ENABLING TRANSFORMATIONAL SCIENCE

PANGEO DEPLOYMENT AT NCI

5th Feb 2021

Special thanks to all supporters:

NCRIS
National Research Infrastructure for Australia
An Australian Government Initiative

Australian Government
Bureau of Meteorology

CSIRO

UNIVERSITY of TASMANIA

Earth Systems and Climate Change Hub
National Environmental Science Programme

Australian Government
Geoscience Australia

NSW GOVERNMENT
Planning, Industry & Environment

AAPP
Australian Antarctic Program Partnership

ARC CENTRE OF EXCELLENCE FOR
CLIMATE EXTREMES

NCI AUSTRALIA

Agenda

Time	topic
11:00-11:20	Overview of Pangeo Deployment at NCI *(this presentation)*
11:20-11:40	Documentation and Live demo of setting up Pangeo *(this presentation)*
11:40-12:00	Q&A and 10min break
12:00-12:10	Introduction to VDI and NCI-data-analysis environment
12:00-12:30	Overview of Xarray and Dask
12:30-12:45	Live demo
12:45-1:00	Q&A and wrap up the workshop

NCI AUSTRALIA

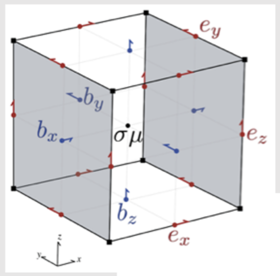
WHAT DRIVES PROGRESS IN (GEO)SCIENCE?

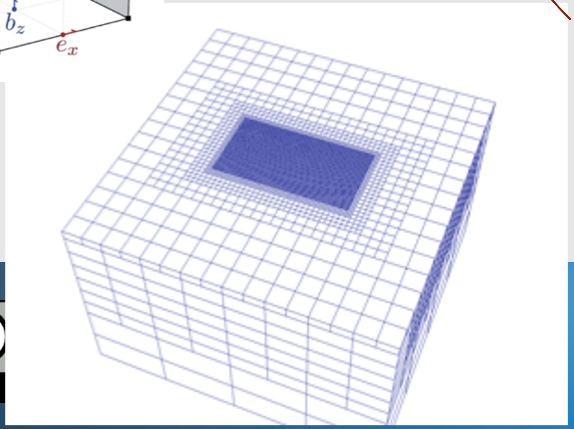
$$\left.\begin{aligned}\nabla \times \mathbf{e} &= -\frac{\partial \mathbf{b}}{\partial t}\\ \nabla \times \mathbf{h} &= \mathbf{j} + \frac{\partial \mathbf{d}}{\partial t}\end{aligned}\right\}$$

