Title
Cross-facility science with the Superfacility Project at LBNL

ISBN
978-1-6654-2282-6

Authors
Enders, Bjoern
Bard, Debbie
Snavely, Cory
et al.

Publication Date
2020-11-12

DOI
10.1109/xloop51963.2020.00006

Peer reviewed

Cross-facility science with the Superfacility Project at LBNL

Bjoern Enders*, Debbie Bard*, Cory Snavely*, Lisa Gerhardt*, Jason Lee*, Becci Totzke*, Katie Antypas*, Suren Byna*, Ravi Cheema*, Shreyas Cholia*, Mark Day*, Aditi Gaur*, Annette Greiner*, Taylor Groves*, Mariam Kiran*, Quincey Koziol*, Kelly Rowland*, Chris Samuel*, Ashwin Selvarajan*, Alex Sim*, David Skinner*, Rollin Thomas* and Gabor Torok*
* Lawrence Berkeley National Lab
Berkeley, California 94720
Email: benders@lbl.gov

***Abstract*—As data sets from DOE user science facilities grow in both size and complexity there is an urgent need for new capabilities to transfer, analyze and manage the data underlying scientific discoveries. LBNL's Superfacility project brings together experimental and observational research instruments with computational and network facilities at the National Energy Research Scientific Computing Center (NERSC) and the Energy Sciences Network (ESnet) with the goal of enabling user science. Here, we report on recent innovations in the Superfacility project, including advanced data management, API-based automation, real-time interactive user interfaces, and supported infrastructure for "edge" services.**

I. Introduction

The Superfacility concept is a framework for integrating experimental and observational instruments with computational and data facilities. We work in an ecosystem of connected facilities, where data produced by microscopes, telescopes and other devices can stream in real-time to large computing facilities where it can be analyzed, curated, combined with simulation data and served to the science user community via powerful computing, storage and networking systems. This model is more than the sum of its parts, enabling discoveries across data sets, institutions and domains and making data from one-of-a-kind facilities and experiments broadly accessible. In a not-so-distant future, we anticipate that scientists will interact with the computing facilities directly through their experimental facility, without direct access or accounts at an HPC center.

The LBNL Superfacility project is designed to identify technical and policy challenges in this concept for connected facilities. Superfacility project activities coordinate and manage the work needed to address these challenges in partnership with target science teams that provide requirements and use cases for our development work. We are working with facilities across the DOE Office of Science, in cosmology (DESI [1], LSST-DESC [2]), dark matter detection (LZ [3]), lightsources (ALS [4] and LCLS [5]) and electron microscopes (NCEM [6]). The project is designed to ensure that the solutions developed are widely useful - scalable to multiple user groups, and scalable for NERSC staff to support.

Since its inception in 2018, the project has made substantial advances in our target capabilities. This paper describes our work so far in the following areas:

1) The "Superfacility API", a REST interface into NERSC.
2) Data management tools and capabilities.
3) Spin: an "Edge Services" platform based on containers.
4) Interactivity with Jupyter.

II. The Superfacility API: Automation for Complex Workflows at Scale

Complex workflows typically involve tasks that are impractical to do manually: tracking hundreds or thousands of job runs, adjusting job run parameters in real time based on initial results or other conditions, coordinating data movement etc. Ideally, workflow developers would automate as much of their interaction with HPC systems as possible.

The Superfacility API seeks to enable the automated use of all NERSC resources using common development tools and techniques. An evolution of its predecessor, NEWT [7], the newly-designed API adds features designed to support complex distributed workflows, such as placing future job reservations. It will also allow users to offload tedious manual tasks such as data movement via simple REST calls. We have designed the API endpoints and operations to cover a wide range of use cases voiced by our user science teams.

TABLE I
API endpoints.

`/accounting`	retrieve allocation info for a user or project
`/callbacks`	register callbacks for asynchronous or chained operations
`/file`	browse, upload, and download files
`/health`	retrieve system health status, including planned outages
`/jobs`	submit jobs and check job status
`/transfer`	move data between sites with Globus, or between NERSC storage tiers with data movement tools (see Section III)
`/reservations`	submit and manage future compute reservations

A. API endpoints

Inspired by NEWT, we defined a set of endpoints (listed in Tab. I), each serving a specific purpose. Most of these endpoints require the user to authenticate with a JSON web token (JWT) [8], as described in Section II-B.

