Extended Abstract: Enabling Interactive Notebooks on Supercomputers with Jupyterhub

Shreyas Cholia*; R. Shane Canon, Rollin Thomas
Lawrence Berkeley National Laboratory
*1 Cyclotron Road, Berkeley, CA 94720, USA; email: scholia@lbl.gov

Abstract: *Interactive notebook systems such as Jupyter represent a new paradigm in web science gateways that can combine interactive code execution with data analysis and exploration. In our work we demonstrate how one can create and manage interactive notebooks in a multi-user supercomputing environment using the Jupyterhub platform. We describe our architecture along with custom modules that we developed for Jupyterhub to manage authentication, notebook execution and interaction with the job queueing system. We illustrate the power of this system through the OpenMSI use case, and outline future directions for this effort.*

1. Interactive Notebooks and Jupyter

Interactive notebook environments are quickly gaining traction in the sciences by enabling a new kind of paradigm - iterative, live computational exploration. Their ability to combine an annotated mathematical narrative with live code execution and results creates a "literate computing" environment, where narrative and computation go hand in hand. This makes them uniquely valuable for the cycle that ranges from individual exploratory work to collaboration and publication, supporting sharing and reproducibility of results.

In particular the Jupyter [1] framework has played a key role in bringing this style of computing to a large number of users. Jupyter enables web-based notebook interfaces, which can combine backend computing language kernels (Python, R, Julia etc.), explanatory elements in Markdown or HTML along with interactive frontend visualizations and widgets in Javascript. Originally focused on Python, it now supports over 70 language [2] kernels along with a highly customizable codebase to enable development of specialized features.

Presented at Gateways 2016, San Diego Supercomputer Center, La Jolla, CA, November 2-3, 2016.
https://gateways2016.figshare.com/

The Jupyter ecosystem also includes Jupyterhub [3], a platform for deploying multi-user notebook environments. This framework is critical in making Jupyter useful across the project and institutional level, and transforming it into a general purpose Science Gateway service. Jupyterhub centralizes key areas like authentication, scaling and provisioning while also enabling fully integrated backend notebooks (with all the needed tools and libraries pre-installed) for users.

Our work focuses on how we have used the Jupyterhub platform to enable access to large-scale supercomputing resources, such as the NERSC Cori system. Note that the modules that we have developed are generally useful for deploying Jupyterhub in large distributed or cloud computing environments. Our code is freely available to enable others to use these modules [5][6][8].

2. Jupyterhub for Supercomputing

HPC workflows for today's large-scale data-intensive science largely focus on asynchronous, batch execution. However, scientific insight frequently requires interactive, iterative exploration and analysis. A bridge between these two modes of scientific computing is needed, enabling "human in the loop" exploration provided by interactive electronic notebooks, as a seamless element of asynchronous, large scale workflows.

In order to support this mode of computing, we are enabling Jupyter notebooks to access HPC and data resources on the NERSC Cray XC40 system, Cori [4]. Cori includes a "data partition" that is particularly suitable for these kinds of workloads with features like data friendly queuing policies, external network connectivity on compute nodes, burst buffer capabilities and a large pool of big-memory interactive nodes.

The Jupyterhub service enables web notebooks in a multi-user environment. When users log in,

they receive a notebook for their projects, which provides access to NERSC files and jobs, along with pre-installed notebook "kernels" (language interpreters that drive the notebook).

Our setup includes a Jupyterhub web service (running inside a Docker container for portability and scaling) that manages user authentication, and proxies subsequent requests to the Cori system. We have developed a custom authenticator for Jupyterhub called the GSI Authenticator [5] that allows users to acquire a grid certificate upon login. The service then uses a special spawner that we developed (SSH Spawner) [6], which spins up a Jupyter notebook on Cori via SSH (using the GSI credentials). The notebook connects back to the hub over a websocket. The hub then proxies all future user requests to the Cori node via this websocket connection. The NERSC Jupyterhub architecture is illustrated in figure 1.