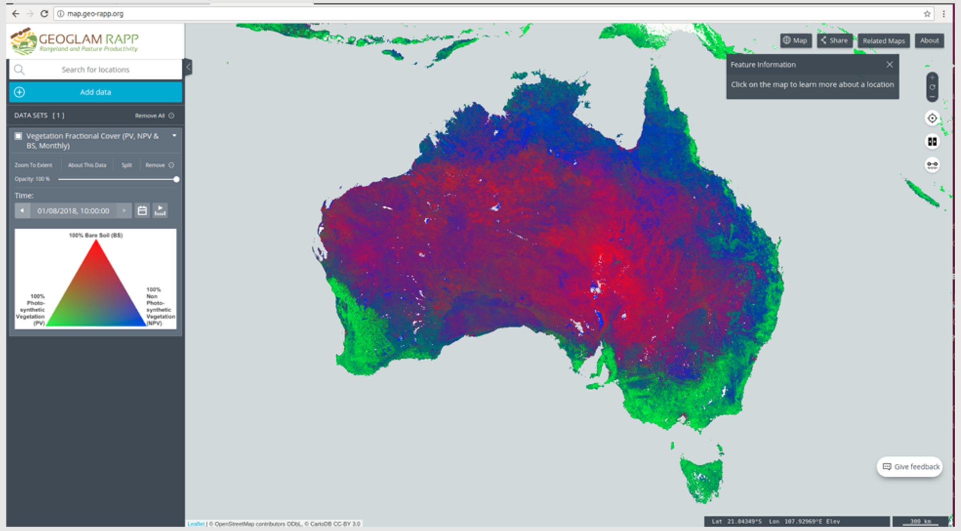
Theory & Ideas

Observations / Data

Simulations, Computation



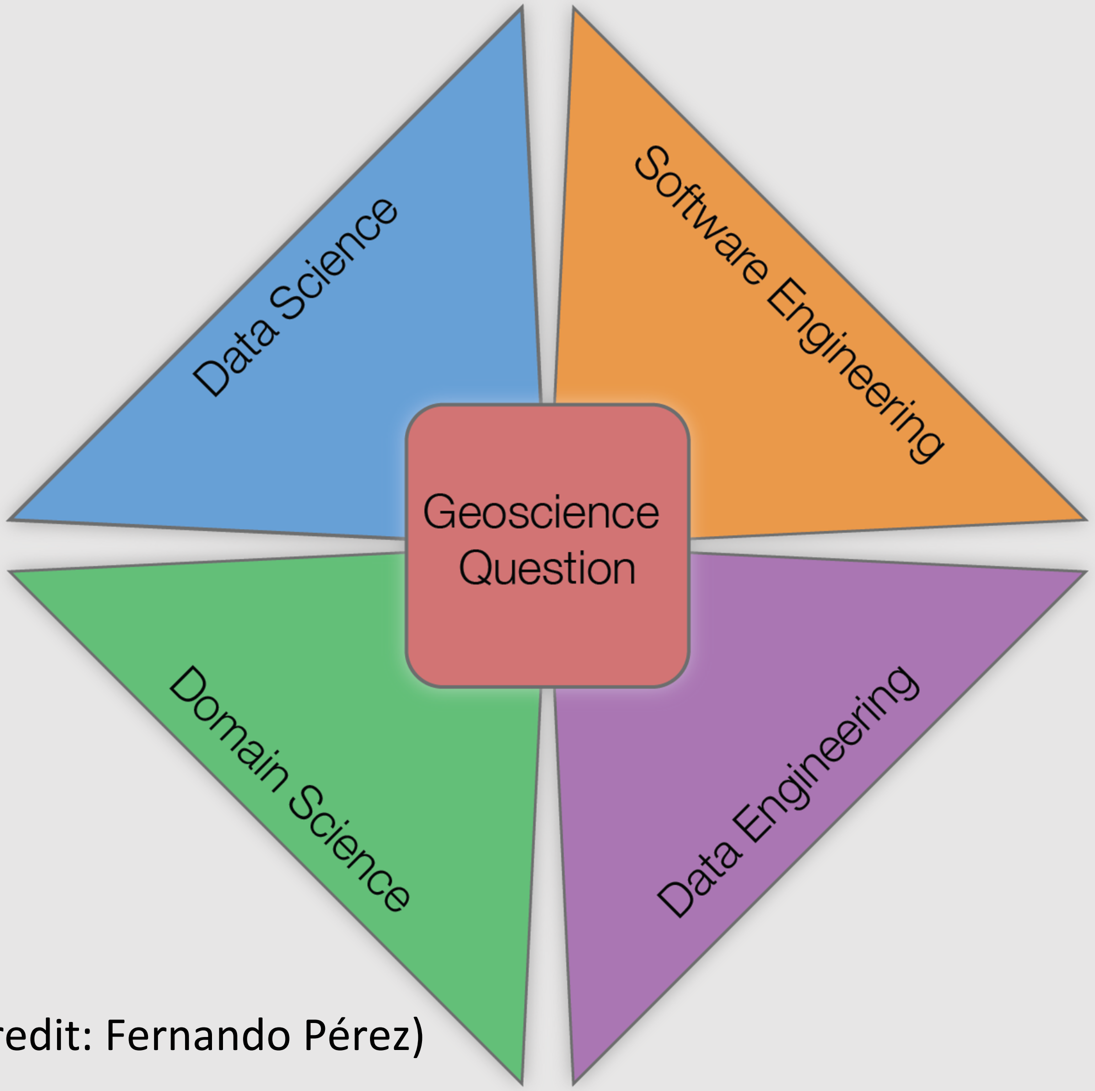




(MODIS Fractional Cover loaded on the GEOGLAM RAPP map)

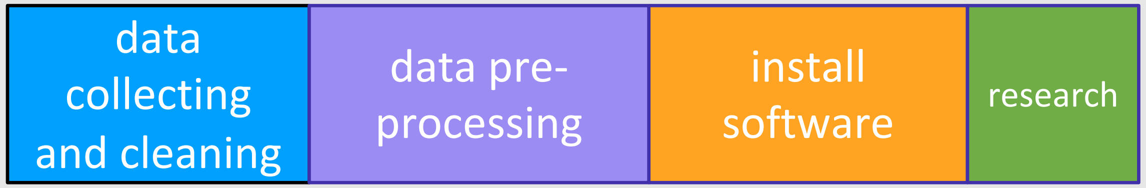
(Credit: Fernando Pérez)

ULTIMATE GOAL: REALLOCATE TIME!

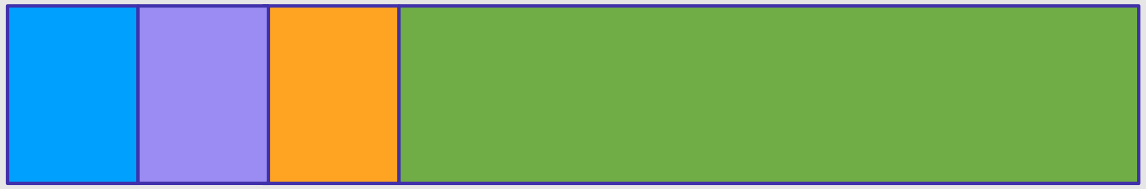


(credit: Fernando Pérez)

Traditional Way



With data, software and compute available…



NCI AUSTRALIA

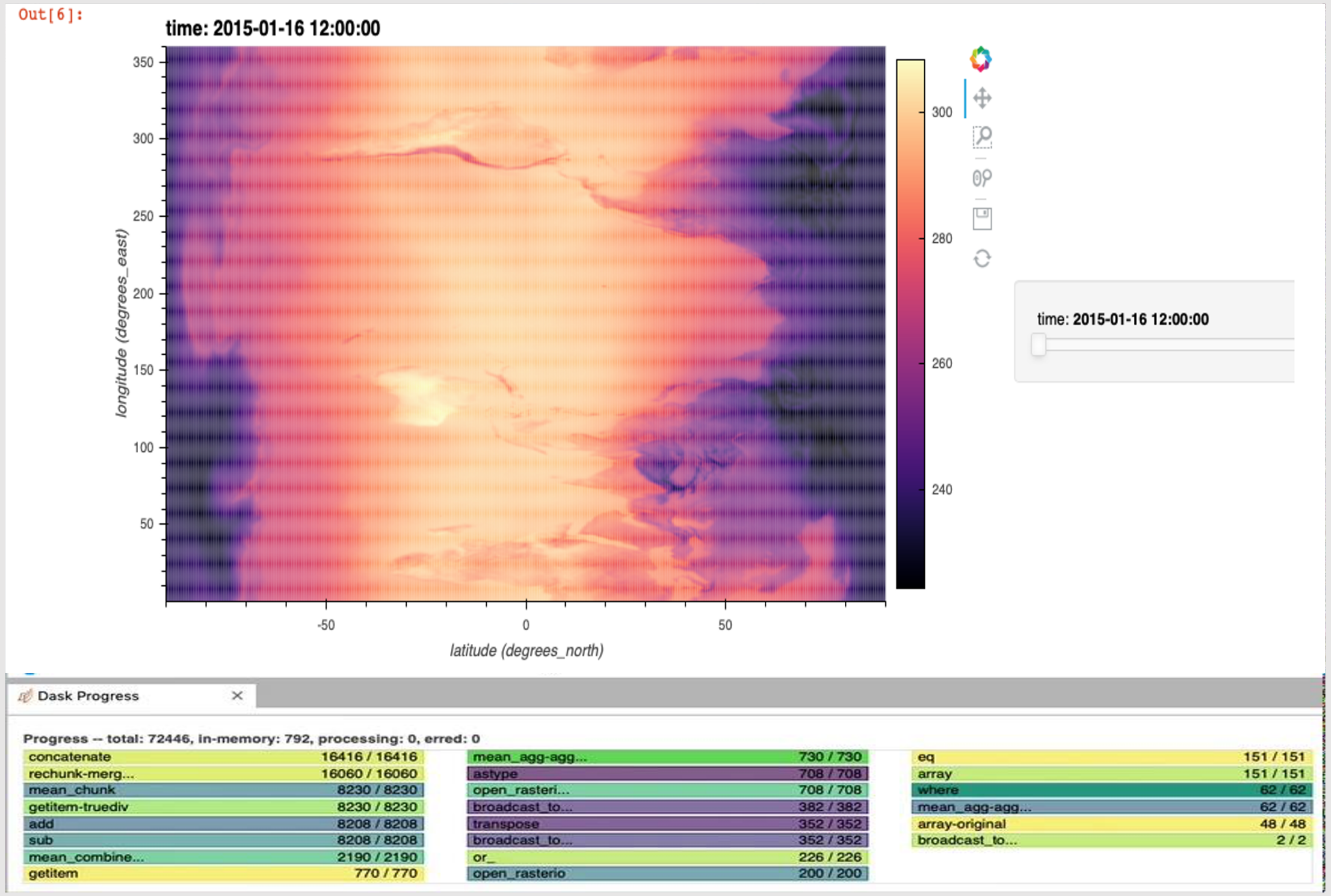
PANGEO

Harnessing the power of cloud computing to study the whole Earth interactively

jupyter Interactivity

DASK Distributed computing

xarray Data models / numeric



```
Progress -- total: 72446, in-memory: 792, processing: 0, erred: 0
concatenate        16416 / 16416    mean_agg-agg...     730 / 730    eq                151 / 151
rechunk-merg...    16060 / 16060    astype              708 / 708    array             151 / 151
mean_chunk          8230 / 8230     open_rasteri...     708 / 708    where              62 / 62
getitem-truediv     8230 / 8230     broadcast_to...     382 / 382    mean_agg-agg...    62 / 62
add                 8208 / 8208     transpose           352 / 352    array-original     48 / 48
sub                 8208 / 8208     broadcast_to...     352 / 352    broadcast_to...     2 / 2
mean_combine...     2190 / 2190     or_                 226 / 226
getitem              770 / 770      open_rasterio       200 / 200
```