These endpoints are under active development. The web interface[1] comes with documentation and the ability to query the API in the browser window. Table II lists use cases that can be made faster, more robust or simplified with a proper API endpoint.

B. Technical Details

The Superfacility API is a Python Flask app built on top of the restx library [9]. The library provides the framework and the Swagger-compatible documentation [10]. The Python service and other auxiliary services (database, HTTP proxy) are hosted in Spin (see Section IV for more details) which provides load balancing and HTTPS termination. All API calls are blocking with the exception of the use of a simple, database-backed message queue to handle long transactions, e.g. running jobs or transferring data.

The API uses OAuth 2 tokens [11] for authorization. Apart from being an open standard, OAuth 2 is an attractive authorization framework for several reasons.

- *Client Credentials Grant*: Individual users directly using the API to perform work with their own allocation and data obtain a client identifier and key in Iris [12]. Workflow code or scripts then use these credentials to request short-lived access tokens from the NERSC OAuth 2 server, which are supplied to the API for authorization.
- *Authorization Code Grant*: Workflow managers that perform work on behalf of many different users and across allocations can utilize this model, which allows a user to authorize actions to be taken on their behalf without exposing passwords or other credentials. Following a web-based approval, the workflow manager obtains a refresh token from the OAuth 2 server that corresponds to the granting user; then, as in the first model above, this token is used to obtain access tokens which are presented to the API for authorization.

[1]https://api.nersc.gov/api/v1

TABLE II
COMPARISON OF NERSC USE CASES WITH AND WITHOUT API.

Before Superfacility API	With Superfacility API
Read the NERSC MOTD with your eyes and see if any down time is planned	Makes a Python Requests call to fetch planned maintenance in JSON.
Test SSH or ping specific services for status	Query the `/health` API endpoint.
SSH in and submit jobs with `sbatch ...`	Manage jobs using Requests calls from a script or Spin service using the `/jobs` endpoint.
SSH in (again) and do `squeue | grep | sort | ...`	Create a Spin service using the Superfacility API to present a custom view of workflow progress.
File a ticket to create a reservation, then wait	Make a Requests call against the /reservations API endpoint.
SSH in and run file transfer tools to move data	Use the `/transfer` API endpoint.
Chain jobs, maintain job state (e.g. in a database), check state when a cron job wakes up to coordinate workflow.	Register callbacks at the `/callback` API that fire when steps complete.

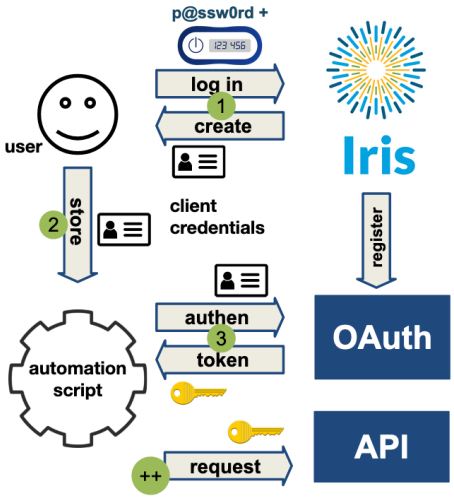

Fig. 1. Illustration of a user creating an OAuth client in Iris, then using the client credentials to obtain an access token for automated API operations.

The OAuth 2 model not only provides a fitting authorization solution for both single-individual and multi-user workflow managers (see Figure 1), but the short-lived access tokens used by both models allow for quick access revocation, satisfying the strict security requirements for HPC centers.

C. An API for the future

Automation has emerged as a critical need for experiment data analysis pipelines, and can no longer be considered an optional part of HPC infrastructure [13]–[15]. The Superfacility API thus meets critical needs of our partner science teams. In the long run, we envisage all NERSC interactions will be callable via policy-driven APIs. Backend tools will assist with large or complex operations, keeping the API interpretable and not just a one-to-one exposure of system tools. As a stable and understandable interface, the API can more readily be integrated into the tools and UIs of our user community, creating a "NERSC inside" experience for their users. Adoption will be simplified further by using standardized tooling (Python/REST ecosystem) which will make it easier for middleware developers to refactor their software or lower the entry barrier for new projects. Embracing established authentication and security models such as OAuth 2 [11], OpenID Connect [16], and JWTs allows software developers to make use of standard libraries and programming methods, reducing the time spent on these tedious aspects of system-building and leveraging the strengths of open source - broad community adoption and testing - to avoid security risk.

