

INTERACTIVE SUPERCOMPUTING WITH



Anderson Banihirwe (@andersy005), Software Engineer

National Center for Atmospheric Research (NCAR)

SciPy 2019, Austin, TX.

Slides: <https://andersonbanihirwe.dev/talks/dask-jupyter-scipy-2019.html>



- Alice, project scientist @ NCAR
- Field of Expertise:
Hydrology/Hydrometeorology



3 INTERESTING THINGS ABOUT ALICE'S NOTEBOOK

1) NCAR INFRASTRUCTURE

JupyterLab

https://jupyterhub.ucar.edu/ch/user/abanihi/lab

NCAR infrastructure/JupyterHub running on Cheyenne

gmet_ensemble-zarr.ipynb

Python [conda env:analysis]

Analysis of Gridded Ensemble Precipitation and Temperature Estimates over the Contiguous United States

For this example, we'll work with 100 member ensemble of precipitation and temperature data.

Link to dataset: https://www.earthsystemgrid.org/dataset/gridded_precip_and_temp.html

Try running this notebook in the cloud: https://binder.pangeo.io/v2/gh/pangeo-data/pangeo-tutorial-agu-2018/master?filepath=notebooks%2Fgmet_ensemble.ipynb

```
[1]: %matplotlib inline
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import xarray as xr
import matplotlib.pyplot as plt
import dask
from distributed.utils import format_bytes
import hvplot.pandas
import hvplot.xarray
```

2) DISTRIBUTED COMPUTING RESOURCES

Connect to Dask Distributed Cluster

```
[2]: from dask.distributed import Client
      from dask_jobqueue import PBSCluster
      cluster = PBSCluster(memory="109GB", cores=12, processes=12, walltime="00:30:00",
                           queue="economy")
      # Scale adaptively (minimum of 10 nodes = 120 dask workers )
      cluster.adapt(minimum=12*10, maximum=12*20, wait_count=60)
      cluster
```

PBSCluster

Workers	120
Cores	120
Memory	1.09 TB

▸ Manual Scaling

▸ Adaptive Scaling

3) ACTUAL SCIENCE

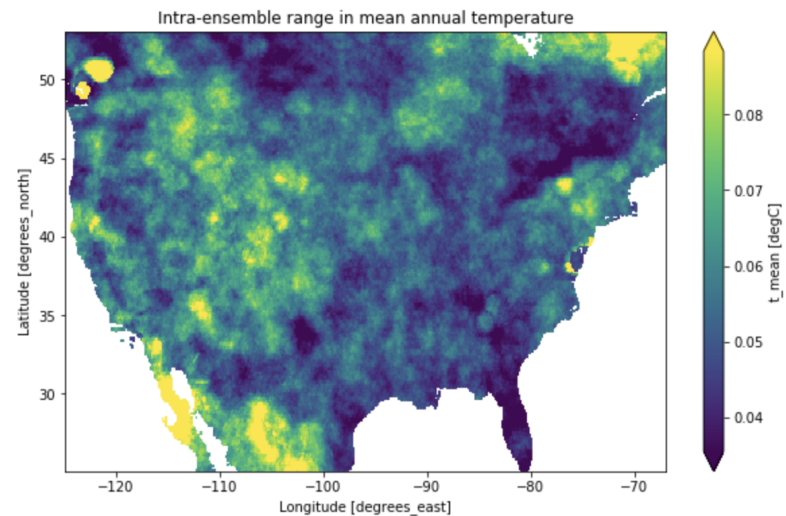
```
[13]: spread = spread.persist(retries=2)
spread

[13]: <xarray.DataArray 't_mean' (lat: 224, lon: 464)>
dask.array<shape=(224, 464), dtype=float32, chunksize=(224, 464)>
Coordinates:
  * lat      (lat) float64 25.12 25.25 25.38 25.5 ... 52.62 52.75 52.88 53.0
  * lon      (lon) float64 -124.9 -124.8 -124.6 -124.5 ... -67.25 -67.12 -67.0
```

