, R., Schwarz, N., and Sandy, A. (2018). Distributed x-ray photon correlation spectroscopy data reduction using Hadoop MapReduce. J. Synchrotron Radiat. *25*, 1135–1143. https://doi.org/10.1107/S160057751800601X.

107. Benecke, G., Wagermaier, W., Li, C., Schwartzkopf, M., Flucke, G., Hoerth, R., Zizak, I., Burghammer, M., Metwalli, E., Müller-Buschbaum, P., et al. (2014). A customizable software for fast reduction and analysis of large x-ray scattering data sets: applications of the new DPDAK package to small-angle x-ray scattering and grazing-incidence small-angle x-ray scattering. J. Appl. Crystallogr. *47*, 1797–1803. https://doi.org/10.1107/S1600576714019773.

108. Gürsoy, D., De Carlo, F., Xiao, X., and Jacobsen, C. (2014). TomoPy: A framework for the analysis of synchrotron tomographic data.

J. Synchrotron Radiat. *21*, 1188–1193. https://doi.org/10.1117/12.2061373.

109. Deslippe, J., Essiari, A., Patton, S.J., Samak, T., Tull, C.E., Hexemer, A., Kumar, D., Parkinson, D., and Stewart, P. (2014). Workflow management for real-time analysis of lightsource experiments. In 9th Workshop on Workflows in Support of Large-Scale Science (IEEE), pp. 31–40. https://doi.org/10.1109/WORKS.2014.9.

110. Talirz, L., Kumbhar, S., Passaro, E., Yakutovich, A.V., Granata, V., Gargiulo, F., Borelli, M., Uhrin, M., Huber, S.P., Zoupanos, S., et al. (2020). Materials Cloud, a platform for open computational science. Sci. Data *7*, 299–312. https://doi.org/10.1038/s41597-020-00637-5.

111. Olds, D., Allan, D.B., Caswell, T.A., Lynch, J., Maffettone, P.M., and Campbell, S.I. (2021). Optimizing high-throughput capabilities by leveraging reinforcement learning methods with the Bluesky suite. In 3rd IEEE/ACM Annual Workshop on Extreme-scale Experiment-in-the-Loop Computing (IEEE), pp. 36–42. https://doi.org/10.1109/XLOOP54565.2021.00011.

112. Buurlage, J.W., Marone, F., Pelt, D.M., Palenstijn, W.J., Stampanoni, M., Batenburg, K.J., and Schlepütz, C.M. (2019). Real-time reconstruction and visualisation towards dynamic feedback control during time-resolved tomography experiments at TOMCAT. Sci. Rep. *9*, 18379–18411. https://doi.org/10.1038/s41598-019-54647-4.

113. Chung, J., Wisniewski, A., Zacherek, W., Liu, Z., Bicer, T., Kettimuthu, R., and Foster, I. (2022). SciStream: architecture and toolkit for data streaming between federated science instruments. In 31st ACM International Symposium on High-Performance Parallel and Distributed Computing, pp. 185–198. https://doi.org/10.1145/3502181.3531475.

114. Beck, M., Moore, T., Plank, J., and Swany, M. (2000). Logistical networking. In Active Middleware Services (Springer), pp. 141–154. https://doi.org/10.1007/978-1-4419-8648-1_12.

115. Barisits, M., Beermann, T., Berghaus, F., Bockelman, B., Bogado, J., Cameron, D., Christidis, D., Ciangottini, D., Dimitrov, G., Elsing, M., et al. (2019). Rucio: scientific data management. Comput. Softw. Big Sci. *3*, 11–19. https://doi.org/10.1007/s41781-019-0026-3.

116. Weitzel, D., Zvada, M., Vukotic, I., Gardner, R., Bockelman, B., Rynge, M., Fajardo Hernandez, E., Lin, B., and Selmeci, M. (2019). StashCache: a distributed caching federation for the open science grid. In Practice and Experience in Advanced Research Computing (ACM), pp. 1–7. https://doi.org/10.1145/3332186.3332212.

117. Harrow, J., Drysdale, R., Smith, A., Repo, S., Lanfear, J., and Blomberg, N. (2021). Providing a sustainable infrastructure for life science data at European scale. Bioinformatics *37*, 2506–2511. https://doi.org/10.1093/bioinformatics/btab481.

