NERSC Superfacility API

Credits to:
LBNL Superfacility Team:
D. Bard, C. Snavely, G. Torok, R. Thomas, A. Greiner, etc.
NCEM: P. Ercius, C. Harris (Kitware)

Bjoern Enders
Data Science Workflows Architect
NERSC/LBNL
NERSC Data Day, Oct 26, 2022
NERSC supports a large number of users and projects from DOE SC’s experimental and observational facilities.

**Superfacility:**
Ecosystem of connected facilities, software and expertise to enable new modes of discovery.

**Superfacility API:**
An API into NERSC to embed HPC into cross-facility workflows. It is also a general purpose API for all NERSC users and projects.

~35% (235) of NERSC projects self-identified as confirming the primary role of the project is to 1) analyze experimental data or; 2) create tools for experimental data analysis or; 3) combine experimental data with simulations and modeling.
Model case

Experiments at ext. facilities use high frame rate 2D detectors for their science.

Hosting data & compute on site has become increasingly demanding.

Requirements

- Planning (HPC as reliable partner)
  - machine-readable status
- Resiliency (needs failover)
  - compatible interfaces
- Realtime (can't wait in queue)
  - workflow endpoint
- Services (portals, data, db)

1. Plan / Check availability of NERSC resource for experiment.
   - check status / accounts
2. Get raw data to NERSC, when experiment is live.
   - move data
3. Start analysis job quasi synchronous with data
   - submit job / monitor job
4. Gather feedback, ideally immediate.
   - download / execute command
5. Move data and results to archive after analysis.
   - move data
The API
Why an API?

- Meets a critical need; automation is no longer optional
  - Unattended operation; minimizing HITL
  - Track/submit large number of jobs
  - Interface with collaborations, workflows and machines

- NERSC becomes “machine readable”
  - Enables easier creation of UIs, portals, etc.
  - Allows integration with control/analysis software
  - “NERSC inside™”

- Less DIY: simpler, standardized tooling (Python, etc)
  - Stable refactor target for established projects or easier on-ramp for new ones
  - Contribute to HPC interface standards for portability
  - Authentication and security models

Drivers:
- Complex workflows
- Data-driven projects
- Real-time compute and streaming data from instruments
- Automation
What specifically can the API do?

**Vision:** all NERSC interactions are callable; backend tools assist large or complex operations.

**Endpoints prototyped or in prep:**

- `/status` query the status of NERSC component system health
- `/account` data about the user’s projects, roles, groups and usage information.
- `/compute` run batch jobs, query job and queue statuses on compute resources.
- `/storage` move data with Globus or between NERSC storage tiers
- `/tasks` get info about asynchronous tasks (eg. from `/compute` or `/storage`).
- `/utilities` traverse the filesystem, upload and download small files, and execute commands on NERSC systems
- `/reservations` submit and manage future compute reservations (coming soon)
<table>
<thead>
<tr>
<th>Action</th>
<th>Manual steps</th>
<th>With SuperFacility API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check status</td>
<td>Test SSH or ping specific services for status</td>
<td>Query the <code>/status</code> API endpoint if resources are active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Submit job</td>
<td>SSH in and submit jobs with <code>sbatch</code> ...</td>
<td>Create jobs using POST calls from a script or Spin service to the <code>/compute</code> endpoint.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monitor job</td>
<td>SSH in (again) and do `squeue</td>
<td>grep</td>
</tr>
<tr>
<td>Plan ahead</td>
<td>Read the NERSC MOTD to see if any down time is planned</td>
<td>Query the <code>/status/outages/planned</code> API endpoint for planned outages</td>
</tr>
<tr>
<td>Move data</td>
<td>SSH in and run file transfer tools to move data</td>
<td>POST to the <code>/storage</code> API endpoint.</td>
</tr>
<tr>
<td>Check account</td>
<td>Log into &quot;Iris&quot; (our accounting web app) and check allocation account balance.</td>
<td>Query the <code>/account</code> API to get the same information.</td>
</tr>
</tbody>
</table>
Model use

Experiments at ext. facilities use high frame rate 2D detectors for their science.

Hosting data & compute on site has become increasingly demanding.

Requirements

- Planning (HPC as reliable partner)
  - machine-readable status
- Resiliency (needs failover)
  - compatible interfaces
- Realtime (can't wait in queue)
  - workflow endpoint
- Services (portals, data, db)

1. Plan / Check availability of NERSC resource for experiment.
   - /status (/reservations)
2. Get data to NERSC, when experiment is live.
   - /storage
3. Start analysis job quasi synchronous with data
   - /compute /tasks
4. Gather feedback, ideally immediate.
   - /utilities /storage
5. Move data and results to archive after analysis.
   - /storage
A science example
The MF provides state-of-the-art expertise, methods, and instrumentation in nanoscale science in a safe environment free of charge.

NCEM is one of 7 facilities in the MF (about ~1/3 of total proposals and staff).

Staff Scientists work in a 50/50 model: 50% of their time is spent on user research and 50% of their time is spent on internal research. User research is often highly collaborative.