III. DATA MANAGEMENT TOOLS AND CAPABILITIES

NERSC offers 35 PB of Lustre [17] and 60 PB of Spectrum Scale (formerly GPFS) [18] disk storage known as the Community File System (CFS). NERSC also has more than 150 PB of tape storage in a High-Performance Storage System (HPSS) [19] archiving system, and 1.8 PB of NVMe Burst Buffer storage [20]. Future projections estimate that the volume will increase by a factor of 10 - 100 by 2025 [21].

While scientists have grown increasingly adept at producing data, there has been no commensurate increase in the sophistication of the tools available to manage data sets at the HPC facility. As a result, scientists spend a large fraction of their time in manual data management. For instance, scientists often have only "du" and "ls" to examine tens to hundreds of terabytes of data spread across tens of millions of files at NERSC. They must manually move this data through a multi-level storage hierarchy (i.e., from high-speed scratch to a long-term tape archive) and manually verify that the transfer succeeded. The commands can often take hours to days to complete, making data management a time-consuming chore and management of petabyte-size data volumes untenable. Like other efforts [22], the Superfacility project aims to take the pain out of data management and leave the scientist with more time to do what really matters: Science.

A. UI improvements: Data Dashboard and PI Toolbox

1) Data Dashboard: The Data Dashboard offers a graphical view into a project's data usage as can be seen in Fig. 2 and Fig. 3. The visualizations in the dashboard are updated by scripts that leverage file scan data stored in either text files (for usage information) or a Postgresql database (for largest files and file browser visualizations). The data from the scans is processed daily by a set of scripts that run Spark jobs and load the resulting data. Access to information from the database and from the stored text files is mediated by API calls. At present, the dashboard makes these calls via the NEWT API, but we are in the process of moving to the Superfacility API.

2) PI Toolbox: The PI Toolbox is a web portal that allow PIs to address many of the common issues with permission drift and ownership changes on CFS. The current process for these actions is handled via active ssh sessions for user-privilege operations (like chmod) and open support tickets for

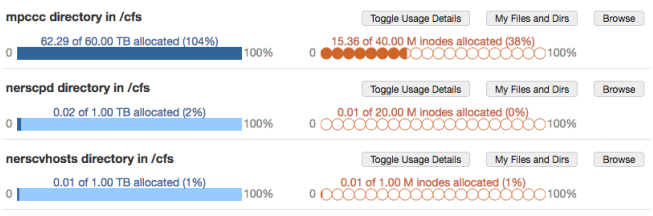

Fig. 2. Data Dashboard. Upon login, the user is shown a list of projects to which they have permissions, and each project can be inspected for details about the current usage of both space and inodes relative to their current quota. One can also view the breakdowns of usage by individual users or by unix groups within the collaboration.

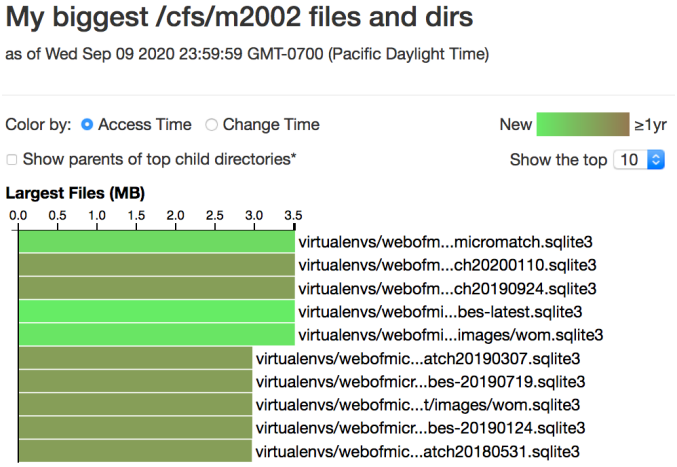

Fig. 3. The Data Dashboard also provides a view of the user's largest files, along with age information visualized by color, enabling rapid identification of candidates for removal.

elevated privileges (like chgrp or chown). The PI Toolbox adds file-browser convenience and automation to this process. It allows PIs to change permissions on their files without having to open a ticket for the following operations:

- chgrp
- chmod (group level)
- chown *(under development)*
- All-in-one request to assign a group and group-read permission for an entire project directory
- Any of the above at any level of recursion.

