Interactive Supercomputing Through Jupyter

Shreyas Cholia (scholia@lbl.gov)[1], Matthew Henderson[1], Rollin Thomas[2], Shane Canon[2], Kelly L. Rowland[2], Steve Farrell[2], Wahid Bhimji[2], Lavanya Ramakrishnan[1], Matthias Bussonnier[3], Ian Rose[4], Fernando Pérez[4]

LBL – Computational Research Division[1], LBL – NERSC[2], UC Merced[3], UC Berkeley[4]

Interactive Supercomputing

Scientific insight is an iterative exploratory process. Tomorrow's *Superfacility* workflows and data analysis pipelines will need to provide **real-time, interactive** support for data analysis and exploration.

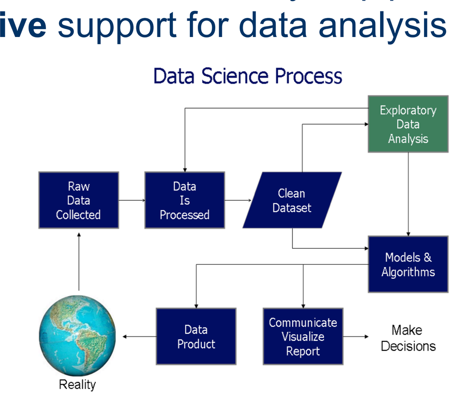


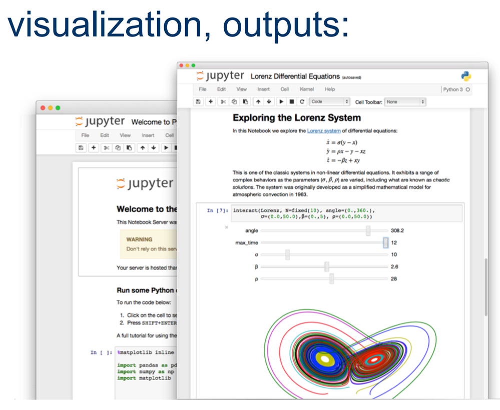
Diagram courtesy of "Farcaster" at English Wikipedia

Jupyter provides a powerful **web-based** platform that allows for **human-in-the-loop** interaction with scientific narratives.

Jupyter enables reproducible, shareable narratives and literate computing.

The Jupyter Notebook: Interactive document served over HTTP containing code, comments, visualization, outputs:

- Live code
- Rich text
- Interactive plots
- Equations
- Widgets



Our vision is to supercharge human-in-the-loop workflows **at scale** for data-intensive science and high-performance computing.
We are enabling exploratory data analytics, deep learning, workflows, and more through Jupyter on NERSC systems.

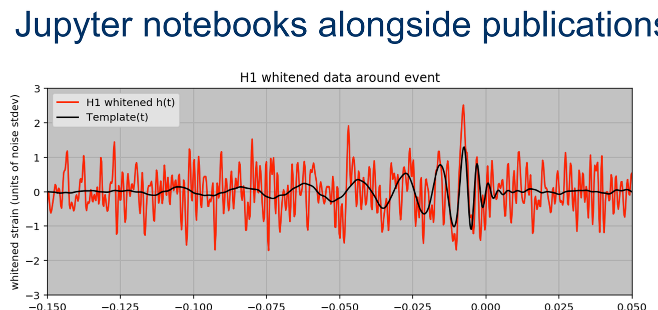
Jupyter In Science

Jupyter is already playing a major role in science and education.

