Jupyter - An Interactive Platform for Scientific Computing

Shreyas Cholia

Group Leader, Usable Software Systems
Lawrence Berkeley National Laboratory
Why?
Why?

- **Ethical**: openness as fairness
- **Human/social**: openness fosters collaboration.
- **Epistemological**: proprietary science is an oxymoron.
- **Technical**: Python was cool :)

In reality, programming languages are how programmers express and communicate *ideas* — and the audience for those ideas is *other programmers*, not computers.

– Guido van Rossum

http://neopythonic.blogspot.com/2016/04/kings-day-speech.html
a *community* of people and an *ecosystem* of open tools and standards for interactive computing
create things that are language-agnostic and modular. Empower people to use other open tools.
What?
IPython: Jupyter
IPython: Interactive Python, 2001

A humble start: IPython 0.0.1, 259 LOC

“Just an afternoon hack”

https://gist.github.com/fperez/1579699
Team today: where all the credit goes

Plus ~ 1500 more Open source contributors!
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.
The IPython/Jupyter Notebook

- Rich web client
- Text & math
- Code
- Results
- Share, reproduce.
How It Works

The Notebook Server sends code (via 0mq) 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.
Core ideas of the web: HTTP & HTML

HTTP: protocol to connect clients and servers
HyperText Transport Protocol

HTML: format to represent content
HyperText Markup Language

Image credit: eviltester.com
Core ideas of Jupyter

Interactive Computing Protocol

Document Format

$\text{ØMQ + JSON}$
Jupyter Protocol
web-age capture of the process of interactive computing

any mime-type output
- text
- svg, png, jpeg
- latex, pdf
- html, javascript
- interactive widgets
Jupyter Protocol is language agnostic

~100 different kernels: https://github.com/jupyter/jupyter/wiki/Jupyter-kernels
Did you know there is a SPARQL kernel for @ProjectJupyter notebooks? I didn’t! So much fun to be had querying wikidata ... try it: mybinder.org/v2/gh/betattim/ ... What do US presidents die of?

```cpp
In [1]: #include <iostream>
#include "xtensor/xarray.hpp"
#include "xtensor/xio.hpp"

In [2]: xt::xarray< double > arr1
{ { 1.0, 2.0, 3.0 },
{ 2.0, 5.0, 7.0 } },
{ 2.0, 5.0, 7.0 } );
xt::xarray< double > arr2
{ { 5.0, 6.0, 7.0 } );
std::cout << xt::view(arr1, 1) + arr2;
{ 7.0, 11.0, 14.0 }
```

```cpp
In [3]: #include <iostream>
#include "xtensor/xarray.hpp"
#include "xtensor/xio.hpp"

In [4]: xt::xarray< int > arr
{ { 1, 2, 3, 4, 5, 6, 7, 8, 9 } );
arr.reshape({ 3, 3 });
std::cout << arr;
{ 1, 2, 3 },
{ 4, 5, 6 },
{ 7, 8, 9 } }
```

Classic ‘Notebook’…
JupyterLab: a grand unified theory of Jupyter

Huge Team Effort!

C. Colbert, S. Corlay, A. Darian, B. Granger, J. Grout, P. Ivanov, I. Rose, S. Silvester, C. Willing, J. Zosa-Forde
The Lorenz Differential Equations

Before we start, we import some preliminary libraries. We will also import (below) the accompanying lorentz.py file, which contains the actual solver and plotting routine.

We explore the Lorenz system of differential equations:

\[
\begin{align*}
\dot{x} &= \sigma (y - x) \\
\dot{y} &= \rho x - y - xz \\
\dot{z} &= -\beta z + xy
\end{align*}
\]

Let's change (\(\sigma, \beta, \rho\)) with ipywidgets and examine the trajectories.

1. The Lorenz Differential Equations

Before we start, we import some preliminary libraries. We will also import (below) the accompanying lorentz.py file, which contains the actual solver and plotting routine.

