Jupyter at NERSC
Redefining the Interface to HPC

Rollin Thomas
Data and Analytics Services

NERSC User Group Meeting
Rockville MD • 2019-07-19
What is Jupyter?

Tool for reproducible, shareable narratives, literate computing: 
*Notebook*: Document containing code, comments, outputs. Rich text, interactive plots, equations, widgets, etc.

Goal: Enable exploratory data analytics, deep learning, workflows, and more through Jupyter on NERSC systems.
Why Jupyter, Why Now at NERSC?

Integral part of Big (Data) Science & Superfacility:
LSST-DESC, DESI, ALS, LCLS, Materials Project
NCEM, LUX/LZ, KBase...

Generational shift in analytics for science + more:
UCB’s Data Science 8 course, entirely in Jupyter
“I’ll send you a copy of my notebook”
Training events adopting notebooks (DL)

Supporting reproducibility and science outreach:
Open source code and open source science
Jupyter notebooks alongside publications (LIGO)

2017 ACM Software System Award: “…a de facto standard for data analysis in research, education, journalism and industry. Jupyter has broad impact across domains and use cases. Today more than 2,000,000 Jupyter notebooks are on GitHub, each a distinct instance of a Jupyter application—covering a range of uses from technical documentation to course materials, books and academic publications.”
F. Perez (IPython creator) gives NUG Talk

Users running IPython via login nodes

Jupyter as a NERSC science gateway app

Access to Cori via cori19 enabled

Jupyter hub infrastructure moves to Spin, cgroups

JupyterLab Beta enabled at NERSC

IPython becomes Jupyter*

NERSC Talks, Papers, Posters, and/or Demos:
- SC16 • CUG17 • JupyterCon17
- IDEAS/ECP • ISC18
- JupyterCon18 • ECP2019
- Community Workshop • NUG2019

First JupyterCon

Jupyter team receives ACM Software Systems Award

* IPython became Jupyter, de-emphasizing the Python branding. Jupyter is language-agnostic.
<table>
<thead>
<tr>
<th>Use Case</th>
<th>Where</th>
<th>Why</th>
</tr>
</thead>
<tbody>
<tr>
<td>Light-weight data analysis and visualization</td>
<td>Spin Container <em>(In production now.)</em></td>
<td>Usable when other systems are down. Simple, interactive access</td>
</tr>
<tr>
<td>Workflow execution and medium-scale data analysis</td>
<td>Cori “Login” Nodes <em>(In production now.)</em></td>
<td>Access to batch and scratch Larger memory shared node</td>
</tr>
<tr>
<td>Heavy weight computation including task frameworks</td>
<td>Cori Compute Nodes <em>(In testing now.)</em></td>
<td>Dedicated resources (e.g. memory and cores). Ability to launch parallel workloads in the notebook.</td>
</tr>
</tbody>
</table>
Jupyter @ NERSC Architecture

Key

- Control
- User

**Web Browser**

**Spin**
- JupyterHub
- Jupyter (Spin)

**Services**
- SSH Auth API

**Cori{13,14,19}**
- SSH
- Jupyter

**Compute Nodes**
- Jupyter (Cori CN)

**Languages**
- R

**APIs**
- SDN API
“I really like the jupyter interface.”

“New jupyter notebooks are awesome!”

“Great interactive workflow (e.g. for postprocessing) via JupyterHub”

“I absolutely love the fact that I can use the Jupyter hub to access the Cori scratch directory. This allows me to analyze data through the browser ... or to quickly check that simulation runs are going as expected without having to transfer data to a different location. I actually also have access to other supercomputer clusters, but this is one of the biggest reasons I mainly use Cori and Edison for debugging and production runs.”

“I would really appreciate it if jupyter.nersc.gov wouldn’t go down as much as it does.”

“MPI cannot be used in jupyter notebook as well, where the jupyter hubs run on login nodes (unless when using the compute nodes through SLURM.)”
Jupyter on Cori Usage Numbers

Supporting these users, their current and future needs is a lot of work!
Jupyter at NERSC now has a ~monthly update schedule. Usually follows Cori maintenances by 2-5 days. We post an announcements when it’s coming up. Jupyter upgrades themselves are very short (usually...). Rarely change notebook env, mostly hub changes:

- **jupyter-test**: Staff only, Experimental work
- **jupyter-stage**: Staff & “friendly” user testing
- **jupyter**: Can roll back if deployment has issues

Brought to you by Spin
Deployment Schedule

18-10
- SSH Auth API (sshproxy)
- Implement MFA

18-12
- Add “offline” communication page
- “Announcements” service for communication