Figure: Intra-ensemble range

```
[14]: spread.attrs['units'] = 'degC'
spread.plot(robust=True, figsize=(10, 6))
plt.title('Intra-ensemble range in mean annual temperature')

[14]: Text(0.5, 1.0, 'Intra-ensemble range in mean annual temperature')
```



```
[18]: buf = 0.25 # look at Austin +/- 0.25 deg
ds_tx = ds.sel(lon=slice(-97.7431-buf, -97.7431+buf),
               lat=slice(30.2672-buf, 30.2672+buf))

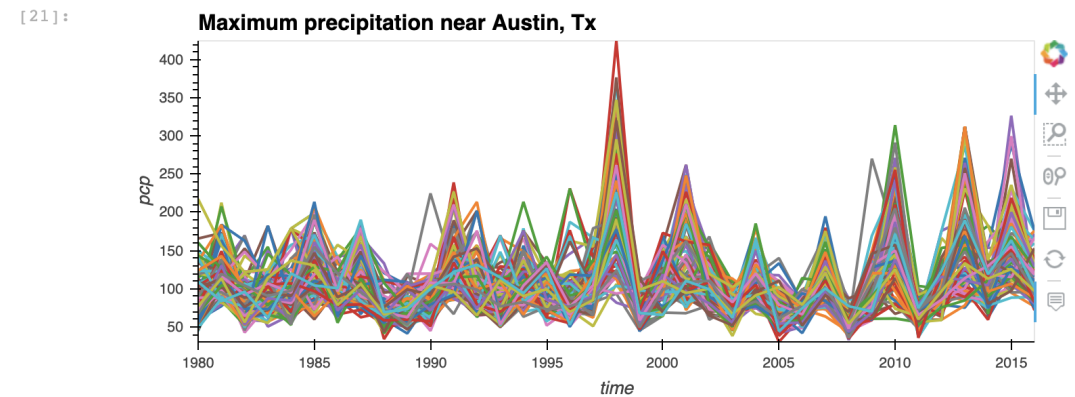
[19]: pcp_ann_max = ds_tx['pcp'].resample(time='AS').max('time')

[20]: pcp_ann_max_ts = pcp_ann_max.max(('lat', 'lon')).persist()
pcp_ann_max_ts

[20]: <xarray.DataArray 'pcp' (member_id: 100, time: 37)>
dask.array<shape=(100, 37), dtype=float32, chunksize=(1, 1)>
Coordinates:
  * time      (time) datetime64[ns] 1980-01-01 1981-01-01 ... 2016-01-01
  * member_id (member_id) int64 1 2 3 4 5 6 7 8 9 ... 93 94 95 96 97 98 99 100
```

Figure: Timeseries of maximum precipitation near Austin, Tx.

```
[21]: pcp_ann_max_ts.hvplot.line(x='time', title='Maximum precipitation near Austin, Tx',
                              legend=False)
```