We are leaders in:
- high resolution
- tomography
- in situ
- soft materials
- 4DSTEM
- image simulation
- electron detector technology

National Center for Electron Microscopy ...

... uses NERSC to process large data sets live during experiments

- 87,000 Hz (480 Gbit/s) readout (typical STEM scanning rates)
- 1kx1k scan is 650 GB captured in 15 seconds
- Data pipeline: FPGA → RAM → Flash storage → Sparse HDF5
NCEM - Distiller app

credits: Chris Harris @ Kitware, Peter Ercius @ NCEM
<table>
<thead>
<tr>
<th>ID</th>
<th>Scan ID</th>
<th>Notes</th>
<th>Location</th>
<th>Created</th>
<th>Progress</th>
</tr>
</thead>
<tbody>
<tr>
<td>81</td>
<td>14</td>
<td>STEM-H same lattice</td>
<td>129.55.132.192</td>
<td>2021-11-04 17:13:04 000000</td>
<td>✔️</td>
</tr>
<tr>
<td>80</td>
<td>13</td>
<td>STEM-H better alignment</td>
<td>129.55.132.192</td>
<td>2021-11-04 17:19:30 008989</td>
<td>✔️</td>
</tr>
<tr>
<td>79</td>
<td>12</td>
<td>STEM-H 512x512 100p Au on edge</td>
<td>129.55.132.192</td>
<td>2021-11-04 17:41:54 938136</td>
<td>✔️</td>
</tr>
<tr>
<td>78</td>
<td>11</td>
<td>512x512 80μm CL</td>
<td>129.55.132.192</td>
<td>2021-11-04 17:43:59 729923</td>
<td>✔️</td>
</tr>
<tr>
<td>77</td>
<td>10</td>
<td>512x512 Au</td>
<td>129.55.133.107</td>
<td>2021-11-04 17:55:54 766514</td>
<td>✔️</td>
</tr>
<tr>
<td>76</td>
<td>9</td>
<td>512x512x4 Au</td>
<td>129.55.132.192</td>
<td>2021-11-03 17:16:19 325501</td>
<td>✔️</td>
</tr>
<tr>
<td>75</td>
<td>8</td>
<td>512x512x4 Au</td>
<td>129.55.133.107</td>
<td>2021-11-03 17:15:54 417666</td>
<td>✔️</td>
</tr>
<tr>
<td>74</td>
<td>7</td>
<td>128x128 Au</td>
<td>129.55.132.192</td>
<td>2021-11-03 17:15:54 417666</td>
<td>✔️</td>
</tr>
<tr>
<td>73</td>
<td>5</td>
<td>128x128 Au</td>
<td>129.55.132.192</td>
<td>2021-11-03 17:15:54 417666</td>
<td>✔️</td>
</tr>
<tr>
<td>72</td>
<td>308</td>
<td>MgO single atom</td>
<td>129.55.132.192</td>
<td>2021-10-27 14:03:03 322885</td>
<td>✔️</td>
</tr>
<tr>
<td>71</td>
<td>307</td>
<td>MgO single atom</td>
<td>129.55.132.192</td>
<td>2021-10-27 13:33:56 469850</td>
<td>✔️</td>
</tr>
<tr>
<td>70</td>
<td>306</td>
<td>MgO single atom</td>
<td>129.55.132.192</td>
<td>2021-10-27 13:33:56 469850</td>
<td>✔️</td>
</tr>
<tr>
<td>50</td>
<td>288</td>
<td>MgO single atom 1x1x1c</td>
<td>129.55.132.192</td>
<td>2021-10-27 13:33:56 469850</td>
<td>✔️</td>
</tr>
<tr>
<td>51</td>
<td>287</td>
<td>MgO single atom 1x1x1c</td>
<td>129.55.132.192</td>
<td>2021-10-27 13:33:56 469850</td>
<td>✔️</td>
</tr>
<tr>
<td>41</td>
<td>390</td>
<td>PtZn SP</td>
<td>129.55.132.192</td>
<td>2021-10-19 15:41:48 227279</td>
<td>✔️</td>
</tr>
</tbody>
</table>
How to use the API
Superfacility API Basics

https://api.nersc.gov/api/v1.2

• A unified programmatic approach to accessing NERSC
• REST API with json input/output
• Standards-based authentication
• End user docs and examples: https://docs.nersc.gov/services/sfapi/

Swagger documentation:
• Interactive, up-to-date and self-documenting
• See endpoints, payloads, example code
• Works with any dev environment

~ 4M logged requests since May, = one request every 3 sec
How to get your hands on the API

As a user:

- The /status endpoints are all public.
- Every NERSC user can get API access to non-public endpoints via Iris [https://iris.nersc.gov](https://iris.nersc.gov)
  - Profile -> scroll down to "Superfacility API Clients" tab
  - R/W clients require filling out a form
- Getting started documentation available at [https://docs.nersc.gov/services/sfapi/](https://docs.nersc.gov/services/sfapi/)

As an HPC facility:

- Please get in touch with us if you have question on how to build an API at your facility.
  - benders@lbl.gov (Bjoern Enders)
  - djbard@lbl.gov (Debbie Bard)
Example use

Experiments at ext. facilities use high frame rate 2D detectors for their science.