19-02
- Ability to leverage more Cori nodes

19-03
- Converged hubs: jupyter-dev and jupyter

19-05
- Computes and GPU node access (special access)
- Leverage Iris for role-based access control
- JupyterHub 1.0
Deployment Schedule

Jupyter on computes (general access)

Configurable jobs:
  Reservation (for training/tutorials/workshops)
  Full shifter image support

Integration with NERSC SSO

Jupyter Slurm integration
  Parameterized notebooks
  Managing curated notebooks

...
Workshop: Jupyter for Science User Facilities and High Performance Computing

Joint Workshop w/BIDS: June 11-12 at NERSC, June 13 at BIDS
Committee: Rollin Thomas • Shane Canon • Shreyas Cholia • Kelly Rowland • Debbie Bard • Dan Allan (BNL) • Chris Holdgraf (BIDS)

Part of “Jupyter Community Workshop” Series
Competitive application process • Granted up to $20K for travel support
Funds from Bloomberg, managed by NumFOCUS and Project Jupyter

User Facilities, HPC & Data Centers Represented
NSLS-II • LSST • APS • SLAC • JGI • ARM • European XFEL
NERSC • ALCF • TACC • MSI (@UMN) • Compute Canada • ESA

Content
Talks: Deployment • Infrastructure • Extending Jupyter for HPC • Use Cases
Breakouts: Organizing Collaboration • Securing Jupyter • Sharing Notebooks
Reproducibility • Best Practices • Future Plans • Tutorials
Roundtable Meeting with Core Jupyter Developers
Demo Intro

NERSC

U.S. DEPARTMENT OF ENERGY
Office of Science

BERKELEY LAB
NCEM Superfacility Highlight Slide

Scientific Achievement
Enable interactive human-in-the-loop computing for Superfacility Workflows at NERSC

Significance and Impact
Superfacility Projects like ALS and NCEM need a flexible interactive user interface to be able to analyze data from experiments in real time. We use Jupyter Notebooks to enable these projects to combine live code execution, reproducibility, interactive visualizations on HPC resources.

Research Details
- Command and control center for integrating distributed workflows under a common platform
- Enable HPC Tools to allow Jupyter notebooks to execute code in parallel on distributed backend resources using Dask/IPyParallel/MPI
- Realtime rendering of results rendered inline in Jupyter to enable live interaction, visualization and job steering
- Parameterized notebooks can to execute a workflow over multiple datasets and parameters.
- Enable reproducible workflows through project curated notebooks that can be cloned / executed

Achieved 90x speedup on NCEM Py4DSTEM Notebook running on HPC resources at NERSC: https://github.com/py4dstem/py4DSTEM/blob
Team

Rollin Thomas

Shane Canon

Kelly Rowland

Trevor Slaton

Shreyas Cholia

Matt Henderson

William Krinsman

Jon Hays
Demo
Future Work

- Streamline notebook ↔ running job interaction
- Exclusive node access (notebook on compute) in real-time-like queue
- Experiment with configurable-http-proxy for dynamic routing (not SDN)
- Support >300 users/day, maybe add 1-2 more shared nodes, manage via Slurm
- Deploy Jupyter solutions at NERSC:
  - Sharing notebooks, parameterized notebooks, curated notebooks
  - Voila and dashboards
  - More HPC-centric extensions and fixes to JupyterLab/Hub.
- Engagement with Jupyter and Jupyter-in-HPC communities
- User engagements through Superfacility and beyond: NCEM, ALS, DESI, LSST, …
Thank You
How It Works

The **Notebook Server** sends code (via **ZeroMQ**) to a language “**kernel**” that executes this code. In addition to running your code, it stores code and output, together with markdown notes, in an editable document called a **notebook**, saved as a **JSON** file with a `.ipynb extension`. 
JupyterHub: Jupyter as a Service

- Service to deploy notebooks in a multi-user environment
- Manages user authentication, notebook deployment and web proxies
JupyterHub Architecture

Components are abstracted:
  Authenticator
  Spawner
  Proxy

Pieces we’ve created:
  GSIAuthenticator (IT IS NO MORE)
  SSHAPIAuthenticator

  SSHSpawner (*Had gsissh support*)
  NERSCSpawner
  NERSCSlurmSpawner

Pieces we re-use and love:
  WrapSpawner (NERSCSpawner)
  BatchSpawner (NERSCSlurmSpawner)
Modern Frontend for Jupyter

- Integrated GUI for Jupyter ecosystem (filebrowser, tabbed panes, notebooks, terminal etc.)
- Common framework to integrate multiple applications - “extensions”
- eg. connect multiple viewers to common underlying kernel

Extensions for many users?
Right now we manage them all.