We explore the Lorenz system of differential equations:

\[
\begin{align*}
\dot{x} &= \sigma (y - x) \\
\dot{y} &= \rho x - y - xz \\
\dot{z} &= -\beta z + xy
\end{align*}
\]

Let's change (\(\sigma, \beta, \rho\)) with ipywidgets and examine the trajectories.
Beyond notebooks

Here is a block of Python code in the markdown file:

```python
# Python

Let's attach a Python 3 Kernel and Console to this markdown file. Then we can select lines of code in the markdown file and run them in the console by pressing 'Shift+Enter'. Let's do something more complicated:

First import `matplotlib`, `numpy` and `pandas`, and create a data frame:

```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

data = {
    'x': np.random.randn(100),
    'y': np.random.randn(100),
    'color': np.random.randn(100),
    'size': 100.*np.random.rand(100)
}
df = pd.DataFrame(data)
df.head()
```

And make a scatter plot:

```python
plt.scatter('x', 'y', c='color', s='size', data=df, cmap=plt.cm.Blues)
plt.xlabel('x')
plt.ylabel('y')
plt.title('The data that we collected')
```

All of the Python objects are live in the console and can be explored further.
Data
Data: reusable within code
Community-extensible: Neurohackademy 2018
JupyterHub: multiuser support

Jupyter for Organizations

JupyterHub is a multiuser version of the notebook designed for centralized deployments in companies, university classrooms and research labs.

- **Pluggable authentication**: Manage users and authentication with PAM, OAuth or integrate with your own directory service system. Collaborate with others through the Linux permission model.

- **Centralized deployment**: Deploy the Jupyter Notebook to all users in your organization on centralized servers on- or off-site.

- **Container friendly**: Use Docker containers to scale your deployment and isolate user processes using a growing ecosystem of prebuilt Docker containers.

- **Code meets data**: Deploy the Notebook next to your data to provide unified software management and data access within your organization.
CODING ENVIRONMENT

AUTHENTICATION

jupyterhub

Slides credit: C. Holdgraf
What does this mean for science + education?

❖ Can utilize…
❖ ...shared hardware/compute for running code
❖ ...shared data storage for big datasets
❖ ...shared environments for doing work
❖ ...shared workflows, ideas, and results
NERSC

• Mission HPC center for US Dept. of Energy
  – 7000+ diverse users across science (e.g. cosmology, climate, biosciences)

• Cori – Cray XC40 (31.4 PF Peak)
  – 9668 Intel Knights Landing (KNL), 2388 Haswell nodes

• Deep learning: Data and analytics (DAS) group:
  – Tools for machine learning; optimized for scale
  – Cutting-edge methods/Collaborations/Training

• Interactive Computing at NERSC:
  – Modifications to SLURM including real-time and interactive queues with dedicated resource
  – Also other interactive features not described here: (visualization; science gateways etc.)

Slides credit: S. Farrell et al, LBNL
Parallel Jobs via Jupyter

- **JupyterHub Web Server**
- **Cori Login Node**
  - Notebook Server Process
  - kernel/ipyparallel client
  - Cori Filesystems
- **Cori Compute Nodes**
  - ipyparallel or Dask Controller
  - Engine/kernel

- MPI
Interactive Distributed Deep Learning

https://github.com/sparticlesteve/cori-intml-examples/
“As mentioned, the ability to access data from the scratch directories through the Jupyter hub is very important to my workflow.”

“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.*”

“New jupyter notebooks are awesome!”

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

[Venkitesh: “... jupyter notebooks are very important for me: *The 3 most important things in life: food, shelter and jupyter... everything else is optional.”*]
Reproducible Research