Under the hood, the web portal tracks requests and updates their status in a database. A background process executes these requests in the order they are received and reports success or any errors back to the database.

B. Globus Sharing endpoints for collaboration accounts

Many users from our scientific partner facilities use collaboration accounts to manage their data at NERSC. A collaboration account is a login account which is not tied to a person but instead to the group for the purposes of shared access to batch jobs or data. Members of an individual collaboration linux group can impersonate the collaboration user via a modified `su` command, `collabsu`. This abstraction layer serves a critical need when the data is created by an experiment or observations, and not necessarily at the hand of a particular user. This eases burdens of data management, but Globus [23] transfers into NERSC using a collaboration account as target previously required several levels of human intervention:

- Transfer in with user credentials.
- File a ticket to get the data chowned to the collaboration account.

NERSC now provides a new program which uses Globus Sharing to create dedicated endpoints that will write as the collaboration account with the same access controls as `collabsu`:

- Only NERSC users who are in the collaboration linux group can access the endpoint.
- Read / write access to all files and directories at NERSC the collab account can access.

Collaboration endpoints are created and maintained by a suite of custom scripts that use the Globus SDK. Collaboration endpoints for Globus have been deployed in production and the LSST-DESC collaboration has used it with great success to move PB of data into NERSC.

C. Globus Command Line Tools

NERSC has written several command line data transfer scripts so users can integrate data transfers into their workflows. These scripts are based on the Globus SDK package and can manage transfers inside and outside NERSC. It has defaults for NERSC endpoints and moves data into and out of HPSS in an optimal ordered way. These scripts will serve as the backbone for the ongoing effort to integrate data movement into the batch system, and are being integrated into the Superfacility API.

D. GPFS-HPSS interface (GHI)

GHI is a new GPFS / HPSS interface that offers the benefits of a more familiar file system interface for HPSS. NERSC users often want to store complex directory structures or large bundles in HPSS which can be difficult to do with the traditional HPSS access tools (hsi, htar). GHI works with a small set of commands (see Tab. III) and can be used to easily move data between HPSS and the GPFS file system. It offers a set of compelling advantages:

- Users interact with a familiar file system directory that's tied to HPSS behind the scenes.
- GHI puts the files optimally into HPSS on your behalf, no need to htar small files together or break things into appropriately-sized chunks.
- GPFS and HPSS are combined in a shared namespace, so deleting a file from GPFS will remove it from HPSS.

To illustrate the capabilities of GHI, we describe here two common use cases for experimental or observational facilities.

- *Archiving Complex Directory Structures*. Experiments at the ALS at LBNL often has complex sets of microscopy image data. An experiment may produce large volumes of image data in tens of MBs along with a few kB-sized text files containing scan parameters. The files are organized in a complex metadata-based directory structure that must be maintained. Now, with a single command (ghi put) the user is able to archive the entire directory into HPSS without having to consider bundling or sorting the files.
- *Large volumes of Infrequently Accessed Data*. Experiments like DESI perform annual reprocessing of large (100TBs) datasets that are otherwise not accessed frequently. The user can now move data into HPSS with ghi put. In between analyses, the user frees up disk space by using ghi punch to move most data off of GPFS, leaving behind a browsable directory structure. After most of the initial work is done the user selectively retrieves files with ghi stage for in-depth reanalysis or download.

GHI has been released to select partners from ALS, LSST-DESC, LZ and others for testing. Their feedback will guide the shape of the GHI deployment, but we expect to add GHI as a separate file system to offer an easier way for users to archive their data.