118. Bicarregui, J., Matthews, B., and Schluenzen, F. (2015). PaNdata: open data infrastructure for photon and neutron sources. Synchrotron Radiat. News *28*, 30–35. https://doi.org/10.1080/08940886.2015.1013418.

119. PaNdata - the Photon and Neutron data infrastructure initiative. http://pan-data.eu.

120. European Open Science Cloud (EOSC). Photon and Neutron Data Service. https://expands.eu.

121. Xu, H., Russell, T., Coposky, J., Rajasekar, A., Moore, R., de Torcy, A., Wan, M., Shroeder, W., and Chen, S.Y. (2017). iRODS primer 2: integrated Rule-Oriented data system. Synthesis Lectures on Information Concepts, Retrieval, and Services *9*, 1–131. https://doi.org/10.2200/S00760ED1V01Y201702ICR057.

122. European Open Science Cloud. https://eosc-portal.eu.

123. Barker, A., and Hemert, J.v. (2007). Scientific workflow: a survey and research directions. In International Conference on Parallel Processing and Applied Mathematics (Springer), pp. 746–753. https://doi.org/10.1007/978-3-540-68111-3_78.

124. Zhao, Y., Raicu, I., and Foster, I. (2008). Scientific workflow systems for 21st century, new bottle or new wine? In IEEE Congress on Services-Part I (IEEE), pp. 467–471. https://doi.org/10.1109/SERVICES-1.2008.79.

125. Deelman, E., Gannon, D., Shields, M., and Taylor, I. (2009). Workflows and e-Science: an overview of workflow system features and capabilities. Future Generat. Comput. Syst. *25*, 528–540. https://doi.org/10.1016/j.future.2008.06.012.

126. Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P.J., Mayani, R., Chen, W., Ferreira da Silva, R., Livny, M., and Wenger, K. (2015). Pegasus, a workflow management system for science automation. Future Generat. Comput. Syst. *46*, 17–35. https://doi.org/10.1016/j.future.2014.10.008.

127. Wilde, M., Foster, I., Iskra, K., Beckman, P., Zhang, Z., Espinosa, A., Hategan, M., Clifford, B., and Raicu, I. (2009). Parallel scripting for applications at the petascale and beyond. Computer *42*, 50–60. https://doi.org/10.1109/MC.2009.365.

128. Goecks, J., Nekrutenko, A., and Taylor, J.; Galaxy Team (2010). Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biol. *11*, R86. https://doi.org/10.1186/gb-2010-11-8-r86.

129. Thain, D., Tannenbaum, T., and Livny, M. (2005a). Distributed computing in practice: the Condor experience. Concurrency Computat, Pract. Exper. *17*, 323–356. https://doi.org/10.1002/cpe.938.

130. Frey, J., Tannenbaum, T., Livny, M., Foster, I., and Tuecke, S. (2002). A computation management agent for multi-institutional grids. Cluster Comput. *5*, 237–246. https://doi.org/10.1109/HPDC.2001.945176.

131. Stansberry, D., Somnath, S., Breet, J., Shutt, G., and Shankar, M. (2019). DataFed: towards reproducible research via federated data management. In International Conference on Computational Science and Computational Intelligence (IEEE), pp. 1312–1317. https://doi.org/10.1109/CSCI49370.2019.00245.

132. Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M.R., Wipat, A., and Li, P. (2004). Taverna: a tool for the composition and enactment of bioinformatics workflows. Bioinformatics *20*, 3045–3054. https://doi.org/10.1093/bioinformatics/bth361.

133. Foster, I., and Kesselman, C. (2011). The history of the grid. In High Performance Computing: From Grids and Clouds to Exascale (IOS Press), pp. 3–30. https://doi.org/10.48550/arxiv.2204.04312.

134. Shiers, J. (2007). The worldwide LHC computing grid (worldwide LCG). Comput. Phys. Commun. *177*, 219–223. https://doi.org/10.1016/j.cpc.2007.02.021.

135. Enders, B., Bard, D., Snavely, C., Gerhardt, L., Lee, J., Totzke, B., Antypas, K., Byna, S., Cheema, R., Cholia, S., et al. (2020). Cross-facility science with the superfacility project at LBNL. In 2nd IEEE/ACM Annual Workshop on Extreme-scale Experiment-in-the-Loop Computing (IEEE), pp. 1–7. https://doi.org/10.1109/XLOOP51963.2020.00006.