Additionally we have also enabled special hooks (known as "magics" in Jupyter) for interacting with the SLURM [7] batch system on Cori. Our tool, called "Slurm-Magic" [8], enables users to directly submit and manage jobs via the notebook interface, and supports various SLURM commands as well as processing and formatting of queue information.

The NERSC notebook infrastructure provides access to several kernels (including different versions of Python, Root and custom libraries for specific projects). It also allows users to run their own kernels by simply creating a small JSON file that points to their own language interpreter.

Key benefits of this service include (i) direct access to large datasets, including terabytes written to the Cori scratch file system and global shared project filesystem, allowing users to write and view their data, (ii) access to the job queueing system, allowing users to submit jobs, query the batch systems and look at results and (iii) a secure, managed setup that allows users to directly login to the system with their NERSC credentials.

3. Use Case: OpenMSI

We illustrate the power of this type of interactive web notebook, through the OpenMSI [9] use case for high-throughput screening of spotted samples using mass-spectrometry imaging. The OpenMSI team can log into NERSC machines from their browsers using Jupyterhub, see all their files on a common file-system, add ions of particular interest (from data stored at NERSC) and then center the samples on a grid for analysis. The grids contain 384 samples, each with a different spectrum. This essentially represents a canned workflow with all the steps needed to analyze a given sample contained in the notebook, while giving the researcher complete control to be able to tweak and modify any step in the analysis. See figure 2 for an example notebook workflow.

4. Related and Future Work

There is similar work being done at SDSC [12], the University of Minnesota [13], and TACC [14] to commission notebooks directly on compute nodes through a graphical interface to the batch system manager. We believe that our work is complementary to these efforts, as we have added new capabilities (GSI Authentication, pure SSH spawning and SLURM magics) to this ecosystem.

As we move forward, it will become very important to integrate tools and frameworks that manage advanced workflow and distributed computing constructs, directly into the Jupyter infrastructure. In particular two frameworks that we are actively investigating are Spark [10] and Dask [11], with the idea that users can control data-parallel map-reduce style operations from within these notebooks in an effective manner.

5. Conclusions

We believe that notebook-style interactive frameworks represent a novel way to approach scientific computing. Using a multi-user platform like Jupyterhub we have demonstrated that this can be extended to include traditional HPC environments, allowing users to develop custom workflows that combine the power of interactivity with large scale jobs and data analysis. This allows users to easily move back and forth between exploratory analysis and large-scale batch processing without leaving the browser environment. Additionally it facilitates reproducibility and sharing of workflow processes across a science team (as demonstrated by the OpenMSI example).