TABLE III
GHI COMMANDS

`ghi ls`	show what file system the files are currently on, files are marked 'G' for GPFS, 'H' for HPSS and 'B' for both
`ghi put`	copy the files to HPSS, makes files dual residents on both files systems
`ghi stage`	move the files back from HPSS to GPFS
`ghi punch`	move the files to HPSS (leaving a stub behind on GPFS)
`ghi pin`	keep the files from being removed from GPFS
`ghi lock`	keep the files from being removed using rm or otherwise modified

E. HDF5 Features and Improvements

Hierarchical Data Format version 5 (HDF5) [24], [25] is the most popular I/O middleware and file format used at NERSC. HDF5 provides a data model, library, and file format along with a rich programming interface for storing and managing data. Due to the versatility of its data model, and its portability, longevity, and efficiency, many applications (including some superfacility science partners), use HDF5 as part of their data management solution. In Fig. 4, we show a use case where a detector sends data to a HDF5 application that writes data to a storage system, and uses multiple reader applications to access data for analysis.

We developed several new features in HDF5 to facilitate SF use cases. Among those, a few of the most are:

- Development of the HDF5 *mirror* Virtual File Driver (VFD) [24], which allows transmitting write-only opera-

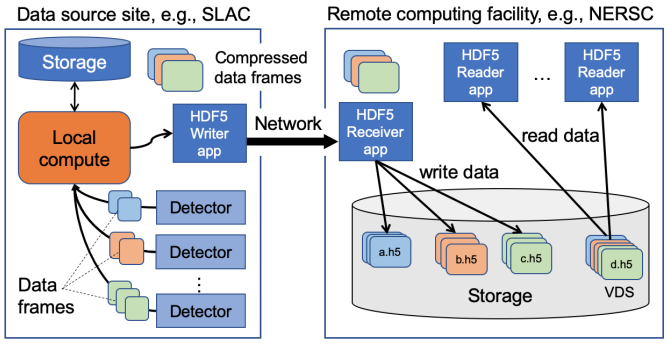

Fig. 4. Data flow from detectors to remote storage when using HDF5 - Compressed data frames are send over the network to an HDF5 writer application, which writes the data as HDF5 dataset objects in files that correspond to each detector. Virtual datasets (VDS) are used by readers, to compose a unified view of the detector data frames for applications at the remote facility.

tions at a science facility to a receiver or writer process located at a remote host;
- To search the metadata (attributes and values) of a large number of HDF5 files, we developed a new in-memory indexing data structure that performs three orders of magnitude faster than databases such as MongoDB [26];
- To improve the performance of streaming applications that append data to existing datasets, we profiled the HDF5 library to identified bottlenecks. By optimizing these append operations, we achieved 10X performance improvement to application I/O.

Collectively, these features and performance optimizations are aimed towards improving productivity and performance of Superfacility-type workloads.

IV. Spin: An "Edge Services" Platform Based on Container Technology

Complex, distributed workflows typically require resources beyond those provided by conventional HPC. In addition to batch compute, high-performance networking, and large-scale storage, these projects need reliable and persistent services to manage and track job execution, coordinate data transfers, store interstitial or ephemeral data, or provide access to results. These "edge services" may be workflow engines, API servers, databases, web-based science gateways, or other services.

Providing and supporting infrastructure for these types of services presents a classical challenge. Edge services must be adjacent to HPC storage and networks to interact directly with datasets, but placing them inside the supercomputer presents problems with security and provides poor availability (uptime). The support of the highly customized software that edge services require takes significant effort from HPC operational staff for even a handful of projects.

Spin [27] offers a solution to these problems and a scalable home for edge services by placing a secure, managed, container-based platform outside - but in direct proximity to - HPC resources. Using Docker [28] container images and a high-level CLI and UI, users are able to deploy their own custom software and services within reach of NERSC networks and storage without any administrative access to worker nodes. Staff manage all of the underlying infrastructure, which is agnostic to the specifics of service workloads.

A. System Components

The key components of Spin are as follows:

- *Orchestration*: the core of the system, responsible for starting and monitoring containers on Spin nodes, including failover and fault recovery. This component also provides the user-facing command-line interface (CLI) and web user interface (UI) where users create and manage services. Spin uses the Rancher [29] orchestration system, which embeds the Kubernetes [30] scheduler.
- *Image Registry*: a versioned repository for Docker images, which typically correspond to individual microservices (e.g. Apache frontends, MongoDB instances and application backends).