An article about computational science in a scientific publication is **not** the scholarship itself, it is merely **advertising** of the scholarship. The **actual scholarship** is the complete software development environment and the complete set of instructions which generated the figures.

*Buckheit and Donoho, WaveLab and Reproducible Research, 1995*
mybinder.org: shareable reproducibility

github.com/freeman-lab
BinderHub

ON-DEMAND ENVIRONMENTS

CONTENT ON THE WEB
repo2docker

Convert a repository into a Docker image that runs the code inside.

repo2docker.readthedocs.io
repo2docker: what does it do?

Identify requirements

- README.md: Update README.md
- index.ipynb: first move
- requirements.txt: Pin requirements.txt to values that actually exist
generate Dockerfile

assemble the environment for the repo

... 

# Copy and chown stuff.
COPY src/ ${HOME}
RUN chown -R ${NB_USER}:${NB_USER} ${HOME}

# Run assemble scripts! These will actually build the specified
# in the repository into the image.
USER ${NB_USER}
RUN ${KERNEL_PYTHON_PREFIX}/bin/pip install --no-cache-dir
   -r "requirements.txt"

...
repo2docker
what does it do?

build (push) image

docker build -t myimage
docker push myimage
repo2docker
what does it do?

$ jupyter repo2docker \
> https://github.com/minrk/ligo-binder

Cloning into
'/var/folders/.../T/repo2dockermu6z66sd'...
Using CondaBuildPack builder
Step 1/31 : FROM buildpack-deps:bionic
    ---> 29f4eef41002
Step 2/31 : ENV DEBIAN_FRONTEND=noninteractive
    ---> Using cache
    ---> ee1ba7c4f5f4
    ---> Using cache
BinderHub

One-click sharable, interactive, reproducible environments from your public git repository

mybinder.org
binderhub.readthedocs.io
binderhub = repo2docker + jupyterhub
BinderHub is open tech...

- Built on Kubernetes
- Cloud-agnostic
- Scalable
- Community driven and deployable by anyone
Turn a GitHub repo into a collection of interactive notebooks

Have a repository full of Jupyter notebooks? With Binder, open those notebooks in an executable environment, making your code immediately reproducible by anyone, anywhere.
2017 NOBEL PRIZE IN PHYSICS AWARDED FOR DISCOVERY OF GRAVITATIONAL WAVES
A long time ago in a galaxy far, far away...

\[ R_{\mu\nu} - \frac{1}{2} R g_{\mu\nu} + \Lambda g_{\mu\nu} = \frac{8\pi G}{c^4} T_{\mu\nu} \]
Two identical detectors: Hanford, WA and Livingston, LA

LIGO: a feat of science & engineering

Detection problem:
- $\approx 1/1000$ proton over 4 km.
- Sensitivity $\approx 1e^{-21}$
- Milky Way: $1e+21m$ across!
LIGO: Open Science with Jupyter
FIG. 1. The gravitational-wave event GW150914 observed by the LIGO Hanford (H1, left column panels) and Livingston (L1, right column panels) detectors. Times are shown relative to September 14, 2015 at 09:50:45 UTC. For visualization, all time series are filtered with a 35–350 Hz bandpass filter to suppress large fluctuations outside the detectors’ most sensitive frequency band, and band-reject...
The song of the universe

Using the IPython.display.Audio object

---

**Make sound files**

Make wav (sound) files from the filtered, downsampled data, +/-2s around the event.

```python
# make wav (sound) files from the whitened data, +/-2s around the event.
from glob import glob
from IPython.display import display, Audio

from scipy.io import wavfile

# function to keep the data within integer limits, and write to wavfile:
def write_wavfile(filename, fs, data):
    d = np.int16(data / np.max(np.abs(data)) * 32767 * 0.9)
    wavfile.write(filename, int(fs), d)

tevent = 1126259462.422 # Mon Sep 14 09:50:45 GMT 2015
deltat = 2. # seconds around the event

# index into the strain time series for this time interval:
indxt = np.where((time >= tevent-deltat) & (time < tevent+deltat))

# write the files:
write_wavfile('GW150914_H1_whitenbp.wav', int(fs), strain_H1_whitenbp[indxt])
write_wavfile('GW150914_L1_whitenbp.wav', int(fs), strain_L1_whitenbp[indxt])
write_wavfile('GW150914_NR_whitenbp.wav', int(fs), NR_H1_whitenbp)

for wav in glob('*whitenbp.wav'):
    display(Audio(filename=wav))