6. Figures

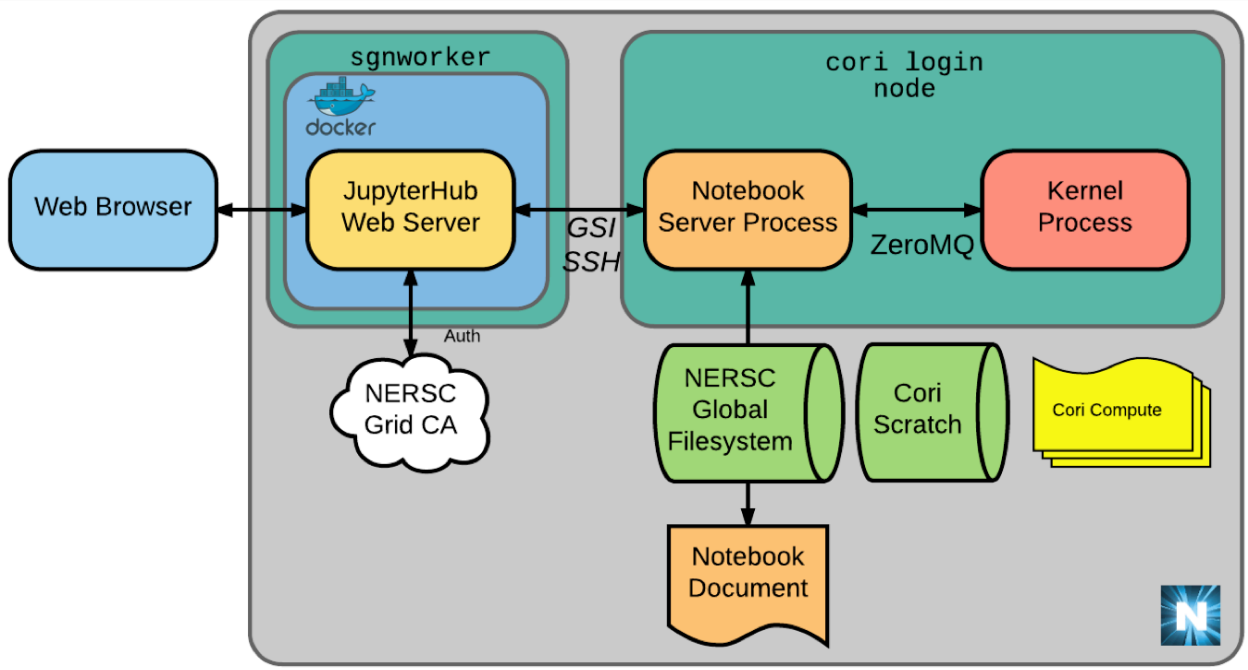
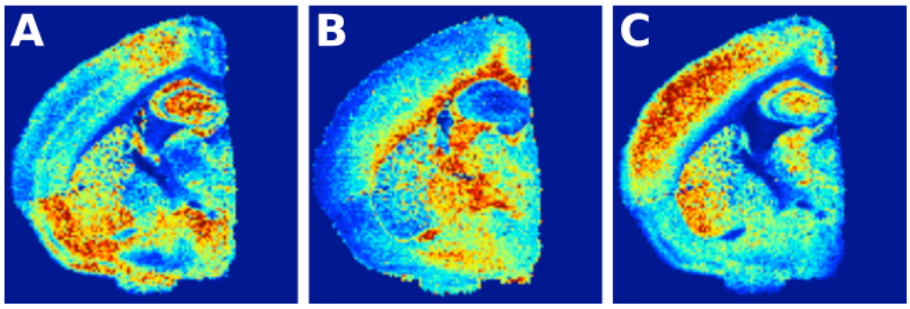


Fig. 1. NERSC Jupyterhub architecture diagram. Jupyterhub runs on a science gateway node called *Sgnworker* while the notebook itself runs inside the *Cori* system.

In [12]:
```
# print model.components_.shape
from skimage import exposure

nmfdata = H.reshape(3,Nx,Ny)
fig = plt.figure(1, (11., 11.))
grid = ImageGrid(fig, 111, # similar to subplot(111)
                 nrows_ncols = (1, 3), # creates 2x2 grid of axes
                 axes_pad=0.1, # pad between axes in inch.
                 )

for i in range(3):
    img = nmfdata[i,:,:]
    p2, p98 = np.percentile(img, (1, 99))
    img_eq = exposure.rescale_intensity(img, in_range=(p2, p98))
    grid[i].imshow(img_eq) # The AxesGrid object work as a list of axes.
    grid[i].axis('off')
    grid[i].text(2,16,chr(i+65),fontsize=30,color='white',weight='bold')
fig.savefig('Figure_4_nmf_images.pdf')
```



In [13]:
```
fig = plt.figure(1, (11.,7.))
for i in range(3):
    plt.subplot(3,1,i+1)
    plt.stem(peakCubeIons, W[:,i], markerfmt=" ")
    plt.xlim([750, 900])
    plt.text(plt.axis()[0]+2,plt.axis()[3]*0.75,chr(i+65+3),fontsize=26,color='black',weight='bold')
    if i==1:
        plt.ylabel('Magnitude (a.u.)',fontsize=20,weight='bold')


plt.xlabel('m/z',fontsize=20,weight='bold',style='italic')
fig.savefig('Figure_4_nmf_spectra.pdf')
```