... we’ve made a Slurm one I’ll show later.
Handling Spawner Options

Populated by queries against internal REST API’s at NERSC (or example ones)
Jupyter architecture

- Allocate nodes on Cori interactive queue and start ipyparallel or Dask cluster
  - Developed %ipcluster magic to setup within notebook
- Compute nodes traditionally do not have external address
  - Required network configuration / policy decisions
- Distributed training communication is via MPI Horovod or Cray ML Plugin
Setting up ipyparallel cluster

Via Magic (entire workflow in notebook) or a console script

In [1]: import ipcluster_magics

In [2]: job_name = "isc_ihpc_mnist"
   nodes = 1
   engines = 1
   module = "python/3.6-anaconda-4.4"
   conda_env = "/global/cscratch1/sdfarrell/conda/isc-ihpc"

In [3]: ipcluster -m $module -e $conda_env -N $nodes -J $job_name -t 01:00:00

salloc: Pending job allocation 13289619
salloc: job 13289619 queued and waiting for resources
salloc: job 13289619 has been allocated resources
salloc: Granted job allocation 13289619

- Use a unique cluster ID for this job
- Use a unique cluster ID for this job

clusterID=cori_${SLURM_JOB_ID}
echo "Launching controller"
ipcontroller --ip="$headIP" \
   --cluster-id=$clusterID &
sleep 20
echo "Launching engines"
srun ipengine --cluster-id=$clusterID

Connect to cluster in notebook

In [7]:
   # Cluster ID taken from job ID above
   job_id = 13272466
   cluster_id = 'cori{0}'.format(job_id)
   
   # Use default profile
   c = ipp.Client(timeout=50, cluster_id=cluster_id)
Distributed Training

- Distributed training in notebooks with IPyParallel and Horovod-MPI
- Notebook cells specified for parallel execution using cell magic
  - MPI code in a notebook
- Scales well with no noticeable overhead from the infrastructure

```
# Model config
h1, h2, h3, h4, h5 = 64, 128, 256, 256, 512
lr = 0.001 * hvd.size()

# Training config
batch_size = 128
n_epochs = 4

# Build the model
model = build_model(train_input.shape[1],
                      h1=h1, h2=h2, h3=h3, h4=h4, h5=h5,
                      optimizer=optimizer, lr=lr,
                      use_horovod=True)
if hvd.rank() == 0:
    model.summary()

Layer (type)               Output Shape              Param #
=================================================================
input_2 (InputLayer)       (None, 64, 64, 1)           0

conv2d_5 (Conv2D)          (None, 64, 64, 64)          640
conv2d_6 (Conv2D)          (None, 32, 32, 128)        73856
conv2d_7 (Conv2D)          (None, 32, 32, 256)        295168
conv2d_8 (Conv2D)          (None, 16, 16, 256)        590080
flatten_2 (Flatten)        (None, 65536)              0
dense_3 (Dense)            (None, 512)               33554944
dense_4 (Dense)            (None, 1)                 513
```

Train with Horovod on all workers

```
# Train the model
history = train_model(model, train_input=train_input, train_labels=train_labels,
                       valid_input=valid_input, valid_labels=valid_labels,
                       batch_size=batch_size, n_epochs=n_epochs,
                       use_horovod=True)
```
Easy but powerful setup for random search HPO

- Define HP sets to evaluate
- Define model training function
- Run the HPO tasks with load-balanced scheduler

Launch user-defined training function and arguments
AsyncResult objects can be queried for status, outputs

---

**Distributed HPO - Setup**

```python
# Define the hyper-parameter search points
n_hpo_trials = 336
h1 = np.random.choice([4, 8, 16, 32, 64], size=n_hpo_trials)
h2 = np.random.choice([4, 8, 16, 32, 64], size=n_hpo_trials)
h3 = np.random.choice([8, 16, 32, 64, 128], size=n_hpo_trials)
conv_sizes = np.stack([h1, h2, h3], axis=1)
fcs = np.random.choice([32, 64, 128, 256], size=(n_hpo_trials, 1))
rlr = np.random.choice([0.0001, 0.001, 0.01], size=n_hpo_trials)
dropout = np.random.rand(n_hpo_trials)
optimizer = np.random.choice(['Adam', 'Adam', 'Nadam'], size=n_hpo_trials)

# Load-balanced view
lv = c.load_balance_view()

# Loop over hyper-parameter sets
results = []
for ihp in range(n_hpo_trials):
    print(f'Hyperparameter trial {ihp+1} conv [%s] fc [%s] dropout [%s] opt [%s] lr [%s] %
    (ihp, conv_sizes[ihp], fcs[ihp], dropout[ihp], optimizer[ihp], lr[ihp])
    checkpoint file = os.path.join(checkpoint dir, 'model %i.h5' % ihp)
    result = lv.apply(build_and_train,
                       input_dir, n_train, n_valid,
                       conv_sizes=conv_sizes[ihp], fc_sizes=fcs[ihp],
                       dropout=dropout[ihp], optimizer=optimizer[ihp], lr=lr[ihp],
                       batch_size=batch_size, n_epochs=n_epochs,
                       checkpoint file=checkpoint file)
    results.append(result)
```