136. Cholia, S., Skinner, D., and Boverhof, J. (2010). NEWT: A RESTful service for building High Performance Computing web applications. In Gateway Computing Environments Workshop (IEEE), pp. 1–11. https://doi.org/10.1109/GCE.2010.5676125.

137. Stubbs, J., Cardone, R., Packard, M., Jamthe, A., Padhy, S., Terry, S., Looney, J., Meiring, J., Black, S., Dahan, M., et al. (2021). Tapis: an API platform for reproducible, distributed computational research. In Future of Information and Communication Conference (Springer), pp. 878–900. https://doi.org/10.1007/978-3-030-73100-7_61.

138. Nickolay, S., Jung, E.S., Kettimuthu, R., and Foster, I. (2021). Towards Accommodating Real-Time Jobs on HPC Platforms. https://doi.org/10.48550/arxiv.2103.13130.

139. Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., et al. (2016). Jupyter notebooks – a publishing format for reproducible computational workflows. In Positioning and Power in Academic Publishing: Players, Agents and Agendas, F. Loizides and B. Schmidt, eds. (IOS Press), pp. 87–90. https://doi.org/10.3233/978-1-61499-649-1-87.

140. Parkinson, D.Y., Krishnan, H., Ushizima, D., Henderson, M., and Cholia, S. (2020). Interactive parallel workflows for synchrotron tomography. In 2nd IEEE/ACM Annual Workshop on Extreme-scale

Experiment-in-the-Loop Computing (IEEE), pp. 29–34. https://doi.org/10.1109/XLOOP51963.2020.00010.

141. Thomas, R., and Cholia, S. (2021). Interactive supercomputing with Jupyter. Comput. Sci. Eng. *23*, 93–98. https://doi.org/10.1109/MCSE.2021.3059037.

142. Pordes, R., Petravick, D., Kramer, B., Olson, D., Livny, M., Roy, A., Paul Avery, Kent Blackburn, Torre Wenaus, Frank Würthwein, et al. (2007). The open science grid. In Journal of Physics: Conference Series, *78* (IOP Publishing), p. 012057. https://doi.org/10.1088/1742-6596/78/1/012057.

143. Jain, A., Ong, S.P., Hautier, G., Chen, W., Richards, W.D., Dacek, S., Cholia, S., Gunter, D., Skinner, D., Ceder, G., and Persson, K.A. (2013). Commentary: the Materials Project: a materials genome approach to accelerating materials innovation. Apl. Mater. *1*, 011002. https://doi.org/10.1063/1.4812323.

144. De Carlo, F., Gürsoy, D., Ching, D.J., Batenburg, K.J., Ludwig, W., Mancini, L., Marone, F., Mokso, R., Pelt, D.M., Sijbers, J., and Rivers, M. (2018). TomoBank: a tomographic data repository for computational x-ray science. Meas. Sci. Technol. *29*, 034004. https://doi.org/10.1088/1361-6501/aa9c19.

145. Blaiszik, B., Chard, K., Chard, R., Foster, I., and Ward, L. (2019a). Data automation at light sources. AIP Conference Proceedings, *2054* (AIP Publishing LLC), p. 020003.

146. Wilkins-Diehr, N., Gannon, D., Klimeck, G., Oster, S., and Pamidighantam, S. (2008). TeraGrid science gateways and their impact on science. Computer *41*, 32–41. https://doi.org/10.1109/MC.2008.470.

147. Marru, S., Gunathilake, L., Herath, C., Tangchaisin, P., Pierce, M., Mattmann, C., Singh, R., Gunarathne, T., Chinthaka, E., Gardler, R., et al. (2011). Apache Airavata: a framework for distributed applications and computational workflows. In ACM Workshop on Gateway Computing Environments, pp. 21–28. https://doi.org/10.1145/2110486.2110490.

148. Welch, V., Walsh, A., Barnett, W., and Stewart, C.A. (2011). A roadmap for using NSF cyberinfrastructure with InCommon. In TeraGrid Conference: Extreme Digital Discovery (ACM), p. 28. https://doi.org/10.1145/2016741.2016771.

149. Atherton, C.J., Barton, T., Basney, J., Broeder, D., Costa, A., van Daalen, M., Dyke, S., Elbers, W., Enell, C.-F., Fasanelli, E.M.V., et al. (2018). Federated Identity Management for Research Collaborations. https://doi.org/10.5281/zenodo.1307551.

