Data Object Distribution for Experimental Science Pipelines

Justin M. Wozniak,[1] Ray Osborn,[2] and Jacob Ruff[3]
[1] Data Science and Learning Division (ANL), [2] Materials Science Division (ANL),
[3] Cornell High Energy Synchrotron Source

Topic: Fine-grained, wide area data management and access for experimental science.

In order to perform both data reduction and more advanced analysis, such as machine learning (ML), on data collected at major facilities such as the Advanced Photon Source, it is largely the responsibility of the visiting scientists to navigate the patchwork of computational resources available at facilities themselves or to transfer large volumes of data to their own institutions. If an experiment involves multiple collaborators who each contribute to different modes of data analysis, the experimental team may have to transfer TBs of data to multiple locations. For example, a team in Argonne's Materials Science Division regularly collects several TB/day at Sector 6 of the APS. This entire volume is currently streamed to an APS data cluster for initial data reduction, then transferred to MSD for spectral analysis and, in parallel, copied to Cornell for ML via Gaussian mixture modeling (GMM) for unsupervised Bragg peak clustering, then to ALCF Theta to fit parameters. The raw data are archived by the APS for 1-2 years, but the intermediate results from each stage of analysis are distributed over multiple machines in multiple locations, as shown in Figure 1. Furthermore, the rate-limiting step for each stage of analysis is the data transfer (1 day of transfer for 15 minutes of analysis!). The Management and Storage of Scientific Data community could have a big impact on improving the data movement costs of this workload by enabling and optimizing remote data operations for a range of data creation and access patterns.

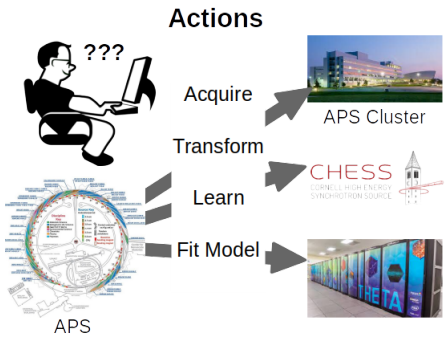

Figure 1: Multiple stages in data analysis workflow distributed across multiple computing sites, each with differing processing and I/O capabilities. Whole data sets are copied in bulk, incurring complexity and inefficiencies.

This white paper sketches an alternative paradigm that could profoundly impact how facilities handle large-scale experimental data in the future. In our approach, users are presented with a unified view accessible to all collaborators, which is populated on-demand with experimental data and/or theoretical simulations from multiple sources, and synchronized with distributed remote locations (cloud, exascale computing facilities) as needed. From the user perspective, this will have three advantages: (1) accelerated analysis and learning pipelines due to reduced data transfer overheads; (2) improved integration of data analysis and advanced modeling (digital twins), and (3) increased productivity due to less management overhead and development overhead. To this end, we need to design and develop a portable, adaptable infrastructure that exposes high-level data manipulation primitives to filter, query and assimilate simulation/experiment/collaborator data into unified views at fine granularities, while optimizing interactions with the remote data sources through a combination of local caching and data planning strategies.

The choices a scientist makes in performing this post-pipeline analysis are usually specific to the particular scientific question being addressed by the experiment, and therefore impossible to encapsulate in a predefined pipeline. We envision that pipeline construction by exposing Python network-accessible object interfaces. The community is already moving toward Python for sequential processing and basic

data manipulation, while performance-critical sections are expected to be written in C, C++, or Fortran, and exported to Python. Such libraries are likely to be usable in other settings, such as high-performance computers. The ongoing development of high resolution, high frame rate detectors and the x-ray and neutron scattering science capabilities they offer has created a data management and I/O challenge. Large experimental datasets must be rapidly stored and managed for near-real time analysis. The construction of analysis pipelines partially automates the analysis process, transferring data to storage and among multiple analysis software packages, but approaches are often too rigid for dynamic studies by human or learning agents.

It is as important therefore that we have a plan for ensuring that the results of any pipeline are accessible in a convenient and reasonably standardized form as it is to develop tools for constructing the automated pipelines in the first place. It is no longer sufficient to deposit the data in an archive, if the user is then required to download it in order to perform any followup operations. Current methods of data management at large-scale facilities have not adapted to the needs of facility users as data volumes have grown and the speed of data collection accelerated. Facilities typically provide medium-term stores to archive the data, but these are only accessible as file repositories through SFTP or Globus.

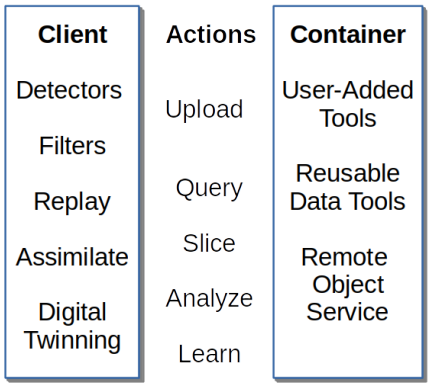

Figure 2: Multiple stages in data analysis workflow supported by remote data tools served from the containerized service library, including Pythonic objects for learning from small slices of large data.

We have to develop an I/O abstraction and architecture that allows fine-grained access to the data without requiring significant data transfers. Thus we propose several key I/O abstraction challenges: 1) Detector **ingest, filtering and upload (IFUP)**, via Pythonic filter plug-ins wrapped around advanced buffering and staging with multiple back-ends; 2) **Fine-grained local caching** that maintains the illusion of a fully available local view, but efficiently populates it on-demand and maintains its coherence in the background; 3) **Orchestration of remote data sources,** which automatically selects the best remote source to interact with (based on proximity, availability of data, permissions) and keeps the remote data sources synchronized; 4) **High level indexing and query support** to extract, group and present the data to analytics and learning tasks using familiar plug-and-play abstractions, from multiple data services.

Our approach is to develop a toolkit of learning and analysis-ready Python libraries that can be integrated into workflows by users from a broad range of disciplines, depicted in Figure 2. We envision a pluggable framework in which user Python fragments can be wrapped around existing IFUP technologies (e.g., Globus) but with a range of callback points controllable via Python tools. Fine-grain local caching will be addressed at lower levels, investigating how to efficiently buffer and issue fine-grain put/get operations in bulk to hide the latency of accessing remote data sources without compromising coherence. To this end, we envision approaches based on snapshot isolation that can take advantage of related efforts. Enhanced data analysis and learning will be enabled through extensions to a remote object toolkit which provides a familiar numerical interface, and optimizable aggregated data pipelines for deep learning frameworks. For example, in the GMM case, a container library entry for Bragg peak identification would be run at data ingest time, and the learning agent would walk the resulting peak index over remote object interfaces to quickly and efficiently produce usable results which are also stored for use by others.