PANGEO DEPLOYMENT AT GADI: AN ACCELERATION TOOL

Research use-cases

- Climate data analysis
- Weather data analysis
- Earth observation
- Geophysics
- Natural Hazards

+

Tech developments

- Libraries and Tools
- Interactivity
- Cloud/HPC infrastructure
- Direct data access

Pangeo – a big-data analysis platform at scale

Learning Goals for this presentation:

- Provide an overview of the Pangeo ecosystems
- Announce the data analysis environment
- Point to the documentation space
- Demonstrate data analysis examples

What is Pangeo?

“Pangeo is first and foremost a *community* promoting open, reproducible, and scalable science.” (http://pangeo.io)

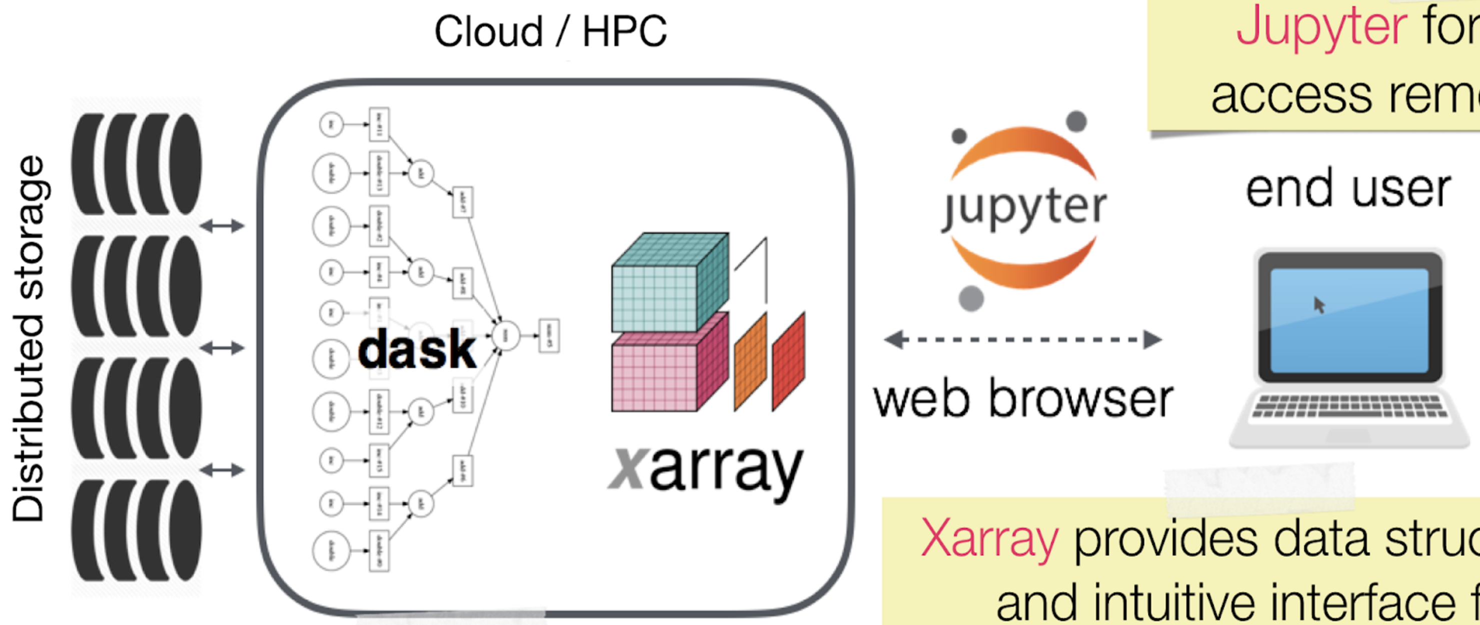
PANGEO

A community platform for Big Data geoscience

Why do we need Pangeo?

Need better tools to support exploration of increasingly large data volumes

PANGEO ARCHITECTURE



"Analysis Ready Data" stored on globally-available distributed storage.

Jupyter for interactive access remote systems

Xarray provides data structures and intuitive interface for interacting with datasets

Parallel computing system allows users deploy clusters of compute nodes for data processing.

Dask tells the nodes what to do.

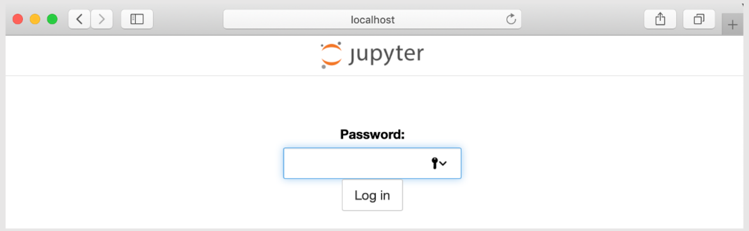
(http://pangeo.io)