2017 ACM Software System Award:
"… a de facto standard for data analysis in research, education, journalism and industry … more than 2,000,000 Jupyter notebooks are on GitHub …"

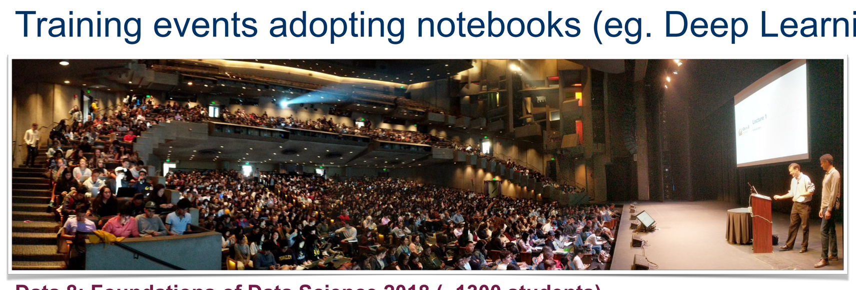
Integral part of Big (Data) Science & Superfacility:
ALS, NCEM, LSST-DESC, DESI, LCLS, Materials Project, KBase …

Supporting Reproducibility and Science Outreach:
Open source code and open source science
Jupyter notebooks alongside publications (LIGO)



LIGO Binary BH-BH Merger GW Signature
Figure from LIGO EPO/Publication Jupyter Notebook

Generational Shift in Analytics for Science and More:
UC Berkeley's Data Science 8 course, entirely in Jupyter
"I'll send you a copy of my notebook"
Training events adopting notebooks (eg. Deep Learning tutorials)

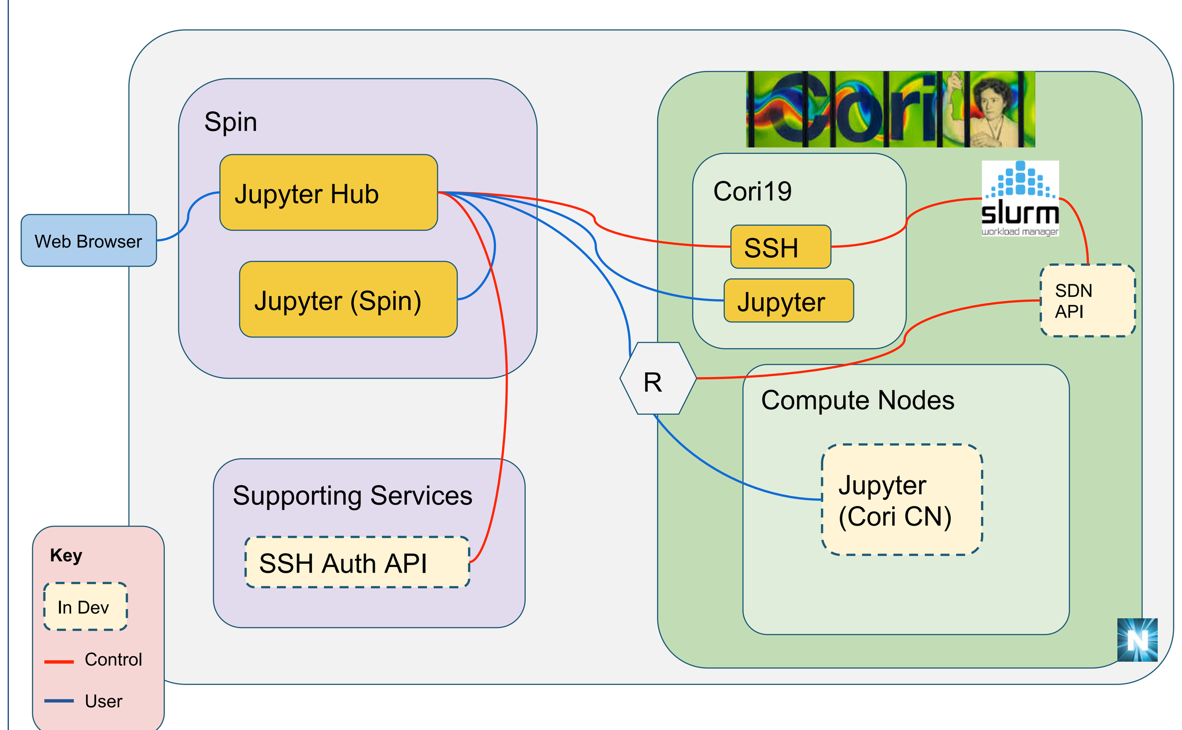


Data 8: Foundations of Data Science 2018 (~1300 students)