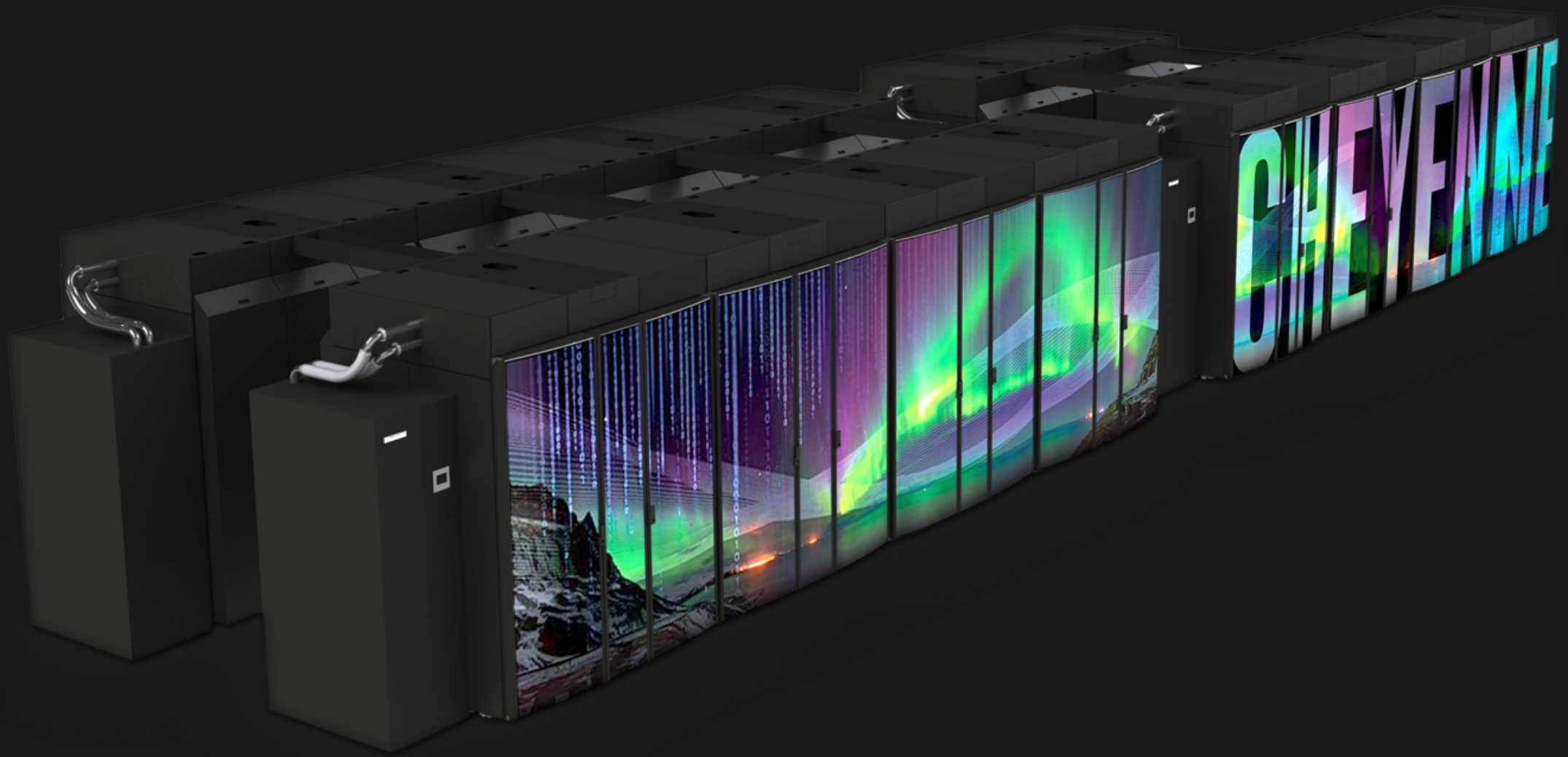


WHAT DO WE MEAN BY SUPERCOMPUTING?

WHAT DO WE MEAN BY SUPERCOMPUTING?

- MPI, batch processing...
- Lots of heavy machines managed by sysadmins...

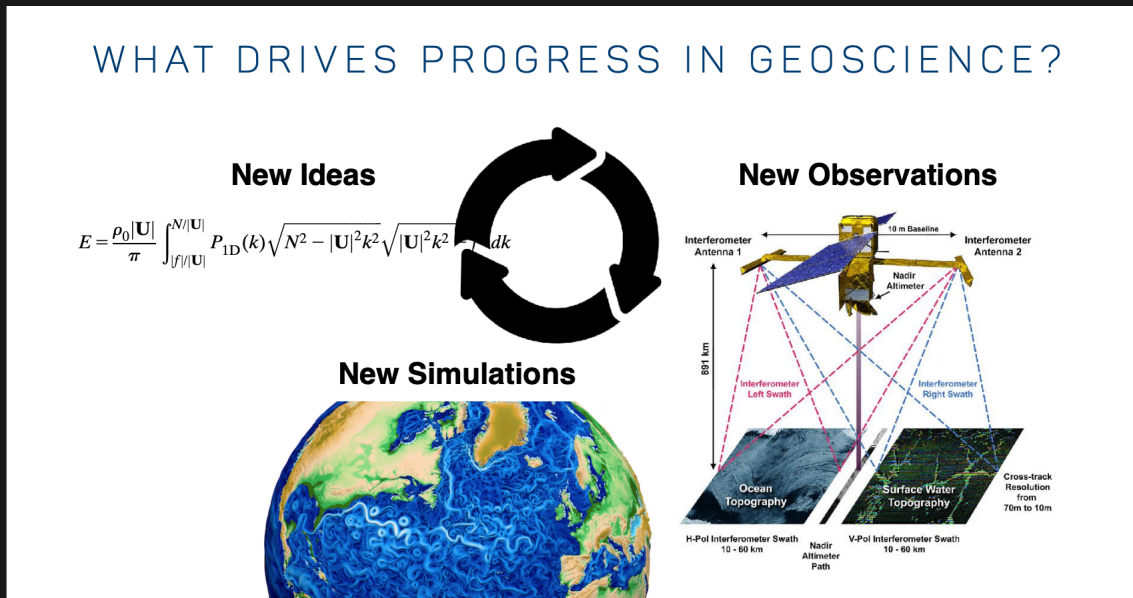
Cheyenne is a 5.34-petaflops, high-performance computer operated by NCAR.