150. Linden, M., Procházka, M., Lappalainen, I., Bucik, D., Vyskocil, P., Kuba, M., Silén, S., Belmann, P., Sczyrba, A., Newhouse, S., et al. (2018). Common ELIXIR service for researcher authentication and authorisation. F1000Res. *7*. https://doi.org/10.12688/f1000research.15161.1.

151. Umbrella. https://www.umbrellaid.org.

152. Gasser, M., and McDermott, E. (1990). An architecture for practical delegation in a distributed system. In IEEE Computer Society Symposium on Research in Security and Privacy (IEEE Computer Society), p. 20. https://doi.org/10.1109/RISP.1990.63835.

153. Foster, I., Kesselman, C., Tsudik, G., and Tuecke, S. (1998). A security architecture for computational grids. In 5th ACM Conference on Computer and Communications Security, pp. 83–92. https://doi.org/10.1145/288090.288111.

154. Welch, V., Foster, I., Kesselman, C., Mulmo, O., Pearlman, L., Tuecke, S., Gawor, J., Meder, S., and Siebenlist, F. (2004). X.509 proxy certificates for dynamic delegation. 3rd Annual PKI R&D Workshop, *14*.

155. Saint, N.D., Vescovi, R., Ryan, Jaligama, R.K., Kelly, D., Blaiszik, B., and Foster, I. (2022a). globus-gladier/gladier: Gladier v0.7.1.post1. https://doi.org/10.5281/zenodo.7042076.

156. Saint, N.D., Vescovi, R., Jaligama, R.K., Chard, R., Pruyne, J., Blaiszik, B., and Foster, I. (2022b). globus-gladier/gladier-tools: Gladier Tools v0.4.0.post1. https://doi.org/10.5281/zenodo.7042081.

157. Foster, I., Saint, N.D., Vescovi, R., Pruyne, J., Chard, R., and Blaiszik, B. (2022). globus-gladier/gladier-patterns-examples-2022: Gladier Examples - Cell Patterns Release v0.1.1. https://doi.org/10.5281/zenodo.7042063.

158. Vescovi, R., Saint, N.D., Parraga, H., Foster, I., Zhang, Q., Chard, R., Chu, M., and Blaiszik, B. (2022a). globus-gladier/gladier-xpcs: Gladier XPCS - Cell Patterns Release v0.1. https://doi.org/10.5281/zenodo.7042050.

159. Sharma, H., Vescovi, R., Chard, R., Saint, N., Blaiszik, B., and Foster, I. (2022). globus-gladier/gladier-hedm: Gladier HEDM - Cell Patterns Release v0.1. https://doi.org/10.5281/zenodo.7042054.

160. Vescovi, R., Chard, R., Saint, N.D., Blaiszik, B., and Foster, I. (2022b). globus-gladier/gladier-kanzus: Gladier Kanzus - Cell Patterns Release v0.1.1. https://doi.org/10.5281/zenodo.7042056.

161. Vescovi, R., Chard, R., Saint, N., Blaiszik, B., and Foster, I. (2022c). globus-gladier/gladier-nntrain: Gladier NNTrain - Cell Patterns Release v0.1.1. https://doi.org/10.5281/zenodo.7042065.

162. Vescovi, R., Chard, R., Saint, N., Blaiszik, B., and Foster, I. (2022d). globus-gladier/gladier-ptycho: Gladier Ptychography - Cell Patterns Release v0.1. https://doi.org/10.5281/zenodo.7044777.

163. Vescovi, R., Chard, R., Saint, N., Blaiszik, B., Pruyne, J., Bicer, T., Lavens, A., Liu, Z., Papka, M.E., Narayanan, S., et al. (2022e). Gladier - Linking Scientific Instruments and Computation - Example Data. https://doi.org/10.18126/17YW-EEHT.

164. Blaiszik, B., Chard, K., Pruyne, J., Ananthakrishnan, R., Tuecke, S., and Foster, I. (2016). The Materials Data Facility: data services to advance materials science research. JOM *68*, 2045–2052. https://doi.org/10.1007/s11837-016-2001-3.