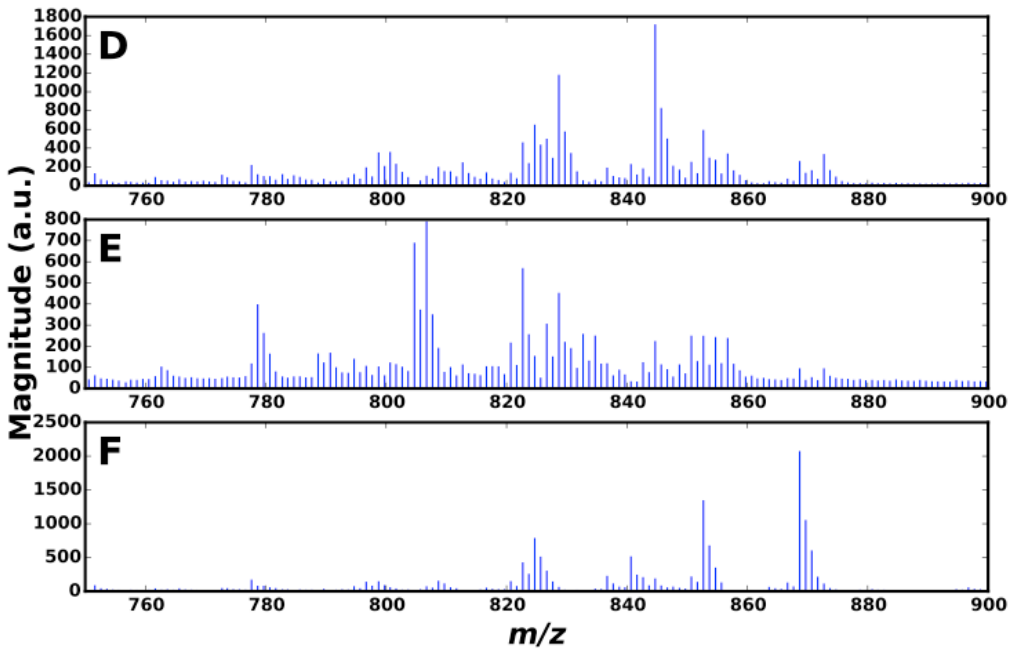


Fig 2. Programmatic visualization illustrating the use of dimensionality reduction using non-negative matrix factorization (NMF) on publicly available OpenMSI data using a Jupyter notebook

7. Acknowledgments

This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

8. References

[1] Jupyter Project, http://jupyter.org/
[2] IPython Documentation, https://github.com/ipython/ipython/wiki/IPython-kernels-for-other-languages
[3] Jupterhub, https://github.com/jupyterhub/jupyterhub
[4] Wright, Nicholas J et al. "Cori: A Pre-Exascale Supercomputer for Big Data and HPC Applications." Big Data and High Performance Computing 26 (2015): 82.
[5] GSI Authenticator, https://github.com/NERSC/gsiauthenticator
[6] SSH Spawner, https://github.com/NERSC/sshspawner
[7] Yoo, Andy B, Morris A Jette, and Mark Grondona. "Slurm: Simple linux utility for resource management." Workshop on Job Scheduling Strategies for Parallel Processing 24 Jun. 2003: 44-60.
[8] SLURM Magic, https://github.com/NERSC/slurm-magic
[9] Fischer, Curt R, Oliver Ruebel, and Benjamin P Bowen. "An accessible, scalable ecosystem for enabling and sharing diverse mass spectrometry imaging analyses." Archives of Biochemistry and Biophysics 589 (2016): 18-26.
[10] Shanahan, James G, and Laing Dai. "Large scale distributed data science using apache spark." Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 10 Aug. 2015: 2323-2324
[11] Dask, http://dask.pydata.org/
[12] Zonca, Andrea, "Run Jupyterhub on a Supercomputer", https://zonca.github.io/2015/04/jupyterhub-hpc.html
[13] Batch Spawner, https://github.com/jupyterhub/batchspawner
[14] TACC Visualization Portal https://vis.tacc.utexas.edu