WHAT DO WE MEAN BY INTERACTIVE SUPERCOMPUTING?

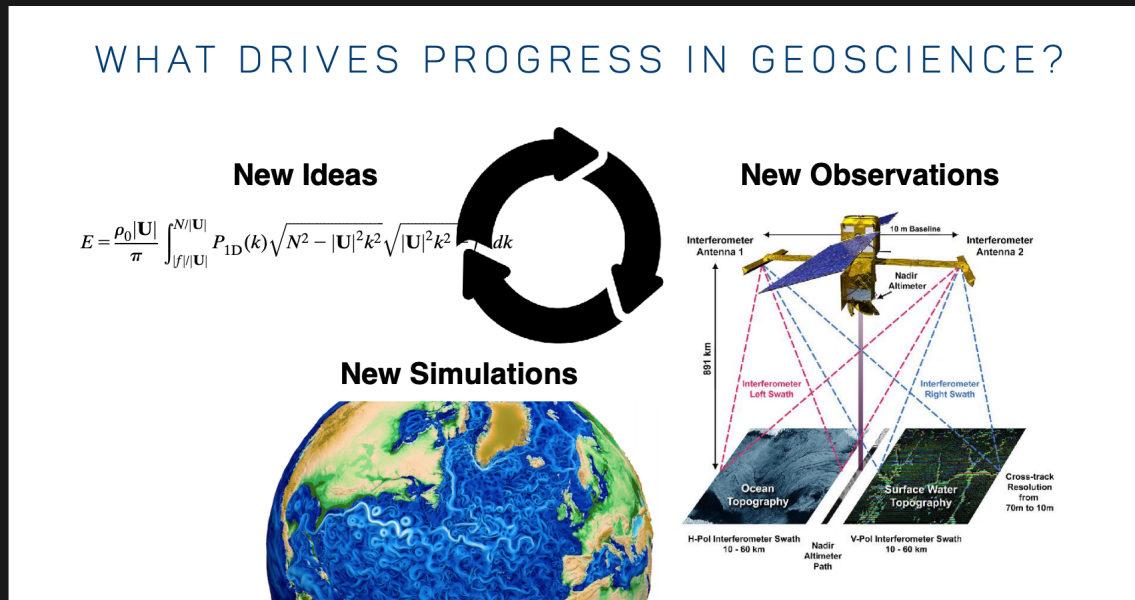
WHAT DO WE MEAN BY INTERACTIVE SUPERCOMPUTING?

WHAT DRIVES PROGRESS IN GEOSCIENCE?



- Need for more "human-in-the-loop" workflows, rapid iteration due to huge growth in data creation
- Jupyter notebooks, interactive visualization, etc
- Adaptive scaling of computing resources based on the load