Upcoming Workshops:
Jupyter for Scientific User Facilities and High-Performance Computing
June 11-13 2019, hosted in Berkeley at LBL & BIDS

Jupyter At NERSC

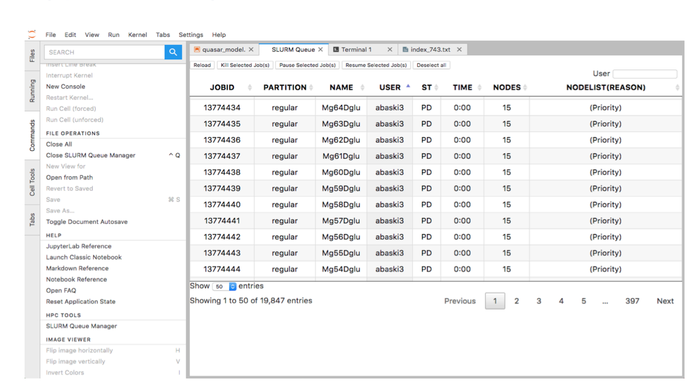
NERSC is making Jupyter available to its users through **JupyterHub,** a web service that enables multi-user institutional deployments of Jupyter. Notebooks are launched on various backends, including Cori, and have access to underlying resources such as Global and Scratch filesystems, job queues and network resources.



NERSC has also been an early adopter of **JupyterLab**, the next-generation web-based user interface for Project Jupyter that enables multiple applications and viewers alongside the Jupyter notebook through a common backplane.

NERSC and CRD have partnered to develop various tools to support the Jupyter ecosystem including:

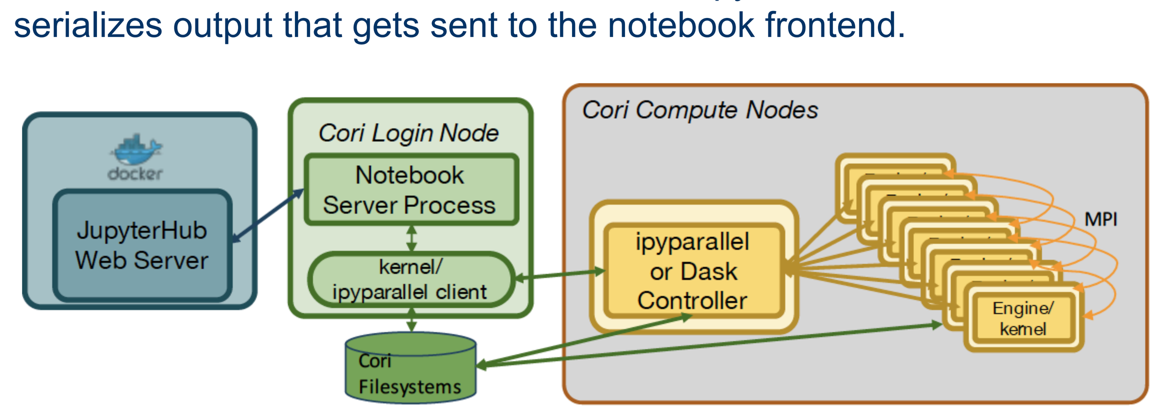
- Async SSH Spawner
- Pre-Spawn Access Checks (quota etc.)
- SSHProxy Authenticator
- Slurm Magics
- Dockerized Deployment
- JupyterLab Slurm
- Kale



JupyterLab with SLURM plugin

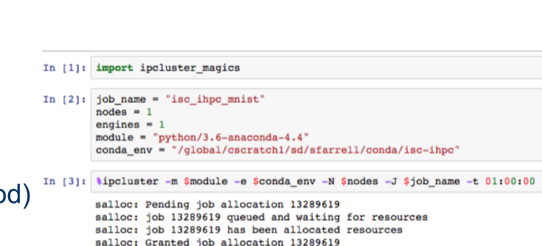
Parallel Computing

A key component of this problem involves scaling up the Jupyter machinery for big HPC and data-intensive jobs. Our approach involves running a core notebook server process alongside a task execution engine like Dask or IPyParallel. The execution engine handles the task orchestration and communicates with the Jupyter kernel. The kernel serializes output that gets sent to the notebook frontend.



