How Jupyterhub Tamed Big Science

Tales From a Supercomputing Center

Shreyas Cholia, Rollin Thomas, and Shane Canon

August 24 2017
NERSC (at Lawrence Berkeley Lab) is the production HPC & Data Facility for the Department of Energy Office of Science
Cori: Friendly for “Data Users”

- Two architectures in one system:
  - Data 2388 nodes
    - 32-core Intel Xeon “Haswell”
    - 128 GB DDR4
  - HPC 9688 nodes
    - 68-core Intel Xeon Phi “KNL”
    - 96 GB DDR4 + 16 GB MCDRAM
  - Haswell login and **special-purpose large memory nodes** (512 & 768 GB)
  - NVRAM Burst Buffer for IO acceleration
  - Shared and real-time queues
  - Shifter for containerized HPC

Gerty Cori: Biochemist and first American woman to win a Nobel Prize in science
Data Science [Wikipedia Definition]

- Get manageable chunk of data and copy it to your laptop/workstation
- Write code/scripts, make diagnostic plots, construct and test models
- Loop is very short between thinking up a query and executing it on data
  - **Real-time testing of models that explain the data**
  - **Real-time feedback in the form of plots and results**
  - ... hard to keep it all organized and explain what you did

Diagram courtesy of “Farcaster” at English Wikipedia
Enter Jupyter

- **Jupyter Notebooks: Literate Computing, “Narratives”**
  - Code and comments: Reproducibility, show your work! Document your workflow
  - Rich text, plots, equations, widgets, etc.
  - Iterate and explore to arrive at meaningful insights

Data Science Process

- Raw Data Collected
- Data Is Processed
- Clean Dataset
- Models & Algorithms
- Exploratory Data Analysis
- Reality
- Data Product
- Communicate Visualize Report
- Make Decisions

Diagram courtesy of “Farcaster” at English Wikipedia
Old School Supercomputing

- Scientific insight => interactive, iterative exploration and analysis
- But HPC is still very much a command-line and batch job driven enterprise
  - User log in via SSH, compile their codes
  - Asynchronous jobs via batch scripts
  - Not conducive to the interactive discovery loop
  - Not ideal for usability, visualization, analysis as far as the average scientist is concerned

How can Jupyter bridge this gap?
Why Jupyter@NERSC?

Deep Questions → Expensive Detector Technologies → Insightful Real time predictions?
Instruments/Facilities
High-bandwidth Networks
Simulations
Exploratory analysis?
Decision making?
Expose, Integrate NERSC Resources

submit, monitor, interact

query, analyze, visualize

standardize, reproduce results

Batch Queues
sbatch squeue srun sacct

NERSC Global File System
/project $SCRATCH $HOME

Database Servers
mongodb01... scidb1...

Software Environment Modules
python/2.7-anaconda python/3.5-anaconda
Python is the most popular language at NERSC used to:

- Script workflows for both data analysis and simulations
- Perform exploratory data analysis
Motivation For Jupyterhub Service

❌ Users running their own notebook servers on a supercomputer makes security folks very nervous.

❌ Difficult to support and manage different kernels and environments

Jupyterhub to rescue

✓ Centralized service to deploy notebooks in a standard authenticated manner

✓ Package known kernels out of the box (Anaconda)

✓ Access to NERSC resources through this interfaces
  • Filesystems, Batch Queue, Network, DBs
Jupyterhub: Jupyter as a Service

- Service to deploy notebooks in a multi-user environment
- Manages user authentication, notebook deployment and web proxies
Jupyter@NERSC
Evolution of Architecture

Step 1: Give people access to their data
First Architecture: “Edge Service”

August 2015:
- Single **Docker** container with access to NERSC Global File System
- Very popular service: 100+ users
- Missing:
  - Access to Cori Lustre Scratch
  - Interactivity with Cori batch queues
  - Cori Python environment.

Projects:
- OpenMSI
- Metabolite Atlas
- LUX
...
Jupyter@NERSC
Evolution of Architecture

Step 2: Integration with Cori compute and filesystems
Second Architecture: Cori Login Node

August 2016:
- Standalone Hub server in Docker
- SSH spawner spins up notebook on special-purpose Cori login node
- Access to Cori Lustre Scratch
- Same Python environment as Cori login
- Interactivity with batch queues

Projects:
- LSST
- DESI
- MaterialsProject
...
Our Extensions to JupyterHub

**jupyterhub.auth.Authenticator**

- Use MyProxy to login to NERSC CA server with user/pass to get X509 certificate credentials.
- No need to run JupyterHub with additional privileges, or root access.

**GSIAuthenticator**

https://github.com/NERSC/GSIAuthenticator

**jupyterhub.spawner.Spawner**

- SSH to Cori with user’s credential. Uses GSISSH, but can use SSH.
- Notebook starts up, spawner goes away, Notebook communicates w/Hub, keep PID.

