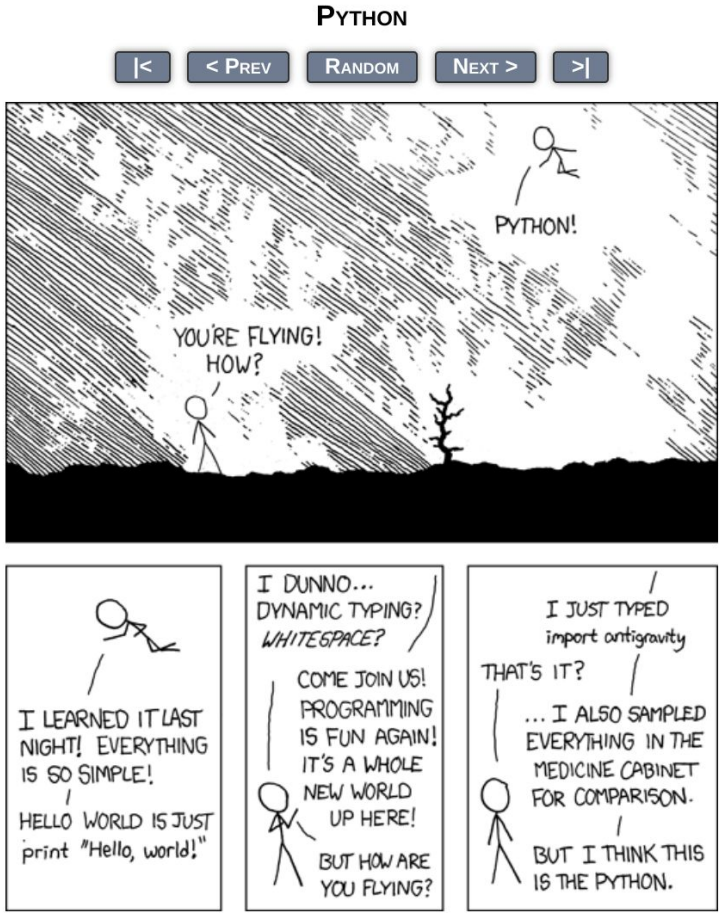
We start at **10:20**

Python for HPC

Andrea Zonca - SDSC



Topics

- Architecture of JupyterLab on Expanse
- Single-node Python code optimization with numba
- Dask tutorial: overlap functions, introduction to dask array, distributed scheduler
- Dask array in-depth tutorial for multi-core, out-of-core, **multi-node** computing

Jupyter Notebook

Data exploration in your browser

What is the notebook?

