Easy Interactive Supercomputing on Maxwell

Christopher Passow

Deutsches Elektronen-Synchrotron

September 17, 2019

Scientific perspective

1. Open Laptop
2. ...
3. ...
4. ...
n. Do Science

Scientific perspective

1. Open Laptop
2. ...
3. ...
4. ...
n. Do Science

We do not care & it should be easy!

Easy | Interactive | Supercomputing on Maxwell

Easy Interactive Supercomputing on Maxwell

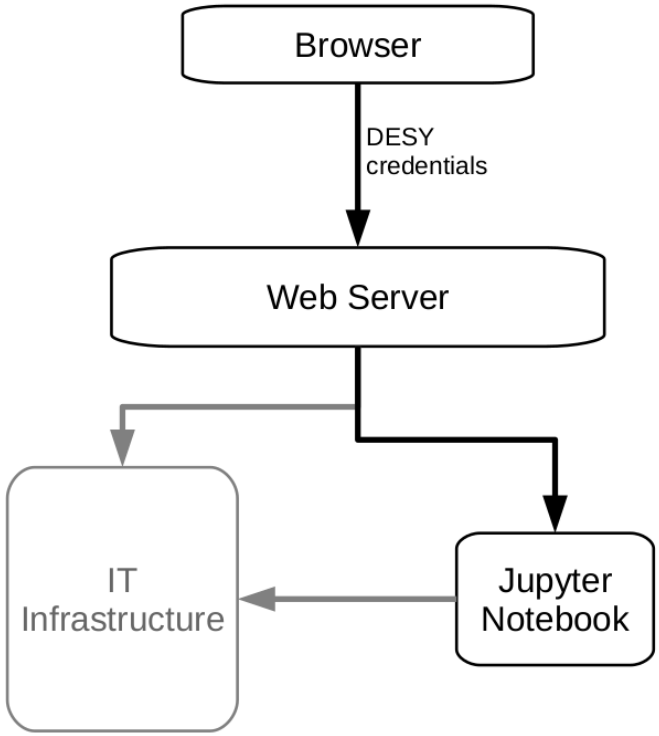
- Slurm scheduler for batch processing [1]
- Slurm submission nodes:
  - max-fsc | max-fsc
  - max-display
  - max-jhub

```
# simple job which prints hostname
[@max-wgs ~]$ cat hostname.sh
#!/bin/bash
#SBATCH --partition=maxwell
#SBATCH --time=00:10:00                                  # Maxi
#SBATCH --nodes=1                                        # Numb
##### note: from SLurm news file 17.11.0rc1:
##### Change --workdir in sbatch to be --chdir as in all
#####SBATCH --workdir   /home/mmuster/slurm/output    # 
#SBATCH --chdir   /home/mmuster/slurm/output      # dire
#SBATCH --job-name  hostname
#SBATCH --output    hostname-%N-%j.out            # File
#SBATCH --error     hostname-%N-%j.err            # File
#SBATCH --mail-type END                           # Type
#SBATCH --mail-user max.muster@desy.de            # Emai

/bin/hostname

# submit to batch queue for one node with one task
# requesting 10 mins of wall time
[@max-wgs ~]$ sbatch hostname.sh
Submitted batch job 2163

[@max-wgs ~]$ ls
hostname.sh  slurm-2163.out

[@max-wgs ~]$ cat slurm-2163.out
max-wn004.desy.de
```

Easy Interactive Supercomputing on Maxwell

- IPython Console
- Jupyter Notebooks
- JupyterLab

- JupyterHub @DESY [2]



Easy Interactive Supercomputing on Maxwell

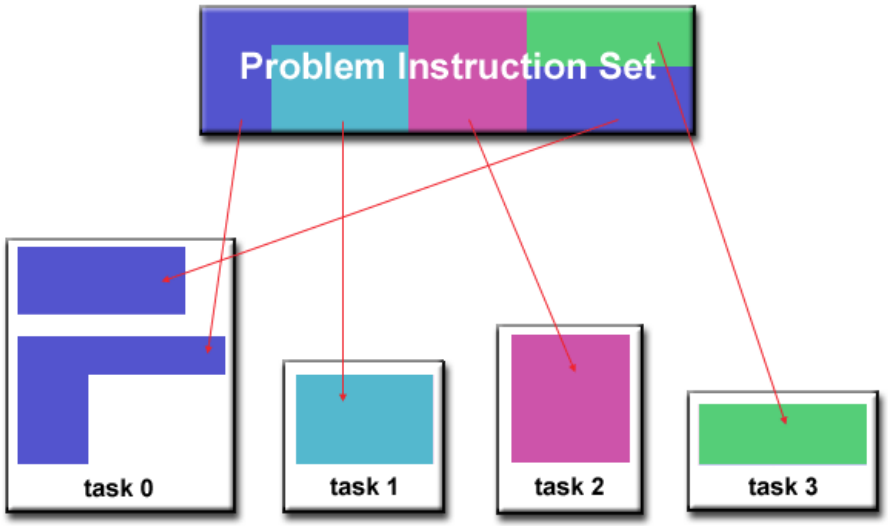
- Anaconda: venvs
  - requirements.txt

You shall not

- import mpi4py

Use external framework

- Dask for parallel computing.
  - Dynamic task scheduling
  - "Big Data" collections for distributed environments or larger-than-memory