Hyperparameter trial 0 conv [ 64  16 128] fc [128] dropout 0.3234 opt Nadam, lr 0.0100
Hyperparameter trial 1 conv [ 4  8 64] fc [64] dropout 0.6747 opt Adamelta, lr 0.0010
Plots update live

Table shows different configurations:
- Status
- Current loss and accuracy
- Sort

Can add further quantities to plot and interaction buttons

https://github.com/sparticle/steve/cori-intml-examples/
Model: (conv_sizes: [54, 6, 108], fc_sizes: [256], dropout: 0.8454368724771278, optimizer: Nadam, \eta: 0.0001)

<table>
<thead>
<tr>
<th>index</th>
<th>Not Started</th>
<th>start</th>
<th>conv shape</th>
<th>fc_sizes</th>
<th>dropout</th>
<th>optimizer</th>
<th>\eta</th>
<th>test loss</th>
<th>val loss</th>
<th>train acc</th>
<th>val acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td>[9, 4, 128]</td>
<td>[512]</td>
<td>0.890611</td>
<td>Nadam</td>
<td>0.2021</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>[9, 4, 128]</td>
<td>[512]</td>
<td>0.890611</td>
<td>Nadam</td>
<td>0.2021</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>[9, 4, 128]</td>
<td>[512]</td>
<td>0.890611</td>
<td>Nadam</td>
<td>0.2021</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>[9, 4, 128]</td>
<td>[512]</td>
<td>0.890611</td>
<td>Nadam</td>
<td>0.2021</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>[9, 4, 128]</td>
<td>[512]</td>
<td>0.890611</td>
<td>Nadam</td>
<td>0.2021</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>[9, 4, 128]</td>
<td>[512]</td>
<td>0.890611</td>
<td>Nadam</td>
<td>0.2021</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>[9, 4, 128]</td>
<td>[512]</td>
<td>0.890611</td>
<td>Nadam</td>
<td>0.2021</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>[9, 4, 128]</td>
<td>[512]</td>
<td>0.890611</td>
<td>Nadam</td>
<td>0.2021</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td>[9, 4, 128]</td>
<td>[512]</td>
<td>0.890611</td>
<td>Nadam</td>
<td>0.2021</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td>[9, 4, 128]</td>
<td>[512]</td>
<td>0.890611</td>
<td>Nadam</td>
<td>0.2021</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td>[9, 4, 128]</td>
<td>[512]</td>
<td>0.890611</td>
<td>Nadam</td>
<td>0.2021</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td>[9, 4, 128]</td>
<td>[512]</td>
<td>0.890611</td>
<td>Nadam</td>
<td>0.2021</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td>[9, 4, 128]</td>
<td>[512]</td>
<td>0.890611</td>
<td>Nadam</td>
<td>0.2021</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td>[9, 4, 128]</td>
<td>[512]</td>
<td>0.890611</td>
<td>Nadam</td>
<td>0.2021</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td>[9, 4, 128]</td>
<td>[512]</td>
<td>0.890611</td>
<td>Nadam</td>
<td>0.2021</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td>[9, 4, 128]</td>
<td>[512]</td>
<td>0.890611</td>
<td>Nadam</td>
<td>0.2021</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Curated Notebook Environments

- Browse curated examples managed by project using tools like nbviewer
- Clone and launch notebook into users workspace with appropriate conda environment
- Reproducible Notebooks – Similar to Binder
Parameterized Notebooks

- Run the same notebook against
  - different sets of parameters
  - Different datasets
  - Think “data parallel”
- Running as notebook gives you a live document of each task
- Save successful runs

```python
In [1]:
parameters =
1  # This cell is tagged 'parameters'
2  alpha = 0.1
3  ratio = 0.1

Executing a Notebook

The two ways to execute the notebook with parameters are: (1) through the Python API and (2) through the command interface.

Execute via the Python API

```python
import papermill as pm
pm.execute_notebook('path/to/input.ipynb', 'path/to/output.ipynb', parameters = dict(alpha=0.6, ratio=0.1))
```
Jupyterlab Extensions

- SLURM Extension
- Resource Usage Monitoring Extension