Step1: submit a job on Gadi to spin up a Jupyterlab server

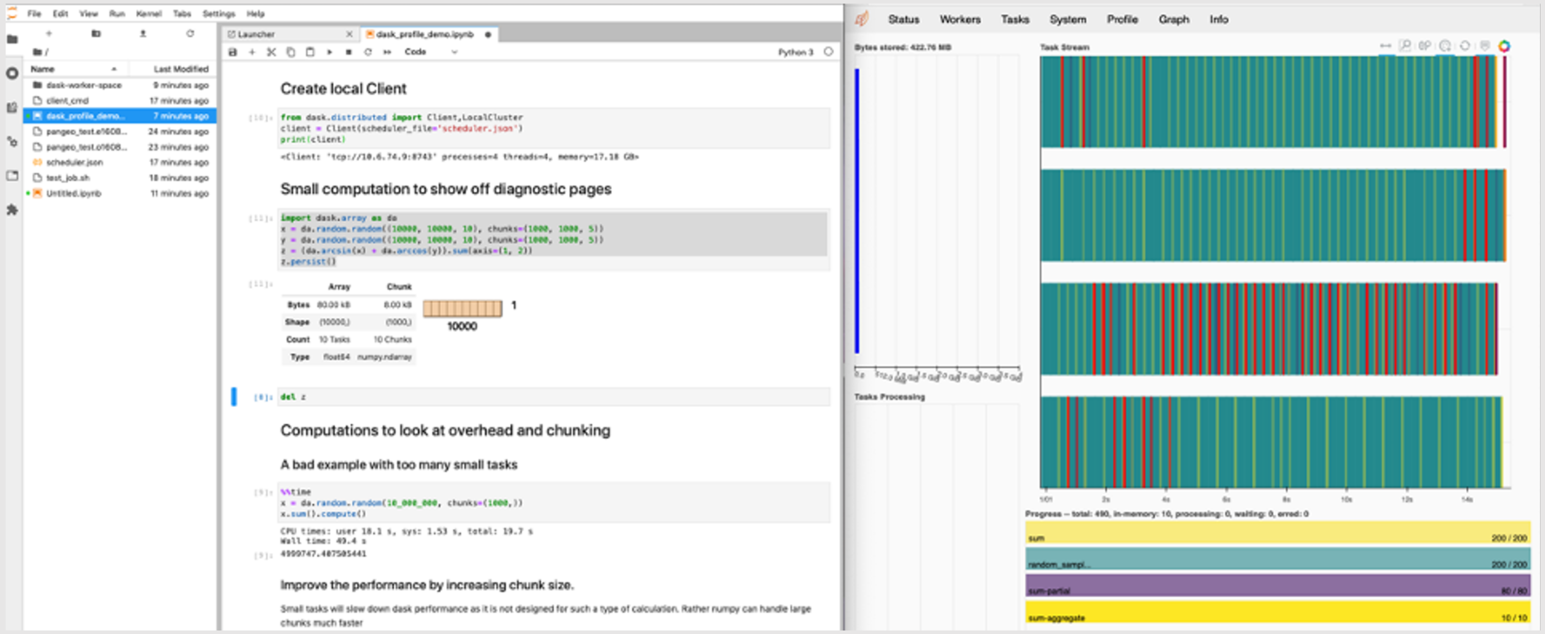
```
#!/bin/bash
#PBS -N pangeo_test
#PBS -P <project code>
#PBS -q normal
#PBS -l walltime=5:00:00
#PBS -l ncpus=96
#PBS -l mem=384GB
#PBS -l jobfs=100GB
#PBS -l storage=scratch/<project code>+gdata/<project code>
module use /g/data/dk92/apps/Modules/modulefiles
module load pangeo
module load NCI-data-analysis/2020.12
pangeo.ini.all.sh
sleep infinity
```

Step2: ssh Gadi and open the ports on web browser (two tabs)

```
$ ssh -N -L 8388:gadi-cpu-clx-2224.gadi.nci.org.au:8388 abc123@gadi.nci.org.au &
```



Step3: open the JupyterLab app and the dask dashboard



Create local Client

```
[10]: from dask.distributed import Client,LocalCluster
      client = Client(scheduler_file='scheduler.json')
      print(client)

<Client: 'tcp://10.6.74.9:8743' processes=4 threads=4, memory=17.18 GB>
```

Small computation to show off diagnostic pages

```
[11]: import dask.array as da
      x = da.random.random((10000, 10000, 10), chunks=(1000, 1000, 5))
      y = da.random.random((10000, 10000, 10), chunks=(1000, 1000, 5))
      z = (da.arcsin(x) + da.arccos(y)).sum(axis=(1, 2))
      z.persist()
```

[11]:

	Array	Chunk
Bytes	80.00 kB	8.00 kB
Shape	(10000,)	(1000,)
Count	10 Tasks	10 Chunks
Type	float64	numpy.ndarray

```
[8]: del z
```

Computations to look at overhead and chunking

A bad example with too many small tasks

```
[5]: %%time
     x = da.random.random(10_000_000, chunks=(1000,))
     x.sum().compute()

CPU times: user 18.1 s, sys: 1.53 s, total: 19.7 s
Wall time: 49.4 s
[5]: 4999747.407505441
```

Improve the performance by increasing chunk size.

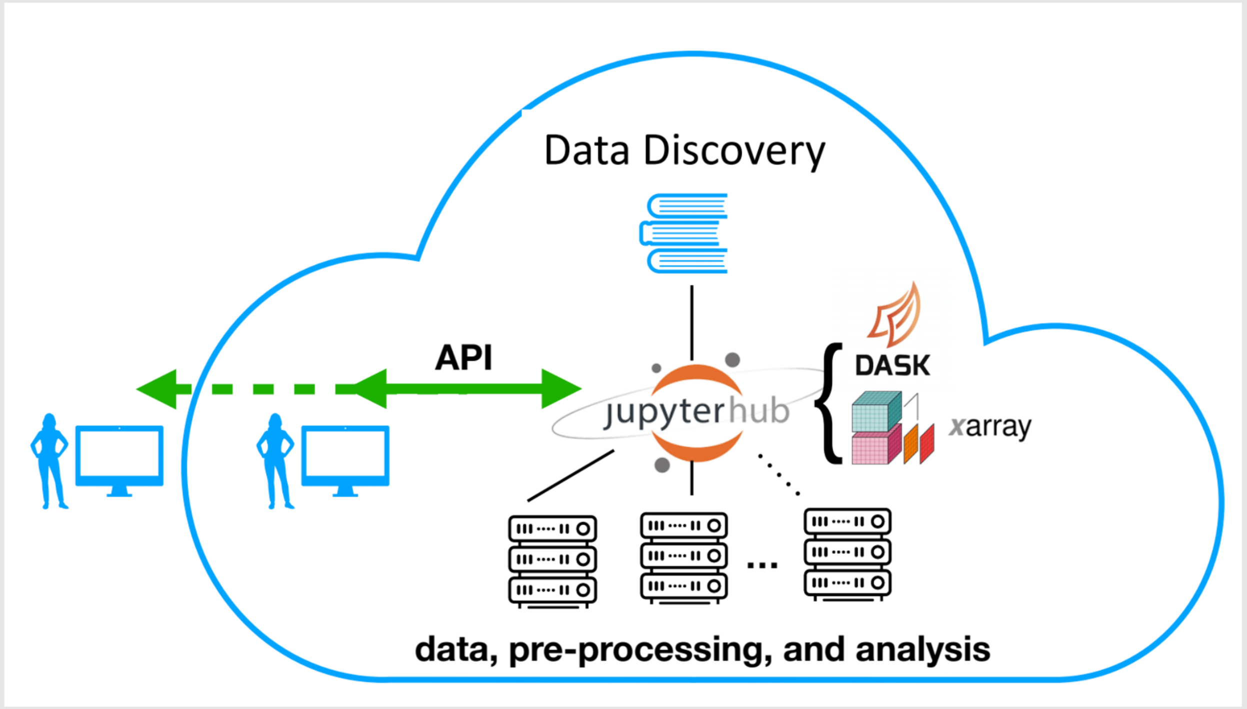
Small tasks will slow down dask performance as it is not designed for such a type of calculation. Rather numpy can handle large chunks much faster