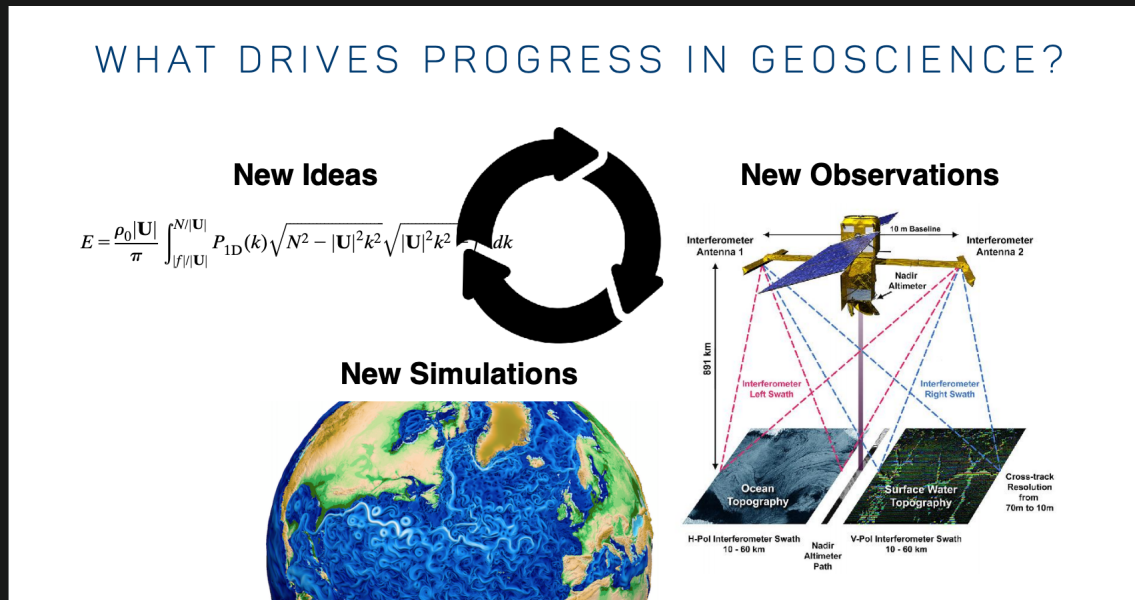
WHAT DO WE MEAN BY INTERACTIVE SUPERCOMPUTING?



- Need for more "human-in-the-loop" workflows, rapid iteration due to huge growth in data creation
- Jupyter notebooks, interactive visualization, etc
- Adaptive scaling of computing resources based on the load

This combination would be powerful...

WHAT DO WE MEAN BY INTERACTIVE SUPERCOMPUTING?



- Need for more "human-in-the-loop" workflows, rapid iteration due to huge growth in data creation
- Jupyter notebooks, interactive visualization, etc
- Adaptive scaling of computing resources based on the load

This combination would be powerful...

But it is hard...

INTERACTIVE SUPERCOMPUTING CHALLENGES

- Every high performance computing (HPC) system is unique:
 - Security policies
 - Container experience/policy
 - Queue configuration
 - External node access policies
- Tension between interactive availability and machine utilization (HPC centers often measured on this)...
- Lack of "elastic scaling" support in HPC workload managers...

ENABLING TECHNOLOGIES FOR INTERACTIVE SUPERCOMPUTING

jupyter

...back to oxygen

Set contour levels to non-uniform intervals and make the colormap centered at the hypoxic threshold using DivergingNorm.

```
In [7]: levels = [0, 10, 20, 30, 40, 50, 60, 80, 100, 125, 150, 175, 200, 225,
              250, 275, 300]

norm = colors.DivergingNorm(vmin=levels[0], vmax=levels[-1], vcenter=60.)
```

Add a cyclic point to accommodate the periodic domain.

```
In [8]: from cartopy.util import add_cyclic_point
field, lon = add_cyclic_point(ds.O2, coord=ds.lon)
lat = ds.lat
```

Putting it all together...

```
In [9]: fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(1, 1, 1, projection=ccrs.Robinson(central_longitude=305.0))

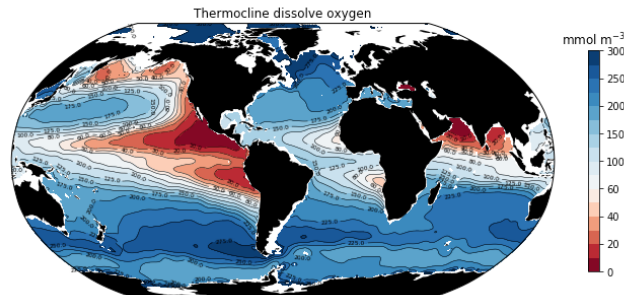
# filled contours
cf = ax.contourf(lon, lat, field, levels=levels, norm=norm, cmap='RdBu',
                transform=ccrs.PlateCarree());

# contour lines
cs = ax.contour(lon, lat, field, colors='k', levels=levels, linewidths=0.5,
                transform=ccrs.PlateCarree())

# add contour labels
lb = plt.clabel(cs, fontsize=6, inline=True, fmt='%r');

# land
land = ax.add_feature(
    cartopy.feature.NaturalEarthFeature('physical', 'land', '110m', facecolor='black'))

# colorbar and labels
cb = plt.colorbar(cf, shrink=0.5)
cb.ax.set_title('mmol m-3')
ax.set_title('Thermocline dissolve oxygen');
```



- Interactive, web browser-based computing environment
- Reproducible document format.
 - Code
 - Prose
 - Equations (LaTeX)
 - Visualizations

JUPYTER NOTEBOOKS ON HPC SYSTEMS

Q: But isn't Jupyter already usable on HCP systems?