Hosting data & compute on site has become increasingly demanding.

Requirements

- Planning (HPC as reliable partner)
  - machine-readable status
- Resiliency (needs failover)
  - compatible interfaces
- Realtime (can't wait in queue)
  - workflow endpoint
- Services (portals, data, db)

1. Plan / Check availability of NERSC resource for experiment.
   - /status (/reservations)
2. Get data to NERSC, when experiment is live.
   - /storage
3. Start analysis job quasi synchronous with data
   - /compute /tasks
4. Gather feedback, ideally immediate.
   - /utilities /storage
5. Move data and results to archive after analysis.
   - /storage
Check queue status, wait for job to complete.

```python
print(json.dumps(sacct['stdout'], indent=2))
```

Read from the slurm output file

```python
slurmfile = home + '/apidemo/slurm-{:id}.out'
response = api("utilities/command/cori", {"executable": "tail -n 1 slurmfile"}, as_form=True)
if isinstance(response, Task):
    print(response.wait_for_result()["output"].strip())
```

Extract the file

We could just copy paste from above or use this one-liner to search the output for the saved file.

```bash
for line in open(slurmfile, 'r').readlines():
    if line.startswith('Session'): print(line.strip())
```
Check queue status, wait for job to complete.

```python
print(json.dumps(api('compute/jobs/perlmutter/\{jobid\}?sacct=true'), indent=2))
print(api('compute/jobs/perlmutter/\{jobid\}?sacct=true')['output']['\{state\}'])
```

Read from the slurm output file

```python
slurmfile = home\{/apidemo\}/slurm-\{jobid\}.out
response = api('utilities/command/perlmutter', (\{executable\}:\"tail -n 20 \{slurmfile\}\})
if isinstance(response, Task):
    print(response.wait_for_result()['\{output\}'].strip())
```

Poll 03:('id': '2237', 'status': 'completed', 'result': {'status': 'ok', 'output': ''}, 'subpix': linear, 'update_object_first': True, 'overlap_converge...': 0.05\n * overlap_max_iter...: 10\n * probe_inertia : 1e-09\n * object_inertia : 0.0001\n * fourier_power_bound : None\n * fourier_relax_factor : 0.05\n * obj_smooth_std : None\n * clip_object : None\n * probe_center_tol : None\n * compute_log_like...: True\n * probe_update_cuda... : False\n * obj_update_cud...: True\n * fft_lib : reikna\n * alpha : 1.0\n * name : DM_pycuda

Iteration #10 of DM_pycuda :: Time 1.249\nErrors :: Fourier 5.94e+01, Photons 4.73e+01, Exit 3.51e+01\n, "error": null\}

{'status': 'ok', 'output': ''}, 'subpix': linear, 'update_object_first': True, 'overlap_converge...': 0.05\n * overlap_max_iter...: 10\n * probe_inertia : 1e-09\n * object_inertia : 0.0001\n * fourier_power_bound : None\n * fourier_relax_factor : 0.05\n * obj_smooth_std : None\n * clip_object : None\n * probe_center_tol : None\n * compute_log_like...: True\n * probe_update_cuda... : False\n * obj_update_cud...: True\n * fft_lib : reikna\n * alpha : 1.0\n * name : DM_pycuda

Iteration #10 of DM_pycuda :: Time 1.249\nErrors :: Fourier 5.94e+01, Photons 4.73e+01, Exit 3.51e+01\n, "error": null\n
Roadmap
Roadmap

• Clients and Tokens with more granular scope (~ weeks)
  o new client interface (draft see right)
  o more source IP ranges per client
  o short-lived full featured clients without manual review

• SF API to retire NEWT (~ months)
  o login-based route to get tokens for mynersc, science gateway apps or other web apps.

• Common API interface (~ year)
  o a set of endpoints/methods that work with many facilities
  o we're talking with CSCS (firecrest API), HPCS@LBL, S3DF@SLAC, OLCF
Outreach

- High-level overviews of the API have been given at workshops and meetings oriented toward software development, such as the DOE Workflow Workshop and Hack-a-thon and NERSC GPUs for Science Day, both in 2019.

- A proof-of-concept demonstrations with Jupyter notebooks were shown at the DOE exhibition booth at SC’19 and SC’21 (the latter already with Perlmutter) (https://scdoe.info/demonstrations/)

- A detailed presentation of the API architecture and usage coupled with a Jupyter-based demonstration was given at the Superfacility Project Demo Series in 2020. https://www.youtube.com/watch?v=dmbBJmMUERU&list=PL20S5EeApOSsv6RVG6m0l6tx2wMp2T4PP&index=3

- A paper was published with the ISC’21 SuperCompCloud workshop and accompanied by a presentation “Automation for Data-Driven Research with the NERSC Superfacility API”

- Science examples of earlier adopters were presented at a SC’21 BoF about HPC APIs.
  - Building an HPC API community.

- We're in touch with OLCF to adapt a similar API for their facility