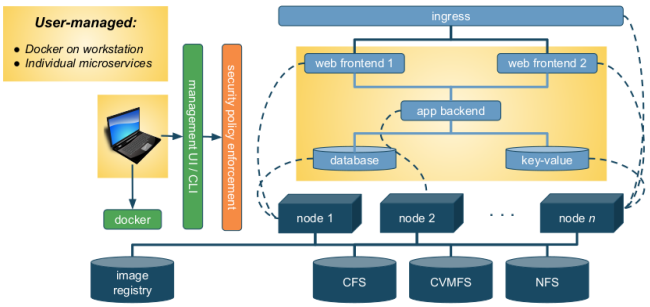

Fig. 5. Spin architecture depicting a typical edge service. The Docker-based workstation and service deployment are user-managed; all underlying infrastructure and HPC integration are managed by NERSC.

- *Overlay network*: an abstraction of the physical network, this virtual network isolates related groups of microservices into their own private, encrypted IP space. External routes are explicitly declared to allow outgoing access to HPC storage and job management and incoming connections to web or database services.
- *Worker nodes and local storage*: commodity servers running container workloads in Docker. Containers are distributed across different worker nodes by the scheduler to level resource usage. Enterprise storage is mounted for data persistence outside of the HPC environment.
- *Security and Policy Enforcement*: encoded policy declarations and real-time logic that perform user authentication, enforce authorization, and inspect user activity to permit or deny deployment operations. Spin uses the Open Policy Agent (OPA) [31] system for security policy enforcement.

B. Usage

Building and deploying services in Spin loosely follows the Docker *"Build - Ship - Run"* methodology:

- *Build*: images for individual microservices are built and tested on a local workstation using `docker build`, `run`, and `exec`;
- *Ship*: when complete, images are uploaded to the Registry with a specific name and version using `docker push`;
- *Run*: microservices are defined and deployed using the Rancher CLI or UI, where they can communicate with each other via the overlay network.

Microservices may be run (or *"deployed"*) using the Rancher CLI or UI. The CLI requires preparing YAML declaration files that describe the deployment; this mode of access lends itself well to version control. The UI is a more manual, interactive process suitable for new users, initial deployments from scratch, or proof-of-concept experimentation. The two approaches may also be used interchangeably.

C. Security Measures

The conventional Docker daemon runs with privilege, which is necessary for it to interact with worker node resources

(e.g. network interfaces and storage). Because direct access to Docker is equivalent to privileged access to the node, additional measures must be taken to secure a multi-tenant, shared infrastructure.

- *Access via Rancher CLI / UI*: all access to Docker is mediated through Rancher, which enforces authorization and security policy.
- *Multi-tenancy / RBAC*: users can see and manipulate only the services associated with their NERSC projects, giving them a virtual private view on the shared environment.
- *Limited allowable Linux capabilities*: containers may run only with a limited subset of capabilities, which prevents typical "root" privilege and subverts privilege escalation.
- *Dynamic policy enforcement*: the OPA system integrates with Kubernetes, inspecting user actions and applying rules to allow or deny execution, such as allowing mounts only of approved file system paths or preventing users from running containers as another's UID.

"Edge services" are a critical component of many complex, distributed workflows. Spin provides a supported platform for these services built on worker node clusters placed adjacent to the HPC environment, an overarching orchestration system, and a rule-based security policy enforcement subsystem.

V. Interactive Computing with Jupyter

Large scale Superfacility-type experimental science workflows require support for a unified, interactive, real-time platform that can manage a distributed set of resources connected to High Performance Computing (HPC) systems. The Jupyter platform [32] plays a key role in this space - it provides the ease-of-use and interactivity of a web science gateway while providing scientists the ability to build custom, ad-hoc workflows in a composable and sharable way. Jupyter Notebooks combine live executable code cells, with inline documentation and embedded interactive visualizations. This allows us to capture an experiment in a fully contained executable Notebook that is self-documenting and incorporates live rendering of outputs and results as they are generated. The Notebook format lends itself to a highly modular and composable workflow, where individual steps and parameters can be adjusted on the fly. Additionally, the Jupyter platform can support custom applications and extensions that live alongside the core Notebook interface.