Setup and Execution:

- Allocate nodes on Cori interactive queue
- Start IPyParallel or Dask cluster
- %ipcluster magic setup in notebook
- Distributed communication via MPI (eg. Horovod)



Kale: Jupyter Extension for HPC

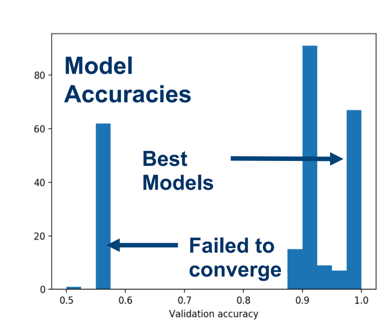
- Enables human-in-the-loop computing for HPC with Jupyter
- Control (start/stop) remote tasks from Notebooks
- Resource monitoring (Task/Node)
- Hook into HPC tasks and interact with them directly

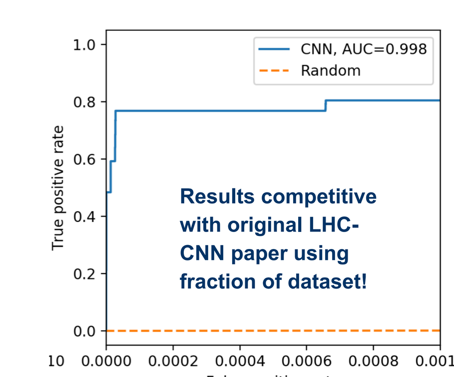
Jupyter in Deep Learning

Deep learning using neural networks represents the next frontier in scientific insight and discovery. Training of complex networks can take days. Jupyter boosts this iterative process through interactive exploration and human intuition, to go alongside brute-force scans and automated optimization. As such, Jupyter has become the de facto tool of choice for the DL community and is the standard development environment. In our work we explore using Jupyter to use deep learning to classify ATLAS particle physics data.

Distributed Deep Learning and Hyperparameter Optimization

- Use-case: CNN for particle physics (LHC ATLAS) classification
- IPyParallel and Keras + Horovod-MPI (MPI in notebook)
- Scales well - no overhead from notebook infrastructure
- Run HPO tasks with load-balanced IPyParallel scheduler





Interactive Human-in-the-Loop HPC

Jupyter is connected to a parallel engine to enable steering of distributed training tasks, exploration of the hyper-parameter space and rapid insight through real-time rendering of model training results.

- Live plots of model output + dynamic status table with *bqlot* and *qgrid*
- Buttons and form for stopping/restarting models via *IPywidgets*
- *IPyParallel* engines publish data monitored via background threads
- *Kale* resource monitoring and control with *Plotly*

Remote Data with Jupyter

Some datasets are too big for Jupyter frontends. Only a subset of the data may actually be needed by the client. How do we handle this transparently? This is one of the topic areas under the *Usable Data Abstractions* project, in collaboration with UC Berkeley and UC Merced. We are building mechanisms in Jupyter to use a "just-in-time" model of data fetching, where the dataset lives remotely and Jupyter only pulls in the subset it needs.

- User requests a dataset => the server intercepts request before contents are returned
- Send Metadata, or API endpoints to stream the data to frontend => Jupyter frontend sees smaller payload, and can decide what to do with it (inspect, stream, etc)
- Mount the remote dataset so that it appears to be "local"
- Enable streaming of big remote datasets into client

Acknowledgements and Links

Links:

- NERSC Jupyter Deployment: https://github.com/NERSC/
  - sshspawner, sshapiauthenticator, jupyterlab-slurm, jupyterhub-deploy slurm-magic
- Deep Learning Examples: https://github.com/sparticlesteve/cori-intml-example
- Kale: https://github.com/Jupyter-Kale/kale
- Jupyter Community Workshop: https://bit.ly/jup-sfhpc

Funding:
Jupyter research and development at LBL has been supported through funding from the LBL LDRD program, NERSC, and DOE ASCR.