Q: But isn't Jupyter already usable on HCP systems?

A: Yes, But.....

- SSH-in

```
$ ssh <remote_user>@<remote_host>
```

- Launch Jupyter on a remote machine

```
$ jupyter lab --no-browser --ip=`hostname` --port=<port>
```

- Set up SSH-tunnel to the remote machine

```
$ ssh -N -L <port>:<hostname>:<port> <remote_user>@<remote_host>
```

- Open the notebook in a browser on the local machine

```
$ open http://localhost:<port>/
```

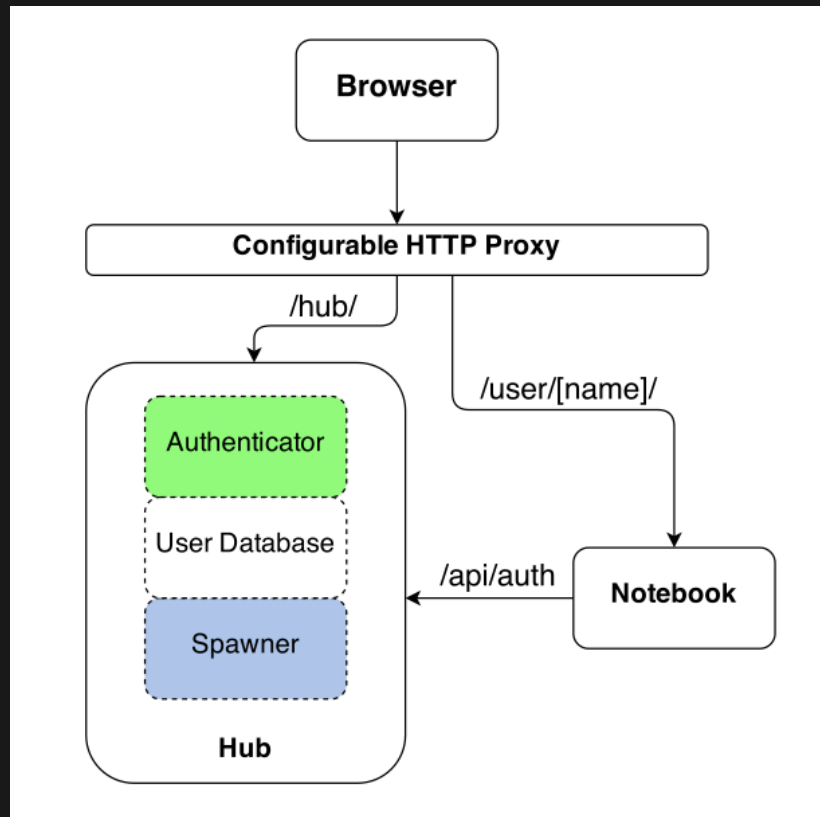
JUPYTER NOTEBOOKS ON HPC SYSTEMS

What is missing?

- Multi-user support
- Pure web-access to HPC resources



to the rescue...



- Manages authentication
- Spawns single-user servers on-demand
- Each user gets a complete notebook server

JUPYTERHUB @ NCAR

JupyterHub @ NCAR: Login

Assigned username on
Cheyenne Supercomputer

“CIT” password in conjunction
with the Duo Mobile app

or

Personal Identification Number
(PIN) in conjunction with a
Yubikey authentication token

Sign in

Username:

Password:

Sign In

JupyterHub @ NCAR: Specifying Job Configuration

Spawner Options

Job Name (-N)

Enter Queue or Reservation (-q)

Specify your project account (-A)

Specify N node(s) (-l select=N)

Specify N CPUs per node (-l ncpus=N)

Specify N MPI tasks per node (-l mpirs=N)

Specify N threads per process (-l ompthreads=N)

Specify wall time (-l walltime=HH:MM:SS) (12 Hr Maximum)

JupyterHub @ NCAR: A Running Jupyter Server


















JupyterLab

https://jupyterhub.ucar.edu/ch/user/abanihi/lab


















File Edit View Run Kernel Hub Tabs Settings Help

Launcher



Notebook

 Python 3	 Bash	 C++11	 C++14	 C++17	 Julia 1.0.3	 Matlab R20...	 Pangeo (Py...	 Python [con...	 Python [con...	 Python [con...
 Python [con...	 Python [con...	 Python [con...	 Python [con...	 R 3.5.1	 Xonsh [con...					