As part of the LBNL Superfacility project we have made substantial improvements to our Jupyter service and demoed its use for HPC applications.

A. Scaling up Jupyter notebooks

Notebooks provide a concise way to capture a set of steps involved in an experimental analysis workflow. However, a typical notebook runs inside a single notebook "kernel" process. This means that if there are steps in the notebook that need to be scaled up, we need to run those on additional compute resources. In our work with experimental facilities like NCEM, ALS and LCLS we have implemented a process [33] for enabling this kind of scale-up using data parallel frameworks like Dask [34]. In this mode when the user starts a notebook we spawn a compute job that runs on backend compute nodes and starts up a set of Dask workers that can accept tasks. The notebook can then communicate with these workers to farm out the operations that need to be parallelized and scaled up, effectively giving us a mechanism to run a given piece of Python code at much higher levels of concurrency. We can also communicate with the running tasks to pull results, allowing us to interact with results in real-time.

B. Sharing and cloning

Modern experimental science is highly collaborative. Scientists want to provide curated analysis notebooks for their colleagues to run a set of predefined analyses that can then be modified, and applied to other datasets. Towards this end we have developed a service (based on the *nbviewer* [35] tool), that allows a project to browse a repository of curated notebooks, along with a predefined software environment at NERSC. Our service *CloneNotebooks* allows a user to browse this repository of notebooks along with a static HTML rendering of a given notebook file. If the user wishes to use this analysis they can then clone the notebook into their own Jupyter environment at NERSC, along with a reference to the pre-installed software environment, allowing the user to then have a live working copy of the analysis.

VI. Conclusion

The experimental and observational science community is increasingly looping in HPC facilities and high-performance networks to manage, move and analyse their data. The LBNL Superfacility project has been designed to coordinate and focus the technical and policy work required to support these workflows at ESnet and NERSC. This paper highlights key areas of technical innovation in automation, data management, edge services and interactivity with Jupyter that have been driven by the science needs of our user community. Our goal is to enable truly automated use of ESnet and NERSC for end-to-end user workflows. We are now mid-way through the project, and our focus for the future will be on increased automation, federated identity management, workflow resiliency and providing a seamless user experience for the tools we have deployed.

References

[1] The Dark Energy Spectroscopic Instrument, "DESI," https://www.desi.lbl.gov/.

[2] Rubin Observatory Legacy Survey of Space and Time Dark Energy Science Collaboration, "LSST-DESC," https://lsstdesc.org/.

[3] LUX-ZEPLIN Dark Matter Experiment, "LZ," https://lz.lbl.gov/.

[4] Advanced Light Source, "ALS," https://als.lbl.gov/.

[5] Linac Coherent Light Source, "LCLS," https://lcls.slac.stanford.edu/.

[6] National Center for Electron Microscopy, "NCEM," https://foundry.lbl.gov/about/facilities/the-national-center-for-electron-microscopy-ncem/.

[7] S. Cholia, D. Skinner, and J. Boverhof, "Newt: A restful service for building high performance computing web applications," in *2010 Gateway Computing Environments Workshop (GCE)*, 2010, pp. 1–11.

[8] Json web tokens. [Online]. Available: https://jwt.io

[9] Restx, formerly restplus. [Online]. Available: https://github.com/python-restx/flask-restx

[10] Swagger. [Online]. Available: https://swagger.io/

[11] Oauth v2. [Online]. Available: https://oauth.net/2/

[12] Management and reporting for your account, compute and storage allocations, and projects at NERSC. [Online]. Available: https://iris.nersc.gov

[13] E. Lingerfelt, A. Belianinov, E. Endeve, O. Ovchinnikov, S. Somnath, J. Borreguero, N. Grodowitz, B. Park, R. Archibald, C. Symons, S. Kalinin, O. Messer, M. Shankar, and S. Jesse, "Beam: A computational workflow system for managing and modeling material characterization data in hpc environments," *Procedia Computer Science*, vol. 80, pp. 2276 – 2280, 2016, international Conference on Computational Science 2016, ICCS 2016, 6-8 June 2016, San Diego, California, USA. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050916308869

[14] M. A. Salim, T. D. Uram, J. T. Childers, P. Balaprakash, V. Vishwanath, and M. E. Papka, "Balsam: Automated scheduling and execution of dynamic, data-intensive hpc workflows," 2019.

