Integrating Interactive Jupyter Notebooks at the BNL SDCC

D. Allan, D. Benjamin*, M. Karasawa, K. Li, O. Rind, W. Strecker-Kellogg
Brookhaven National Laboratory, *Argonne National Laboratory
BNL Scientific Data & Computing Center (SDCC)

• Located at Brookhaven National Laboratory on Long Island, NY — Largest component of the Computational Science Initiative (CSI)

• Serves an increasingly diverse, multi-disciplinary user community: RHIC Tier-0, US ATLAS Tier-1 and Tier-3, Belle-II Tier-1, Neutrino, Astro, LQCD, NSLS-II, CFN, sPHENIX….more than 2000 users from 20+ projects

• Large HTC infrastructure accessed via HTCondor (plus experiment-specific job management layers)

• Growing HPC infrastructure, currently with two production clusters accessed via Slurm

• Limited interactive resources accessed via ssh gateways
Two modes, Two workflows

- **HPC & HTC** (parallel vs interlinked, accelerator vs plain-cpu)
  - High-performance systems for GPUs / MPI / accelerators
  - High-throughput systems for big data parallel processing
- **Batch & Interactive** (working on code/GPUs vs submitting large workflows)
  - Job workflow management
  - Direct development & testing on better hardware

Traditional “Interactive SSH + Batch” paradigm places requirements on the users:

- Must be sufficiently motivated to learn and use batch systems
- Need to buy in to the workflow model: Develop, compile, move data, small-scale run on interactive nodes, full-scale processing on batch
Data Analysis As A Service

• New paradigm: **Jupyter Notebooks** (IPython)
  ‣ Expanding the interactive toolset
  ‣ “Literate Computing”: Combines code, text, equations within a narrative
  ‣ Easy to document, share, and reproduce results; create tutorials…Lower barrier of entry, both for learning curve and user-base
  ‣ Provides a flexible, standardized, platform independent interface through a web browser
  ‣ Can run with no local software installation
  ‣ Many language extensions (kernels) and tools available
Jupyter Service UI

Kernels

Notebook Documents

Jupyterlab
Production Architecture

• **Goal**: leverage already successful pre-existing resources, expertise, and infrastructure (batch) instead of rolling a new backend service
  ▶ Allow users to leverage any type of computational resource they might need — implies enabling both HTC and HPC/GPU, e.g. upcoming ATLAS ML workflows

• **Requirements**
  ▶ Expose to the world via unified interface [https://jupyter.sdcc.bnl.gov](https://jupyter.sdcc.bnl.gov) — common solution for HTC and HPC resource access
  ▶ Satisfy cybersecurity constraints

• **Design**
  ▶ Insert authenticating proxy as frontend to decouple jupyterhub from cybersecurity requirements (e.g. MFA)
  ▶ Scale notebooks via load-balancing as well as via batch systems
    - Automated deployment of multiple hub instances using Puppet
  ▶ Enable access to GPU nodes in a user-friendly way
    • User-specific UI for Slurm spawner support
Jupyterhub Service Architecture
Frontend Proxy Interface

- For Orchestration: a small cluster of directly-launched jupyter instances
  - HTTP-level Load-balanced from frontend proxy
  - One each on IC and HTCondor shared pool
- For Develop and Test: Use existing batch systems
  - HTCondor and Slurm support running a jupyterlab session as a batch job
  - Containers can enter at batch level to isolate external users or can be based on choice of environment
  - Best way to ensure exclusive, fair access to scarce resources (e.g. GPUs)
- Open questions: Latency, Cleanup, Starvation
Multifactor Auth

• Using Keycloak MFA tokens
• Google Authenticator or FreeOTP app
• Easy setup by scanning QR code first time
* For form spawner code see https://github.com/fubarwrangler/sdcc_jupyter
Challenges of Experiment Environments

- When you get a session (start a notebook-server), which environment?
  - Customization at the kernel level or via notebook-server container
- Whose problem is setting up the environments?
  - Work for a software librarian

Kernel Customization

Custom Container
Example: sPHENIX Test Beam

** Notebook analysis courtesy of Jin Huang using custom sPHENIX Root Kernel**
Orchestration: Integrating Jupyter with Compute

• How to make it easier to use compute from Jupyter?
  ▶ HTMap library from condor
  ▶ Dask / IPyParallel / Parsl etc...

• Goal: abstract away the fact that you are using a batch system at all
  ▶ Either through trivial substitutes
    - map()→htmap()
  ▶ Or through cell "magics"
    - %slurm or equivalent
  ▶ Or via nice pythonic decorators that submit to batch systems (e.g. Dask-jobqueue)

```
1 from condormap import condormap
2 import collections
3 import numpy
4
5 # Sample function
6 def logistic(r, len=10):
7     d = collections.deque(maxlen=len)
8     x = 0.4
9     for _ in xrange(5 * 10**7):
10        x = x * r * (1.0 - x)
11        d.append(x)
12     return list(d)
13
14 for k, d in condormap(logistic, numpy.arange(1.5, 3.6, 0.01), withdata=True):
15    print sorted(d)
16    t = set(round(x, 5) for x in d)
17    print k, "Mode ", len(t)
```
Notebook Sharing: Short Term

- Low-effort, short-term sharing between users on the same Hub
- Sender creates shareable link that provides last saved version of notebook to link recipient
  - Short-term link expires after certain time
  - Link encodes notebook options, such as container, to ensure compatible software environment
- See https://github.com/danielballan/jupyterhub-share-link

* Courtesy Daniel Allan, illustrative gif: https://github.com/danielballan/jupyterhub-share-link/blob/master/demo.gif?raw=true
Notebook Archiving/Sharing

- Prepare a gallery of notebooks on a local Binder deployment, with a carefully defined software environment that anyone can recreate from a git repo with standard environment specs (e.g. requirements.txt)
  1. Enter URL of the repo
  2. Clicking "launch"
  3. Waiting and watching the build logs
  4. Copy a special link that will route directly to a Jupyter notebook running in a container that has repo contents and all software needed to run it successfully.
- Easy way for people to try your code and get running immediately
- Tightly coupled to Kubernetes and Docker, but developing similar workflows on HPC using Singularity

* Courtesy Daniel Allan
Conclusions

- The SDCC at BNL is deploying a Jupyterhub infrastructure enabling scientists from multiple disciplines to access our diverse HTC and HPC computing resources
- System designed to meet facility requirements with minimal impact on the backend
- Built-in support for experiment-based computing environment with a number of flexible access modes and workflows
- Continues to develop new techniques for user collaboration
HTTP Frontend Configuration

- Authentication via Mellon plugin (for Keycloak)
- Subdivide URL space for different hub servers
  - /jupyterhub/$cluster for HTC/HPC/others
- Load-balancing configuration
  - Need cookie for sticky-sessions
  - Newest apache on RHEL7
- Requires websockets support