Console

 Python 3	 Bash	 C++11	 C++14	 C++17	 Julia 1.0.3	 Matlab R20...	 Pangeo (Py...	 Python [con...	 Python [con...	 Python [con...
 Python [con...	 Python [con...	 Python [con...	 Python [con...	 R 3.5.1	 Xonsh [con...					

Other

 Terminal	 Text File
---	--

JUPYTERHUB LIVE DEMO

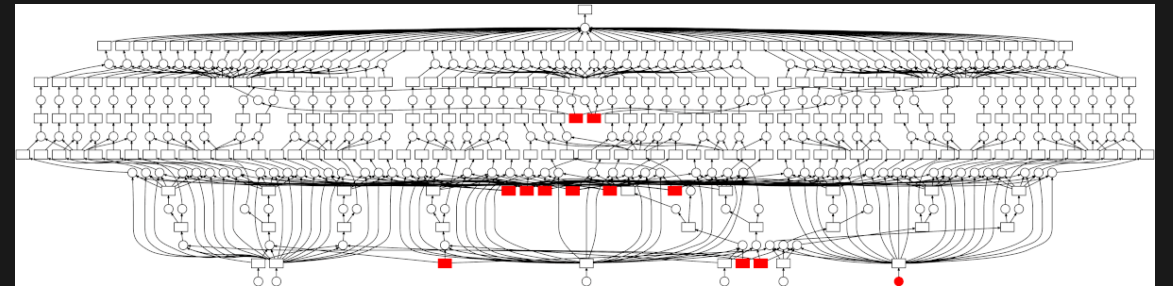
(if live demo gods are in a good mood...)

Accessing JupyterHub running on Cheyenne Supercomputer.





- Parallel programming library for Python
- Scales data libraries like Numpy, Pandas, Scikit-Learn, Xarray...
- Deploys on HPC systems
- Culturally native to Scientific Computing
- Provides schedulers for executing task graphs



DASK-JOBQUEUE

DASK-JOBQUEUE

- Easily deploy Dask on job queuing systems like PBS, Slurm, MOAB, SGE, and LSF, etc...
- Created as a spinoff of the Pangeo project.
- Pythonic user interface that manages dask workers/clusters

DASK-JOBQUEUE

```
from dask_jobqueue import PBSCluster
from distributed import Client
cluster = PBSCluster(project=..,
    queue=.., cores=1, processes=1,
    memory="100GB", walltime=...)
# Ask for 10 nodes
cluster.scale(10)
# OR scale adaptively based on load
cluster.adapt(minimum=1, maximum=100,
    wait_count=60)
# Connect to remote workers
client = Client(cluster)
```

Note: The cluster object stores a configuration for a block of worker nodes that you will be requesting...

DASK-JOBQUEUE

```
from dask_jobqueue import PBSCluster
from distributed import Client
cluster = PBSCluster(project=..,
    queue=.., cores=1, processes=1,
    memory="100GB", walltime=...)
# Ask for 10 nodes
cluster.scale(10)
# OR scale adaptively based on load
cluster.adapt(minimum=1, maximum=100,
    wait_count=60)
# Connect to remote workers
client = Client(cluster)
```

```
from dask_jobqueue import SLURMCluster
from distributed import Client
cluster = SLURMCluster(project=..,
    queue=.., cores=1, processes=1,
    memory="100GB", walltime=...)
# Ask for 10 nodes
cluster.scale(10)
# OR scale adaptively based on load
cluster.adapt(minimum=1, maximum=100,
    wait_count=60)
# Connect to remote workers
client = Client(cluster)
```

Note: The cluster object stores a configuration for a block of worker nodes that you will be requesting...

DASK-JOBQUEUE LIVE DEMO

(if live demo gods are in a good mood...)

Interactive Supercomputing with Dask-Jobqueue, Dask, an...



ADAPTIVE/ELASTIC SCALING, RESILIENCE, ETC...