Scalable –
You can submit PBS jobs for the intensive compute parts of your workflow inside a notebook

```
import dask.config
from dask.distributed import Client,LocalCluster
from dask_jobqueue import PBSCluster
walltime = '01:00:00'
cores = 96
memory = '160GB'

cluster = PBSCluster(walltime=str(walltime), cores=cores, memory=str(memory),
            job_extra=['-P <project code>','-l ncpus='+str(cores),'-l mem='+str(memory),
                  '-l storage=gdata/<project code>+gdata/<project code>+gdata/<project code>...'],
            header_skip=["select"],python=os.environ["DASK_PBS_PYTHON"])
cluster.scale(jobs=2)
client = Client(cluster)
client
```

Pangeo/Jupyter could be one of the essential interfaces to NCI's resources:

Move analysis to the data



Instant access to compute resources and data via PBS queueing system

Democratize access large computations accessed with **web browser**

No downloading data

Scalable computational power used and billed by time

On-demand special resources (GPUs)

Reproducible workflows thanks to network-accessible datasets and containerized software

(Credit: Scott Henderson, University of Washington)

NCI AUSTRALIA

Things to be mindful of when using Pangeo

As the compute nodes do not allow internet access, libraries that download things on the fly won't work. For example, Cartopy might need to download a coastline database. In this case, you need to download the database onto Gadi, then point to the access point in your workflow.

Balance the resources requested for serial or parallel jobs. You can manage this by requesting minimum resource to get Pangeo started, then submit computationally intensive jobs inside your workflow to set up a Dask cluster as needed.

Running notebooks on VDI vs using Pangeo on Gadi

Pangeo on Gadi can utilise Dask or xarray for parallel computing and data processing. However, this will likely require more resources such as CPU, memory and I/O throughput.

Note: Pangeo jobs should be submitted to the queue system and we recommend you request node-based resources.

Don't waste your SUs!

The NCI-provided Virtual Desktop Infrastructure (VDI) is useful for:

- executing lightweight, serial/parallel Jupyter notebooks or Python scripts without consuming SUs
- developing scripts for the Pangeo environment

For more information on the VDI, see https://opus.nci.org.au/display/Help/VDI+User+Guide

WHAT'S NEXT ?

- Deploy Pangeo on a Cloud infrastructure

Live demo

- How to set up Pangeo on Gadi

https://opus.nci.org.au/display/Help/5.1+Set+up+Pangeo

Step 1. Enabling Pangeo in your shell environment

To enable the Pangeo environment, you can use the following command within jobs, or within an interactive environment:

```
$ module load pangeo/<version>
            Loading pangeo/2010.10
                  Loading requirement: intel-mkl/2019.3.199  python3/3.7.4   hdf5/1.10.5
netcdf/4.7.4
```

Step 2. Configure your account on Gadi (once-only)

JupyterLab is bundled within the Pangeo environment and will be used to load notebooks and monitor jobs. To set up this environment, run the following two commands:

```
$ jupyter notebook --generate-config

$ jupyter notebook password
```

This is a stand-alone password that you will use later for accessing the JupyterLab server. It has nothing to do with your NCI login account and password. You can use this command to reset your password at any time.

Step 3. Submitting a multi-node Pangeo job to Gadi

First create a directory where you will run the Jupyter notebook:

```
$ mkdir -p ~/pangeo/tutorial
```

Next you need to create a PBS shell script that will be used to launch a multi-node Pangeo job:

```
#!/bin/bash              ### see https://opus.nci.org.au/display/Help/PBS+Directives+Explained for explanation on PBS directives
#PBS -N pangeo_test
#PBS -P <project code>
#PBS -q normal           ### see https://opus.nci.org.au/display/Help/Queue+Limits for Gadi Queue Limits
#PBS -l walltime=5:00:00
#PBS -l ncpus=96
#PBS -l mem=384GB
#PBS -l jobfs=100GB
#PBS -l storage=scratch/<project code>+gdata/<project code>

module load pangeo
pangeo.ini.all.sh
sleep infinity
```

Step 3. Submitting a multi-node Pangeo job to Gadi

```
#!/bin/bash
#PBS -N pangeo_test
#PBS -P <project code>
#PBS -q normal
#PBS -l walltime=5:00:00
#PBS -l ncpus=96
#PBS -l mem=384GB
#PBS -l jobfs=100GB
#PBS -l storage=scratch/<project code>+gdata/<project code>

module load pangeo
pangeo.ini.all.sh
sleep infinity
```

Note that for Gadi, the “#PBS -l storage=” flag is required which must include **all** the scratch and gdata project IDs that will be used in your codes.

Dask requires whole nodes so jobs beyond a single node must use multiples of 48 for their ncpus request.

Step 3. Submitting a multi-node Pangeo job to Gadi

To submit the job to the queue:

```
$ qsub run_ipynb_job.sh
```

and take note of your job_id (which may look something like 1030967.gadi-pbs).

Check to see when the job is running:

```
$ qstat <job_id>
```

Step 3. Submitting a multi-node Pangeo job to Gadi

To ease the user experience of software installation, NCI provides a versioned virtual environment which includes a number of Python packages for data processing and visualisation. They are managed within project dk92. To use this virtual environment:

```
$ module use /g/data/dk92/apps/Modules/modulefiles
$ module load NCI-data-analysis/2020.12
```

A combination of Pangeo and NCI data-analysis virtual environments reduces the hassle of installing software and solving dependencies, and provides an interactive working environment for using JupyterLab. To use both Pangeo and this virtual environment, just add the above two lines to your PBS job submission script.

Step 3. Submitting a multi-node Pangeo job to Gadi

```
#!/bin/bash
#PBS -N pangeo_test
#PBS -P <project code>
#PBS -q normal
#PBS -l walltime=5:00:00
#PBS -l ncpus=96
#PBS -l mem=384GB
#PBS -l jobfs=100GB
#PBS -l storage=scratch/<project code>+gdata/<project code>

module use /g/data/dk92/apps/Modules/modulefiles
module load pangeo
module load NCI-data-analysis/2020.12
pangeo.ini.all.sh
sleep infinity
```

Step 4. Set up port forwarding to access from your desktop machine

Once the job is running on Gadi, there will be two files that are created in your current Gadi directory:

- cliend_cmd
- scheduler.json

The file client_cmd contains commands to forward network traffic from the defined port number of the worker node to an external client machine (i.e., your desktop) via the login node gadi.nci.org.au

Step 4. Set up port forwarding to access from your desktop machine

Running:

```
$ more client_cmd
```

should give an output that looks something like:

```
ssh  -N  -L  8388:gadi-cpu-clx-2224.gadi.nci.org.au:8388  abc123@gadi.nci.org.au
ssh  -N  -L  8768:gadi-cpu-clx-2224.gadi.nci.org.au:8768  abc123@gadi.nci.org.au
```

For this example, JupyterLab uses port 8388 and Dask dashboard occupies port 8768 respectively from Gadi worker node gadi-cpu-clx-2224.

Note that both port numbers are randomly chosen for each job and will be different each time you run a new job.