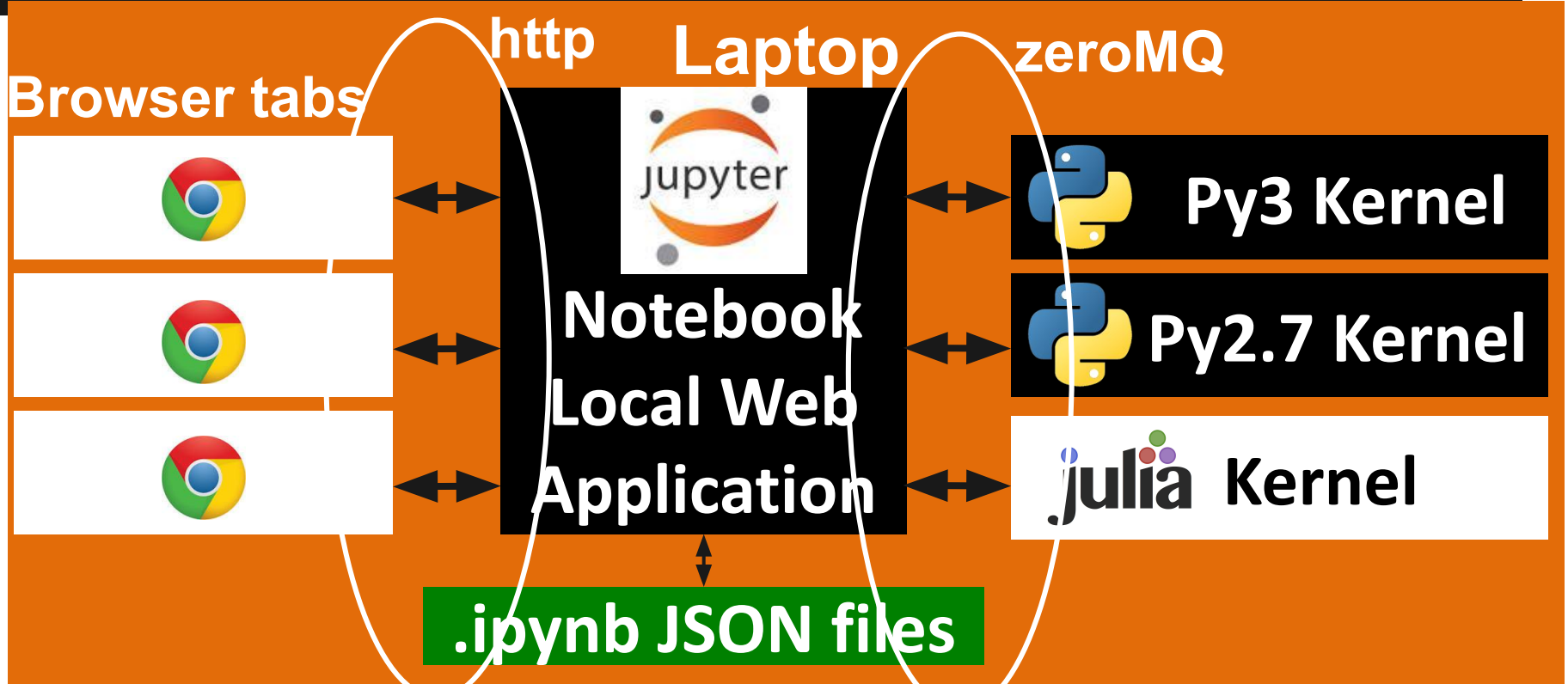
- Browser based interactive console
- Supports multiple sessions in browser tabs
- Each session has a Kernel executing computation
- Saved in JSON format

Notebooks for LIGO

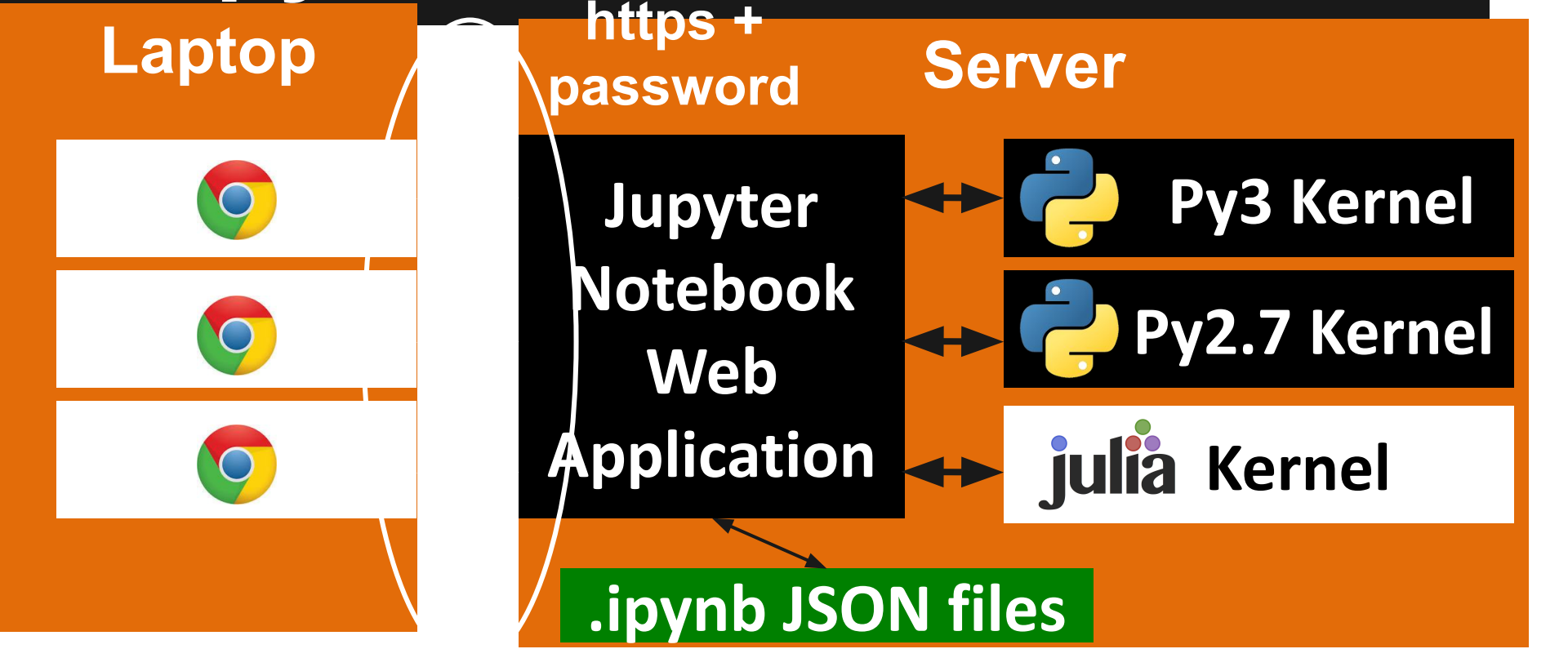
Interactive data analysis of gravitational waves from black holes merging:

http://beta.mybinder.org/repo/losc-tutorial/LOSC_Event_tutorial

Jupyter notebook local



Jupyter notebook remote



Clone workshop repository

ssh into Expanse with training account

```
git clone URL
cd  sdsc-summer-institute-2024
```

URL is
https://github.com/sdsc/sdsc-summer-institute-2024

Launch notebook job

```
cd  *python_hpc/
bash  launch_jupyter_singularity.sh
```

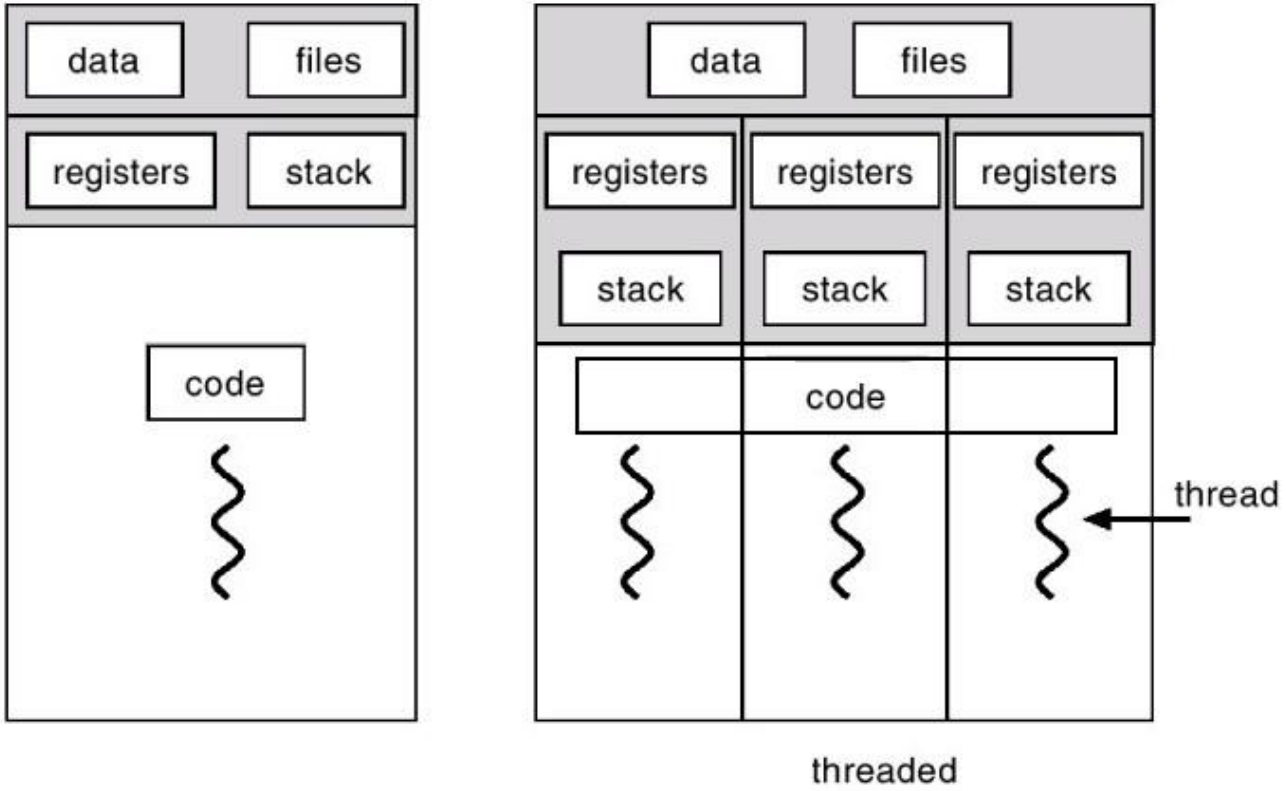
Check your job status with:

```
squeue -u $USER
```