ADAPTIVE/ELASTIC SCALING

Challenges:

- Balancing cluster resources and performance
 - is challenging
 - requires a lot of experimentation...
- Computational workloads fluctuate throughout the analysis...

ADAPTIVE/ELASTIC SCALING

Challenges:

- Balancing cluster resources and performance
 - is challenging
 - requires a lot of experimentation...
- Computational workloads fluctuate throughout the analysis...

Dask thinks about ...

- Scaling up and down
- Resilience
- Load balancing

ADAPTIVE/ELASTIC SCALING ON HPC SYSTEMS

Solution:

1. Start your Jupyter Notebook
2. Instantiate your dask cluster
3. Let dask determine when to scale up and/or down
4. **Do science**

ADAPTIVE/ELASTIC SCALING ON HPC SYSTEMS

Benefits:

- Adaptive scaling improves HPC systems' occupancy / utilization...
- Resilience against the death of all or part of computing resources provides new ways of leveraging job preemption...

ADAPTIVE/ELASTIC SCALING ON HPC SYSTEMS

Benefits:

- Adaptive scaling improves HPC systems' occupancy / utilization...
- Resilience against the death of all or part of computing resources provides new ways of leveraging job preemption...

Dask thinks about these benefits...

NOT ALL JOBS ARE INTERACTIVE

The screenshot shows the GitHub repository page for `dask/dask-mpi`. At the top, it displays repository statistics: 243 commits, 1 branch, 4 releases, 12 contributors, and a BSD-3-Clause license. Below this, there are navigation buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find File', and 'Clone or download'. A commit history table follows, listing recent changes with their descriptions and timestamps. The table includes entries for `.circleci`, `dask_mpi`, `docs`, `.coveragerc`, `.gitattributes`, `.gitignore`, `LICENSE.txt`, `MANIFEST.in`, `README.rst`, `environment-dev.yml`, `environment.yml`, `readthedocs.yml`, `setup.cfg`, `setup.py`, and `versioneer.py`.

The `README.rst` section is titled 'Deploying Dask using MPI4Py' and features a status bar with the following indicators: CHAT, ON GITTER, BUILD (PASSING), COVERAGE (94%), DOCS (PASSING), PYPI (V1.0.3), and CONDA-FORGE (V1.0.3). The text below the status bar reads: 'Easily deploy Dask Distributed in an existing MPI environment, such as one created with the `mpirun` or `mpiexec` MPI launch commands. See [documentation](#) for more details.'

File	Description	Time
<code>.circleci</code>	Add click & jupyter-server-proxy to dependencies	8 days ago
<code>dask_mpi</code>	Use Worker.close instead of the old Worker._close	8 days ago
<code>docs</code>	Pinning distributed until upstream issues are resolved	5 months ago
<code>.coveragerc</code>	Add list of files to omit from coverage	7 months ago
<code>.gitattributes</code>	Adding setup and versioneering	7 months ago
<code>.gitignore</code>	Adding Mac .DS_Store files to gitignores	7 months ago
<code>LICENSE.txt</code>	Adding license	7 months ago
<code>MANIFEST.in</code>	Update MANIFEST	6 months ago
<code>README.rst</code>	Add conda-forge badge to README	23 days ago
<code>environment-dev.yml</code>	Add click & jupyter-server-proxy to dependencies	8 days ago
<code>environment.yml</code>	Add click & jupyter-server-proxy to dependencies	8 days ago
<code>readthedocs.yml</code>	Adding readthedocs config	7 months ago
<code>setup.cfg</code>	Adding setup and versioneering	7 months ago
<code>setup.py</code>	update CI infrastructure	8 days ago
<code>versioneer.py</code>	Adding setup and versioneering	7 months ago

FUTURE

- Heterogeneous resources handling
- Coarse-Grained Diagnostics and History
- Scheduler Performance on Large Graphs

RESOURCES

- <https://jobqueue.dask.org/>
- <https://mpi.dask.org>
- Dask-jobqueue workshop materials
- Jupyter for Science User Facilities and High Performance Computing workshop

Participate

- <https://github.com/dask/dask-jobqueue/issues>
- <https://github.com/dask/dask-mpi/issues>

ACKNOWLEDGMENTS

- Jupyter/JupyterHub development teams
- NCAR/CISL Supercomputer Systems, Consulting Services Groups
- Pangeo collaborators