165. Blaiszik, B., Chard, K., Chard, R., Foster, I., Ward, L., Pike, D., Chard, K., and Foster, I. (2019b). A data ecosystem to support machine learning in materials science. AIP Conf. Proc. *9*, 1125–1133. https://doi.org/10.1063/1.5084563.

Patterns, Volume *3*

Supplemental information

Linking scientific instruments and computation: Patterns, technologies, and experiences

Rafael Vescovi, Ryan Chard, Nickolaus D. Saint, Ben Blaiszik, Jim Pruyne, Tekin Bicer, Alex Lavens, Zhengchun Liu, Michael E. Papka, Suresh Narayanan, Nicholas Schwarz, Kyle Chard, and Ian T. Foster

Globus Flows web interface

Scientists need to be able not only to run flows but to detect, diagnose, and correct errors that may occur when a flow is executing. The Globus Flows service that we use to run flows provides such capabilities, as we illustrate in Figure S1, in which we (a) list recent runs, (b) inspect a summary of a run, and (c, d) list all actions involved in that run; and (e, f) examine actions performed in an unsuccessful run. Other displays, not shown here, allow for examination of flow definitions and input schema.

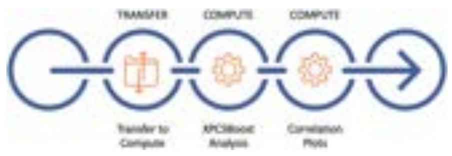
Running simplified versions of our five example applications

The five applications described in the paper have been developed to process big data streams from real light source instruments. To facilitate exploration, we also provide simple versions of each application that can be configured to run on a personal computer. These simplified versions are available on Zenodo at doi:10.5281/zenodo.7042063 and also directly on GitHub[1]; the sample data used in each is published on the Materials Data Facility at doi:10.18126/17YW-EEHT. These simplified applications do not deal with publishing flow products to a Globus Search catalog, and do not have an associated portal.

We first use a simplified version of the XPCS application described in the body of the paper to illustrate how the Gladier toolkit is used to implement a flow, and the process by which a flow is configured and run. Then, we provide brief notes on each of the other simplified applications.

The simplified XPCS application

The simplified XPCS application, `simple_xpcs_client.py`,[2] involves just three steps, as follows:

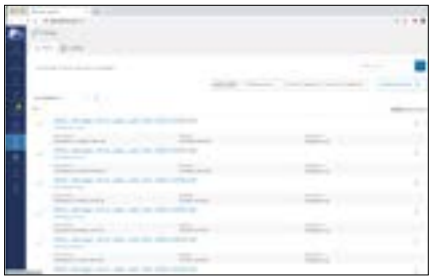


Consider first the following lines of `simple_xpcs_client.py`:

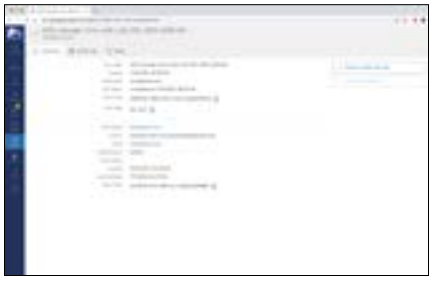
```
from gladier import GladierBaseClient, generate_flow_definition
from tools.xpcs_boost_corr import BoostCorr
from tools.xpcs_plot import MakeCorrPlots


@generate_flow_definition
class XPCSBoost(GladierBaseClient):
```

[1] https://github.com/globus-gladier/gladier-patterns-examples-2022

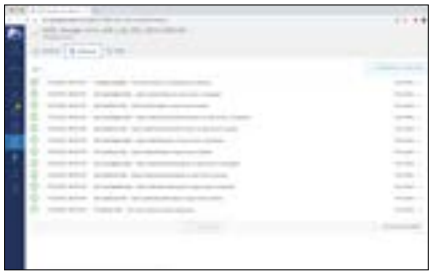
[2] https://github.com/globus-gladier/gladier-patterns-examples-2022/blob/main/simple_xpcs_client.py



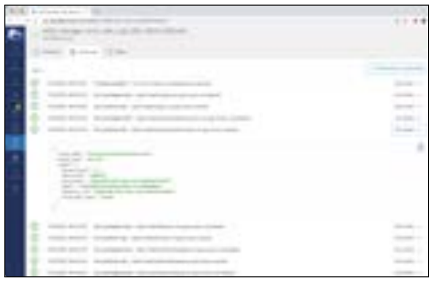
(a) The **Runs** tab in the Flows interface lists runs that I can view or manage. The **Library** tab lists flows that I can run.