**SSHSpawner**

https://github.com/NERSC/sshspawner
GSI Authenticator

- User logs in with username and password. Authenticator uses myproxy to login to NERSC CA server with username/password and retrieves credentials (X509 certificate)
- Jupyterhub runs as a standalone service and doesn’t need root access. In fact, no root access needed across this architecture.
- [https://github.com/NERSC/gsiauthenticator](https://github.com/NERSC/gsiauthenticator)
SSH Spawner

- We wrote an SSH Spawner that will SSH into the Cori node with users credential
  - Supports GSISSH (use with certificates from GSI authenticator)
  - Supports SSH key based auth
- SSH Spawner starts up notebook server process and goes away; Notebook server communicates directly with hub
  - No tunnels or persistent connections needed
- Keep track of the PID for poll and shutdown functions (also via SSH)
- Inspired by Andrea Zonca’s RemoteSpawner (SDSC)
- [https://github.com/NERSC/SSHSawner](https://github.com/NERSC/SSHSawner)
SLURM MAGIC

- Jupyter “%magic” commands:
  - Expose extra-language functionality
  - Outputs are first-class Notebook objects

- Developed wrappers around SLURM commands. [https://github.com/NERSC/slurm-magic](https://github.com/NERSC/slurm-magic)

- `%squeue`
  `%squeue -u rthomas`

- `%sbatch`
  `%sbatch script.sh`

- `%%sbatch`
  `%%sbatch -N 1 -p debug -t 30 -C haswell
  #!/bin/bash
  srun ...`
Enable Custom Kernels

- Users customize their notebooks with libraries and APIs of their own design or from third parties.
- NERSC wants to offer Jupyter to users so they don’t set it up themselves in an insecure way.

```
{
    "display_name": "HEP",
    "language": "python",
    "argv": ["/global/common/cori/software/python/2.7-anaconda/bin/python", "-m", "IPython.kernel", "-f", "{connection_file}"
    ],
    "env": {
        "LD_LIBRARY_PATH": "/usr/common/software/root/6.06.06/lib/root",
        "PYTHONPATH": "/usr/common/software/root/6.06.06/lib/root"
    }
}
```

Example PyROOT Kernel Spec
LIVE DEMO: What Could Go Wrong!?
Jupyter@NERSC
Evolution of Architecture

Step 3: The Future
Next: Cori Compute Nodes

```bash
--qos=interactive
```

- Web Browser
- JupyterHub
- Web Server
- Cori Login Node
  - Notebook Server Process
  - Kernel Process
- Cori Compute Node
  - Notebook Server Process
  - Kernel Process
- Cori Compute Node
  - Notebook Server Process
  - Kernel Process
- Cori Compute Node
  - Kernel Process

The diagram illustrates the components and processes involved in accessing and using Cori Compute Nodes, including the login node and compute nodes, with a focus on the `--qos=interactive` option.
Role of SDN after Authentication

Web Browser

Cori Login Node
- Notebook Server Process
- Kernel Process

Cori Compute Node
- Notebook Server Process
- Kernel Process

Cori Compute Node
- Notebook Server Process

Cori Compute Node
- Kernel Process
Project Kale is a research effort focused on adapting the Jupyter machinery for HPC workflows.

- Master notebook to control workflow
- Jupyter notebooks as **interactive workflow steps**
- Interaction with workflow tasks via kernels
- Realtime Monitoring of HPC jobs and output
- Widgets and dashboards for batch job management
The Ultimate Jupyter@NERSC

Software defined networking

Advertise IP of notebook server back to user.
Notebook on login node, kernel on compute.
Notebook+kernel on login, Spark job on computes.

Leveraging interactive QOS

Immediate access to compute up to four hours.

Docker/Shifter

Customize notebook/kernel’s environment through containers.
Make larger-scale analytics apps actually start up.

Other possibilities

Notebook/scheduler on Haswell, kernels on KNL?
Customizations to Jupyter

**Spawner**
- **BatchSpawnerBase**
  - **BatchSpawnerRegexStates**
    - **UserEnvMixin**
    - **SlurmSpawner**
- **WrapSpawner**
  - “NERSCSpawner”

**Customize Access**
- Burst buffer for your job?
- Cori node or compute?

**Customize NERSC UX**
- “My Docker images”
- “My favorite job templates”
- …

https://github.com/jupyterhub/batchspawner
https://github.com/jupyterhub/wrapspawner
Acknowledgements

Big Thanks to the Community!

- MSI
- TACC
- SDSC
- Jupyter Dev Team
Conclusion I

- Jupyter is a powerful tool for exploratory data analysis that is increasingly popular with NERSC users.
- We anticipate that more users will be asking for tools like Jupyter, and for the data sets they analyze to be getting larger, requiring multi-node “Jupyter jobs.”
- We are working to find ways to scale Jupyter up to handle bigger data sets and interoperate with NERSC resources and environment.
High-level takeaway for JupyterCon:

Most of what we have done/are doing is customizing or extending existing interfaces for the NERSC ecosystem:
   Spawner (SSHSpawner)
   Authenticator (GSIAuthenticator)
   BatchSpawner/WrapSpawner

External to that:
   Developing simple helper scripts
   Networking configuration changes
   API for a supercomputing center?! (NEWT)
“I’ll never have to leave a notebook again … that’s like the ultimate dream”
National Energy Research Scientific Computing Center