Step 4. Set up port forwarding to access from your desktop machine

Next, on your remote machine (e.g., Desktop), run each of the following commands separately if you have already set up SSH keys:

```
$ ssh -N -L 8388:gadi-cpu-clx-2224.gadi.nci.org.au:8388  abc123@gadi.nci.org.au &
```

```
$ ssh -N -L 8768:gadi-cpu-clx-2224.gadi.nci.org.au:8768  abc123@gadi.nci.org.au &
```

If you haven't set up SSH keys, instructions on how to do this can be found at https://opus.nci.org.au/display/Help/Using+SSH+keys

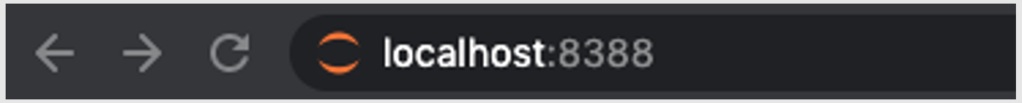
Alternatively, you should enter your Gadi password when prompted.

Step 5. Connect to the remote JupyterLab server from your remote desktop computer

In a browser on your local desktop, type in the following URL:

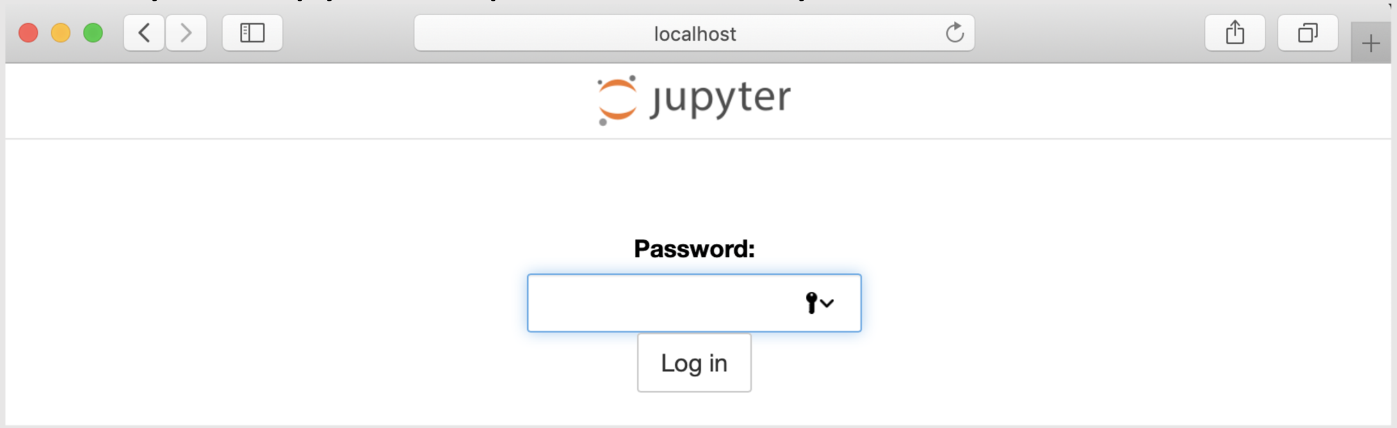
    "localhost:xxxx"

where xxxx is the port number from your client_cmd file



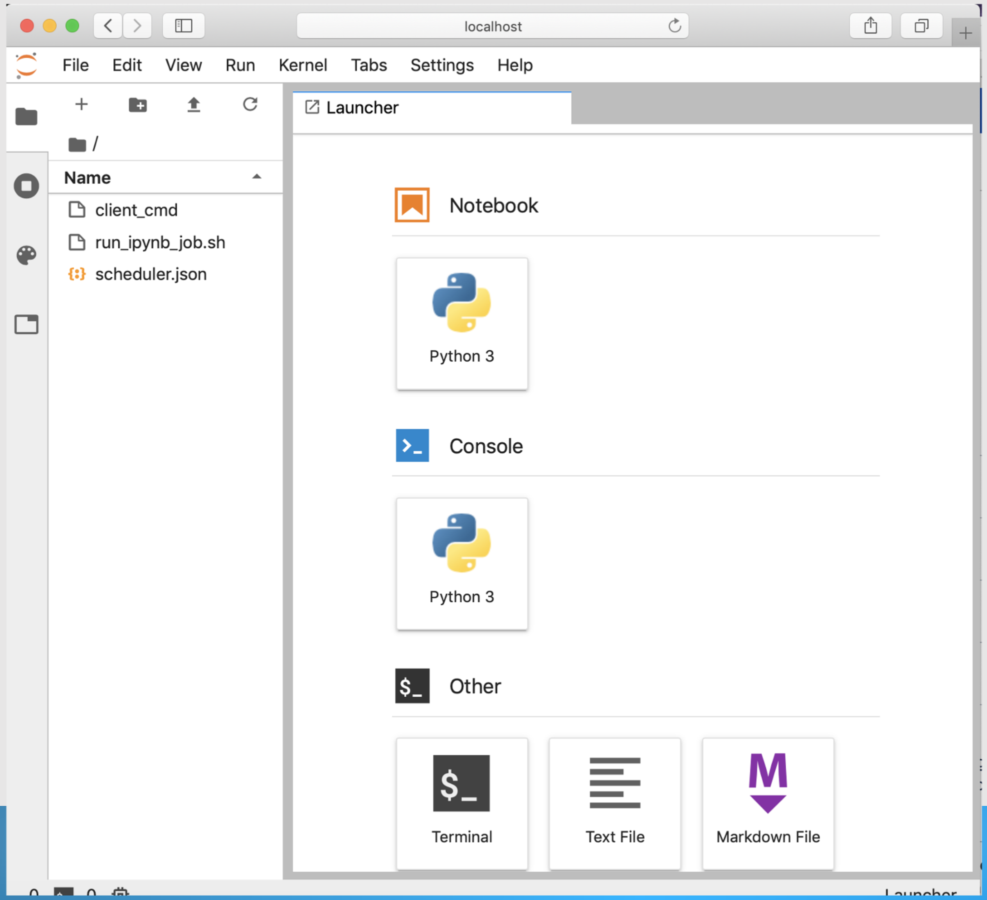
localhost:8388

You will be prompted for your JupyterLab password that you created earlier:



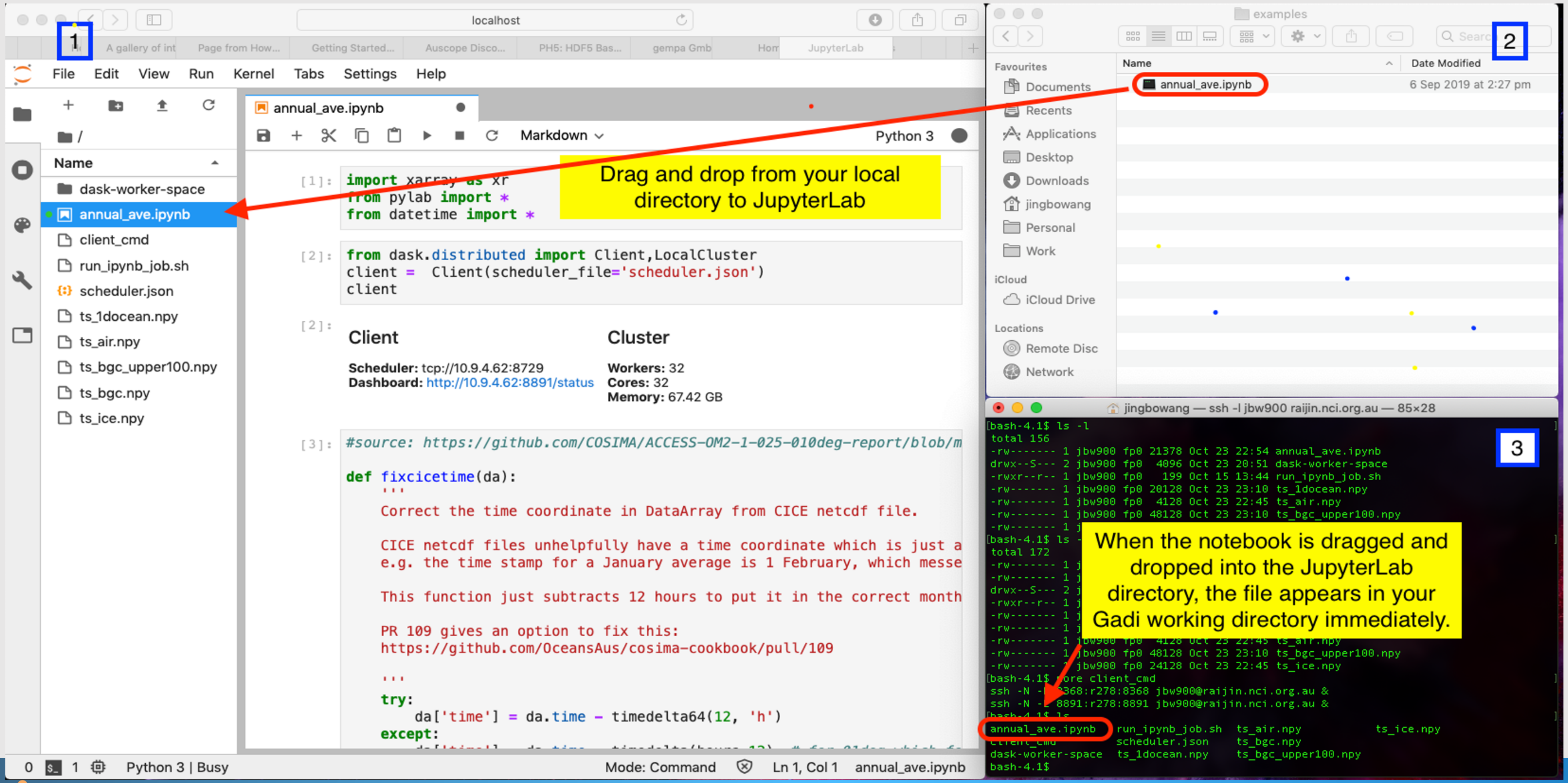
Step 5. Connect to the remote JupyterLab server from your remote desktop computer

Once your authentication has passed, a JupyterLab interface will be launched in a few seconds:



Step 6. Importing a notebook example

You can drag and drop a notebook from your local machine into the JupyterLab. This file will also appear in your working directory on Gadi:



```
[1]: import xarray as xr
     from pylab import *
     from datetime import *

[2]: from dask.distributed import Client,LocalCluster
     client =  Client(scheduler_file='scheduler.json')
     client
```

[2]:

Client

Scheduler: tcp://10.9.4.62:8729
Dashboard: http://10.9.4.62:8891/status

Cluster

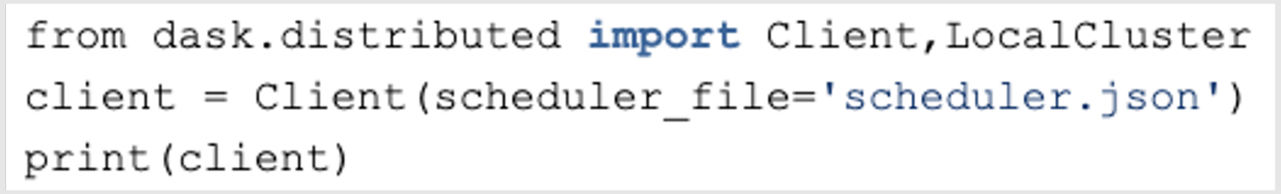
Workers: 32
Cores: 32
Memory: 67.42 GB

```
[3]: #source: https://github.com/COSIMA/ACCESS-OM2-1-025-010deg-report/blob/m

     def fixcicetime(da):
         '''
         Correct the time coordinate in DataArray from CICE netcdf file.

         CICE netcdf files unhelpfully have a time coordinate which is just a
         e.g. the time stamp for a January average is 1 February, which messe

         This function just subtracts 12 hours to put it in the correct month

         PR 109 gives an option to fix this:
         https://github.com/OceansAus/cosima-cookbook/pull/109

         '''
         try:
             da['time'] = da.time - timedelta64(12, 'h')
         except:
```

```
jingbowang — ssh -l jbw900 raijin.nci.org.au — 85×28
bash-4.1$ ls -l
total 156
-rw------- 1 jbw900 fp0 21378 Oct 23 22:54 annual_ave.ipynb
drwx--S--- 2 jbw900 fp0  4096 Oct 23 20:51 dask-worker-space
-rwxr--r-- 1 jbw900 fp0   199 Oct 15 13:44 run_ipynb_job.sh
-rw------- 1 jbw900 fp0 20128 Oct 23 23:10 ts_1docean.npy
-rw------- 1 jbw900 fp0  4128 Oct 23 22:45 ts_air.npy
-rw------- 1 jbw900 fp0 48128 Oct 23 23:10 ts_bgc_upper100.npy
-rw------- 1 j
bash-4.1$ ls
total 172
-rw------- 1 j
-rw------- 1 j
drwx--S--- 2 j
-rwxr--r-- 1 j
-rw------- 1 j
-rw------- 1 j
-rw------- 1 jbw900 fp0  4128 Oct 23 22:45 ts_air.npy
-rw------- 1 jbw900 fp0 48128 Oct 23 23:10 ts_bgc_upper100.npy
-rw------- 1 jbw900 fp0 24128 Oct 23 22:45 ts_ice.npy
bash-4.1$ more client_cmd
ssh -N -L 8368:r278:8368 jbw900@raijin.nci.org.au &
ssh -N -L 8891:r278:8891 jbw900@raijin.nci.org.au &
bash-4.1$ ls
annual_ave.ipynb  run_ipynb_job.sh  ts_air.npy           ts_ice.npy
client_cmd        scheduler.json    ts_bgc.npy
dask-worker-space ts_1docean.npy    ts_bgc_upper100.npy
bash-4.1$
```

Step 7. Setting up your Jupyter notebook to use the Dask server

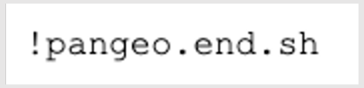
To use the dask server established from the PBS job, it is necessary to add and run the following cell at the beginning of your notebook:



```
from dask.distributed import Client,LocalCluster
client = Client(scheduler_file='scheduler.json')
print(client)
```

The output will show the configuration of the client and Dask cluster. You can check that the number of cores matches what you requested in the job script. Now you can run your notebook as per usual.