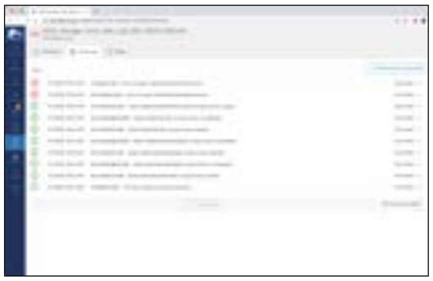
(b) Selecting a run in Figure S1a gives this status summary, with information on the run (above) and the flow that was run (below).



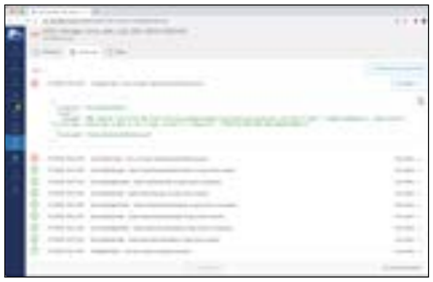
(c) Selecting the **Events** tab in Figure S1b gives this list of events during the run. We see that all completed successfully.



(d) Selecting a single event in Figure S1c provides additional information about the associated action: `PublishTransferSetPermission`.



(e) The events list for an alternative, unsuccessful run of the same flow indicates that an `PublishTransferSetPermission` action failed.



(f) Drilling down on the erroneous event in Figure S1e reveals (arguably opaque) information about the error: an invalid credential.

Figure S1: We use the example of an XPCS flow to illustrate how the Globus web interface enables tracking of flow progress and diagnosing of errors.

```
11    gladier_tools = [
12        "gladier_tools.globus.transfer.Transfer:FromStorage",
13        BoostCorr,
14        MakeCorrPlots,
15    ]
```

Lines 11-15 of this code uses the Gladier toolkit to specify a flow comprising the three tools shown in the figure:

1. A Transfer task to move data from a source storage location to a destination storage location (line 12). In a real deployment, the source will typically be a storage system associated with the scientific instrument and the destination a storage system associated with the data center where the analysis computer(s) are located.

2. A first Compute task to run the XPCSBoost Analysis program (line 13; imported, as specified in line 5, from `tools/xpcs_boost_corr.py`[3]).

3. A second Compute task to run the Correlation Plots program (line 14; imported, as specified in line 6, from `tools/xpcs_plot.py`[4]).

Subsequent statements in `simple_xpcs_client.py` configure various parameters, including the UUIDs that identify the funcX endpoint that is to be used to run the Compute tasks (`analysis_computer_funcx_id`) and the source and destination Globus collections (`instrument_computer_collection_id` and `analysis_computer_collection_id`) for the Transfer task. A value is already provided for `instrument_computer_collection_id`, the source of the data to be processed. Normally, this would be a storage system at the XPCS instrument, but it is configured in `simple_xpcs_client.py` to be a collection that we have established to store XPCS test data. Values are not provided, on the other hand, for `analysis_computer_collection_id` or `analysis_computer_funcx_id`. We will show in the next steps how to configure these on your personal computer.

When first initialized, this code generates a flow definition and registers it with the Globus Flows service. It also registers the two funcX tools, `BoostCorr` and `MakeCorrPlots`, with the funcX service. Subsequent invocations reuse the registered flow and functions.

The `simple_xpcs_client.py` application is easy to run on your own computer. The steps are:

1. **Establish the destination Globus collection**. As noted, the application needs a value for `analysis_computer_collection_id`, the identifier of a Globus collection accessible from the computer on which analysis tasks are to be executed. If no such collection is accessible to us, we can create a

[3] `https://github.com/globus-gladier/gladier-patterns-examples-2022/blob/main/tools/xpcs_boost_corr.py`

[4] `https://github.com/globus-gladier/gladier-patterns-examples-2022/blob/main/tools/xpcs_plot.py`

new collection by installing and configuring Globus Connect Personal software, as described online for Linux, MacOS, and Windows computers.[5] We then record the UUID for the collection by setting it as the value of `analysis_computer_collection_id` in the `simple_xpcs_client.py` application.