'GW150914_H1_whitenbp.wav'
```
**mybinder.org: usage**

Weekly Binder users, Jan 1 2018 to present (~350k/month)

_Berkeley:_ Yuvi Panda, Chris Holdgraf

_Cal Poly:_ Carol Willing

_Simula:_ Min Ragan-Kelley, Jessica Zosa-Forde, Tim Head

Binder users, Nov 1 2018 to present
Education
Berkeley’s Data Science Courses

- Freshmen & upper division
- Interactive textbooks: Jupyter Notebooks
- Course deployment: JupyterHub
Supporting 2,500+ users

Being used for Data 8, as well as several other courses

Requires @berkeley.edu to access

Running on Azure with almost zero maintenance

Data Hub
datahub.berkeley.edu

Slide: C. Holdgraf
Fall 2018, Berkeley: Data 8, ~1,300 students

Data 100, ~820
From K-12 to HPC

Jupyter ‘All-in-One’ Science Platform

Learning and sharing in a flexible, collaborative and interactive way.

Jupyter is an integrative application that incorporates math, science and engineering tools, along with communication and visualization resources, in one web-based platform. Simply put: it enables a broad suite of computing capabilities on any device that has an internet connection. For free.

Cybera and the Pacific Institute for the Mathematical Sciences (PIMS) have teamed up to increase access to, and awareness of, Jupyter. Cybera is hosting the platform on its Rapid Access Cloud, and is offering free access (and advice on how to get started) to Canada’s public and innovation sectors.

Who Is Jupyter Useful For?

- Teachers
  - Create "interactive textbooks" that allow students to actively work on math or data science problems.
- Students
  - Learn and practice multiple programming languages, and log experiments, all in one place.
- K-12
  - An Alaska high school teacher used Jupyter to change the way "Introduction to Programming" is taught
- Post-Secondary
  - A professor at George Washington University used Jupyter to teach Aerodynamics-hydrodynamics (see github notes)

Launch Jupyter at your university, school or company?
Wide industrial adoption
Funding and resources
What’s Next for Jupyter in HPC?
Jupyter R&D at LBL

❖ **Superfacility** - Shreyas Cholia
  ❖ Exploring how Jupyter can act as the primary *Superfacility* interface through interactive widgets and tools for experimental and observational data. Follow up on LDRD work integrating Jupyter with HPC.

❖ **Usable Data Abstractions** - Lavanya Ramakrishnan
  ❖ Enabling access to very large remote datasets through Jupyter and Jupyterlab

❖ **Jupyterhub** - Rollin Thomas
  ❖ Software development to enable integration of Jupyterhub with NERSC systems in a scalable, secure and intuitive manner
Where Do We Go From Here?

Scientists and researchers are already using Jupyter. Now we need to enable this at scale.

- Jupyter is designed to be modular and pluggable
- Increased emphasis on integration with big data and compute

- How do we enable them to use this on the big systems and large datasets?
- How do we connect and capture distributed facilities and workflows in Jupyter?
- How do we enable reproducible science at scale? Binder for HPC?
- What would YOU like to see?
Jupyter Community Workshop

Jupyter for Scientific User Facilities and High-Performance Computing Workshop to be held in Berkeley, California, from **Tuesday June 11 to Thursday June 13, 2019.**

Thank You!

Contact: scholia@lbl.gov
"Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Networking and Information Technology Research and Development Program."

The Networking and Information Technology Research and Development (NITRD) Program

Mailing Address: NCO/NITRD, 2415 Eisenhower Avenue, Alexandria, VA 22314

Physical Address: 490 L'Enfant Plaza SW, Suite 8001, Washington, DC 20024, USA Tel: 202-459-9674, Fax: 202-459-9673, Email: nco@nitrd.gov, Website: https://www.nitrd.gov