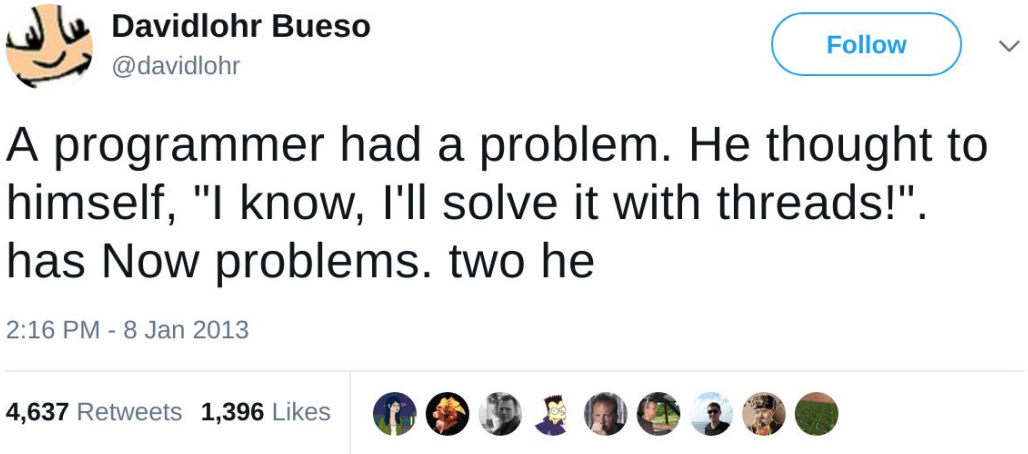
(after 2 min) Open browser on your laptop and connect to URL

Threads and processes

Threads vs processes



Out of order execution



Numba

Run code on GPU with Python

JIT compiler for Python

- based on LLVM (compiler infrastructure behind clang, Apple's C++ compiler)
- turns Python code into machine code
- on-the-fly

Guidelines

For optimizing Python code

Single machine

1) If data fits in memory: profile code (%prun in Jupyter), JIT-compile computationally heavy functions with numba (using nopython=True, parallel=True)
2) If data does not fit in memory: dask can use chunking to process data without loading all in memory (use threads and not processes)

Single machine

3) If have slow processed (disk or network): overlap them to other computations with dask.delayed

Multiple machines

Run dask distributed scheduler and workers, best is if you can run 1 worker per machine using all available threads.

Run numba-optimized code with parallel=False, let dask do threading.

Pack all calculations in 1 single dask.compute(), so dask has freedom to optimize.