2. **Specify the funcX endpoint**. The application also needs a value for `analysis_computer_funcx_id`, the identifier of the funcX endpoint where analysis tasks are to be executed. If no such endpoint is accessible to us, we can create a new funcX endpoint on our personal computer by installing and configuring the funcX software, as described in the repository's `README.md` file.[1] We then record the UUID for the funcX endpoint by setting it as the value of `analysis_computer_funcx_id` in the `simple_xpcs_client.py` application.

3. **Configure execution environment on compute endpoint(s)**. The funcX system that we use to implement COMPUTE actions can run any Python functions or containerized programs invokable from Python that have been registered with the funcX service. We install programs that cannot be thus registered (e.g., a non-containerized application) manually prior to use, so that they may be invoked by COMPUTE actions during flow execution. Here we installed four such programs: the XPCS Boost correlation analysis tool,[6] CUDA Toolkit,[7] PyTorch,[8] and Gladier XPCS repository,[9] which includes custom plotting modules.

4. **Run the application**. We start the flow by executing the supplied `simple_xpcs_client.py`. When first invoked, the user is prompted to login and consent to the flow accessing the Transfer and funcX services. The application provides a link to the Globus Flows service where the flow can be monitored.

Other simplified applications

The **simplified SSX application** (specifically, a simplified version of the SSX-Stills flow described in the paper[10]) implements a flow with four steps: a TRANSFER from instrument to analysis computer followed by three COMPUTE steps that create a Phil-format[11] input file for the DIALS Stills application, run DIALS Stills, and run DIALS unit_cell_histogram, respectively.

[5] https://www.globus.org/globus-connect-personal
[6] https://github.com/AZjk/boost_corr
[7] https://developer.nvidia.com/cuda-toolkit
[8] https://pypi.org/project/torch/
[9] https://github.com/globus-gladier/gladier-xpcs
[10] https://github.com/globus-gladier/gladier-patterns-examples-2022/blob/main/simple_ssx_client.py
[11] https://cci.lbl.gov/docs/cctbx/doc_low_phil/

The **simplified HEDM application**[12] implements a flow with two steps: a TRANSFER from instrument to computer and a COMPUTE step that runs a supplied shell script.

The **simplified BraggNN application**[13] implements a flow with two steps: a TRANSFER from instrument to computer, and a COMPUTE step that runs a supplied shell script.

The **simplified Ptychography application**[14] implements a flow with three steps: a TRANSFER from instrument to computer, and two COMPUTE steps that run a supplied shell script and the `ptychodus_plot` tool,[15] respectively.

The full applications and flows described in the paper

As noted in the **Data and code availability** section of the paper, source code for the five applications described in this paper, plus information on how to run them, is available via Zenodo and GitHub. These production applications differ from our simplified applications in various ways. In particular, they:

- define separate funcX endpoints for non-compute-intensive and compute-intensive COMPUTE tasks, respectively (on an HPC system, these will typically correspond to a login node vs. compute nodes); and
- publish descriptive metadata plus data references to a Globus Search catalog, and establish an associated interactive data portal, so that users can browse, search, and access flow products.

The following information on configuring and running one of these applications, XPCS, is also relevant to the other applications. This application[9] supports the processing of XPCS data generated at the 8-ID beamline of the Advanced Photon Source (APS). The generation of spectroscopy data at 8-ID triggers a flow that transfers data from 8-ID to ALCF for analysis, metadata extraction, and visualization, and then publishes the processed data to an ALCF Community Data Co-Op[16] portal.

The Python program `flow_boost.py`[17] implements the flow described in the paper, with the addition of a step 5 to preallocate nodes on the HPC resource, an optimization that can accelerate flow start. Some notes about how to configure the flow to run:

[12] `https://github.com/globus-gladier/gladier-patterns-examples-2022/blob/main/simple_hedm_client.py`
[13] `https://github.com/globus-gladier/gladier-patterns-examples-2022/blob/main/simple_braggnn_client.py`
[14] `https://github.com/globus-gladier/gladier-patterns-examples-2022/blob/main/simple_ptycho_client.py`
[15] `https://github.com/globus-gladier/gladier-patterns-examples-2022/blob/main/tools/ptychodus_plot.py`
[16] `https://acdc.alcf.anl.gov`
[17] `https://github.com/globus-gladier/gladier-xpcs/blob/main/gladier_xpcs/flows/flow_boost.py`

1. **Configure infrastructure**: The XPCS flow involves Transfer, Compute, and Search actions.