[15] J. T. Childers, T. D. Uram, D. Benjamin, T. J. LeCompte, and M. E. Papka, "An edge service for managing hpc workflows," in *Proceedings of the Fourth International Workshop on HPC User Support Tools*, ser. HUST'17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3152493.3152557

[16] Openid connect. [Online]. Available: https://openid.net/connect/

[17] Lustre home page. [Online]. Available: https://lustre.org/

[18] F. Schmuck and R. Haskin, "GPFS: A shared-disk file system for large computing clusters," in *In Proceedings of the 2002 Conference on File and Storage Technologies (FAST*, 2002, pp. 231–244.

[19] Hpss collaboration. [Online]. Available: http://www.hpss-collaboration.org/

[20] W. Bhimji, D. Bard, K. Burleigh, C. Daley, S. Farrell, M. Fasel, B. Friesen, L. Gerhardt, J. Liu, P. Nugent, D. Paul, J. Porter, and V. Tsulaia, "Extreme I/O on HPC for HEP using the burst buffer at NERSC," *Journal of Physics: Conference Series*, vol. 898, p. 082015, oct 2017. [Online]. Available: https://doi.org/10.1088/1742-6596/898/8/082015

[21] R. Gerber, J. Hack, K. Riley, K. Antypas, R. Coffey, E. Dart, T. Straatsma, J. Wells, D. Bard, S. Dosanjh, I. Monga, M. E. Papka, and L. Rotman, "Crosscut report: Exascale requirements reviews, march 9–10, 2017 – tysons corner, virginia. an office of science review sponsored by: Advanced scientific computing research, basic energy sciences, biological and environmental research, fusion energy sciences, high energy physics, nuclear physics," 1 2018.

[22] D. Stansberry, S. Somnath, J. Breet, G. Shutt, and M. Shankar, "Datafed: Towards reproducible research via federated data management," in *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2019, pp. 1312–1317.

[23] I. Foster, "Globus online: Accelerating and democratizing science through cloud-based services," *IEEE Internet Computing*, vol. 15, no. 3, pp. 70–73, 2011.

[24] The HDF Group, "HDF5," https://www.hdfgroup.org/solutions/hdf5/.

[25] S. Byna, M. S. Breitenfeld, B. Dong, Q. Koziol, E. Pourmal, D. Robinson, J. Soumagne, H. Tang, V. Vishwanath, and R. Warren, "ExaHDF5: Delivering Efficient Parallel I/O on Exascale Computing Systems," *Journal of Computer Science and Technology*, vol. 35, no. 1, pp. 145–160, 2020. [Online]. Available: https://doi.org/10.1007/s11390-020-9822-9

[26] W. Zhang, S. Byna, H. Tang, B. Williams, and Y. Chen, "MIQS: Metadata Indexing and Querying Service for Self-Describing File Formats," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3295500.3356146

[27] Spin. [Online]. Available: https://www.nersc.gov/systems/spin/

[28] Docker. [Online]. Available: https://www.docker.com

[29] Rancher. [Online]. Available: https://rancher.com

[30] Kubernetes. [Online]. Available: https://kubernetes.io

[31] Open policy agent. [Online]. Available: https://www.openpolicyagent.org

[32] T. Kluyver, B. Ragan-Kelley, F. Perez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, C. Willing, and Jupyter Development Team, "Jupyter notebooks – a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, 01 2016, pp. 87 – 90.

[33] M. L. Henderson, W. Krinsman, S. Cholia, R. Thomas, and T. Slaton, "Accelerating experimental science using jupyter and nersc hpc," in *Tools and Techniques for High Performance Computing*. Springer, 2019, pp. 145–163.

[34] Dask Development Team, *Dask: Library for dynamic task scheduling*, 2016. [Online]. Available: https://dask.org

[35] Nbviewer. [Online]. Available: https://nbviewer.jupyter.org/