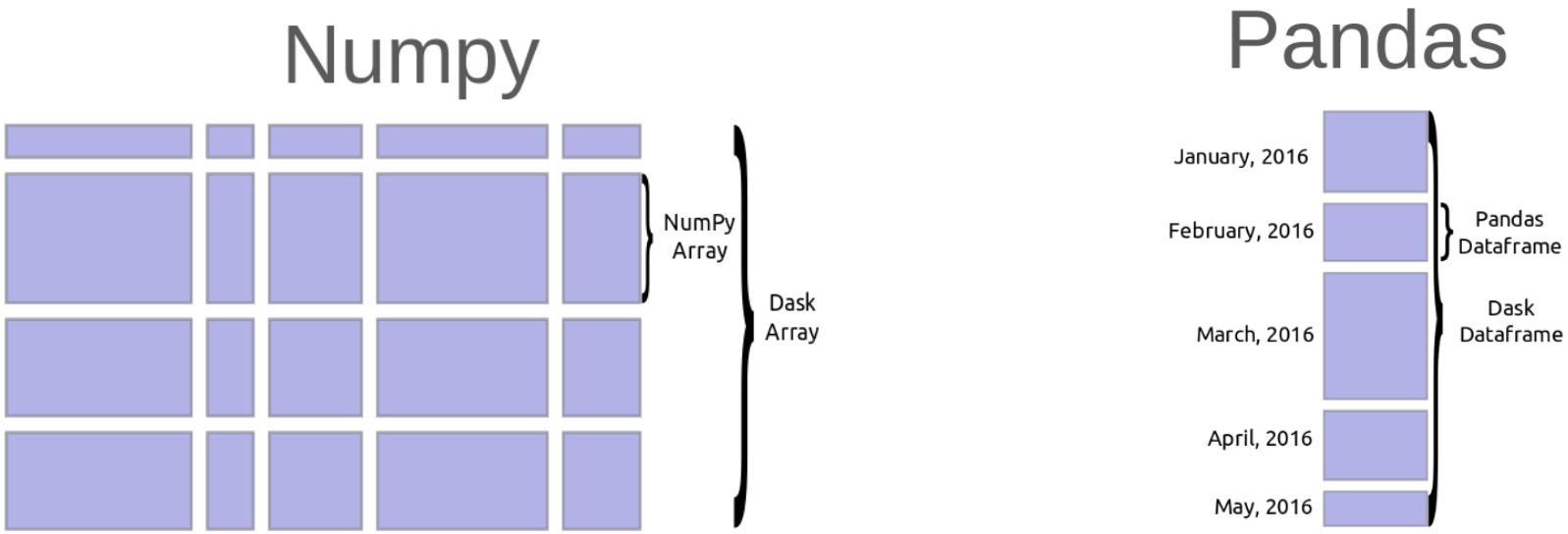
computing.llnl.gov

Demo

Demo #1

- DESY's JupyterHub
- *dask.delayed* as way to parallelize existing code
- fetch via:
  git clone `https://github.com/chrispassow/dask_demo.git`

dask.array & dask.dataframe



https://dask.org/

- for distributed environments or larger-than-memory
- most of *Numpy* & *Pandas* methods are implemented

Demo

Demo #2

- using a single node on Maxwell via *sbatch*
- compare *numpy.array* and *dask.array*

Demo

Demo #3

- using multpile nodes on Maxwell interactivly via *dask_jobqueue*

Terminate Idle Slurm Jobs

Danger Zone

"With great power there must also come – great responsibility!" [3]

Below is an (automatically generated) list of partitions and the limits applying

Partition	# of nodes	Nodes/Job	Max Number of Jobs	Default Time	Maximum Time	Allowed groups
ps	35	1	5	1:00:00	14-00:00:00	max-ps2-users
psx	16	1	5	1:00:00	7-00:00:00	max-psx2-users

Emergancy Commands [4]

scancel -u your_uid

Summary and Outlook

Current situation

- manual work is lengthy and prone to error
- Conda envs already simplify
- Slurm batch procession

The other side of the fence

- Frameworks to speed up analytics or scale to large data
- Dask as a state-of-the-art solution
- Jupyter notebooks for interactive data exploration
- JupyterHub as easy to use gateway

Great responsibility

Be careful about the resources you allocate.

References

[1] https://confluence.desy.de/display/IS/Running+Jobs+on+Maxwell

[2] https://confluence.desy.de/display/IS/JupyterHub+on+Maxwell

[3] Uncle Ben, Amazing Fantasy #15, Marvel Comics

[4] https://www.rc.fas.harvard.edu/resources/documentation/convenient-slurm-commands/