    - As the flow involves Transfer actions, we must ensure that **Globus collections** are in place wherever data are to be accessed: in this case, the APS 8-ID beamline and ALCF Eagle storage systems. As Globus collections are already deployed in both locations as part of their regular infrastructure, no action was required.
    - As the flow involves Compute actions, we must ensure that **funcX endpoints** are deployed wherever computation is to be performed: in this case, the ALCF Theta computer. The endpoint must also be configured to interface with the batch scheduler to appropriately acquire nodes. Here, we define a Cobalt configuration using the example in the funcX documentation.[18]
    - As the flow involves Search actions, we must ensure that a **Globus Search index** has been provisioned and a data portal deployed and customized to visualise search records. The XPCS search index was created via the Globus CLI.[19] The XPCS data portal was implemented by using the Django Globus Portal Framework,[20] with customization to display specific metadata, facets, and images. The portal implementation and installation instructions are on Github.[21]

2. **Configure execution environment on compute endpoint(s)**. As with the simplified XPCS application, we install four programs that cannot be registered automatically with the funcX service: the XPCS Boost correlation analysis tool,[6] CUDA Toolkit,[7] PyTorch,[8] and the Gladier XPCS repository,[9] which includes custom plotting modules.

3. **Configure flow triggers**. A trigger may be configured to invoke an instance of a flow in response to data being generated. In this example, instances of the flow are initiated by the APS Data Management System,[22] which copies each batch of new images, as they are acquired, from the instrument to storage accessible by Globus Transfer, and then starts an instance of the flow.

[18] https://funcx.readthedocs.io/en/latest/endpoints.html#theta-alcf
[19] https://docs.globus.org/cli/reference/search_index_create/
[20] https://github.com/globus/django-globus-portal-framework
[21] https://github.com/globus-gladier/gladier-xpcs/tree/main/xpcs_portal
[22] S. Veseli, N. Schwarz, C. Schmitz. "APS data management system," Journal of Synchrotron Radiation 25(5):1574-1580, 2018, https://doi.org/10.1107/S1600577518010056.

Other software referenced in, or relevant to, the paper

The **Gladier Toolkit**[23,24] (see body of paper) is designed to accelerate and simplify the implementation of new scientific flows for experimental facilities. It provides a Pythonic interface for defining Globus flows and for managing the registration and caching of flows and of funcX functions.

The supporting **Gladier Tools**[25,26] package utility tools that can be incorporated into a flow, such as Transfer and Publication. Tools in this repository are intended to be general purpose and reusable.

The **Globus Python SDK**[27,28] provides a convenient Pythonic interface to Globus web APIs, including the Globus Transfer API and Globus Auth API. It is used extensively by the Gladier Toolkit and tools.

The **Globus Automate CLI and SDK**[29,30] provides a command line interface (CLI) and Python software development kit (SDK) for working with Globus automation services, primarily Globus Flows, any service implementing the Globus Action Provider interface, and Globus Queues.

The **Globus Sample Data Portal**[31,32] implements a simple Web app framework that illustrates how to build a data portal, such as those created for the example applications presented in this paper, by using Globus services.

The **Django Globus Portal Framework**[33,34] provides a modular framework for building Globus-based data portals. It provides utilities for quickly building a data portal around a Globus Search index, using Globus Auth to secure access to data.

[23] `https://github.com/globus-gladier/gladier` September 1, 2022 version archived at https://doi.org/10.5281/zenodo.7042076

[24] `https://gladier.readthedocs.io`

[25] `https://github.com/globus-gladier/gladier-tools` September 1, 2022 version archived at https://doi.org/10.5281/zenodo.7042081

[26] `https://gladier.readthedocs.io/en/latest/gladier_tools`

[27] `https://github.com/globus/globus-sdk-python`

[28] `https://globus-sdk-python.readthedocs.io`

[29] `https://github.com/globus/globus-automate-client`

[30] `https://globus-automate-client.readthedocs.io`

[31] `https://github.com/globus/globus-sample-data-portal`

[32] `https://docs.globus.org/modern-research-data-portal`

[33] `https://github.com/globus/django-globus-portal-framework`

[34] `https://django-globus-portal-framework.readthedocs.io`