To gracefully stop the PBS job:

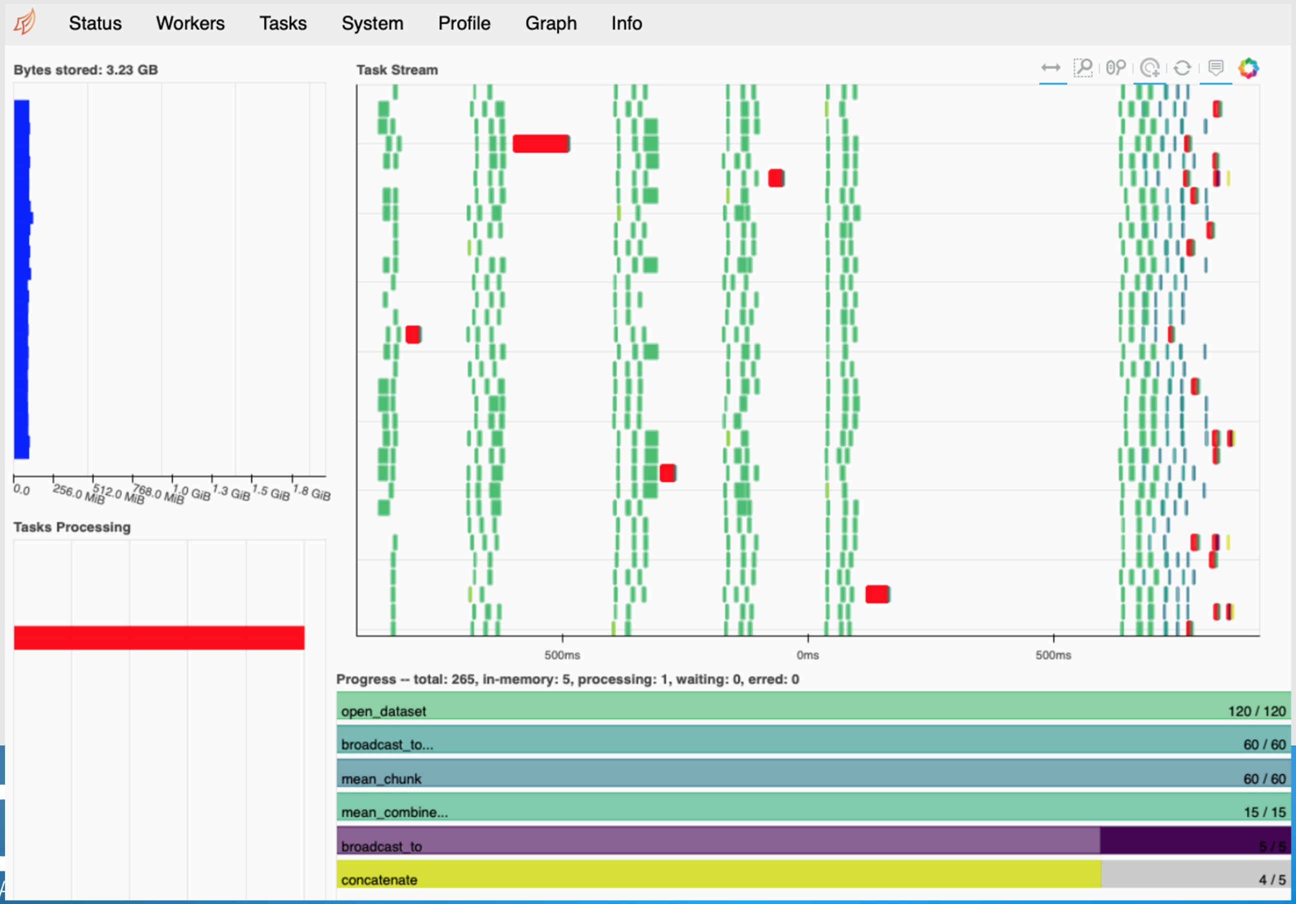


```
!pangeo.end.sh
```

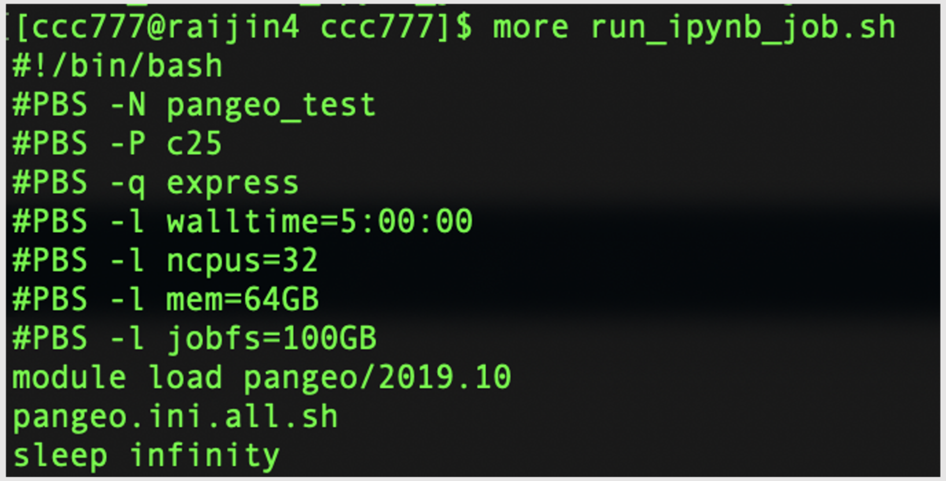
Step 7. Setting up your Jupyter notebook to use the Dask server

View compute threads using Dask dashboard

From your local desktop, open a new tab in your web browser and type the second port in the client_cmd file to open the Dask dashboard (e.g. localhost:8768). This will allow you to see the dynamic resources of the processing.



```
Progress -- total: 265, in-memory: 5, processing: 1, waiting: 0, erred: 0
open_dataset      120 / 120
broadcast_to...     60 / 60
mean_chunk          60 / 60
mean_combine...     15 / 15
broadcast_to         5 / 5
concatenate          4 / 5
```



```
[ccc777@raijin4 ccc777]$ more run_ipynb_job.sh
#!/bin/bash
#PBS -N pangeo_test
#PBS -P c25
#PBS -q express
#PBS -l walltime=5:00:00
#PBS -l ncpus=32
#PBS -l mem=64GB
#PBS -l jobfs=100GB
module load pangeo/2019.10
pangeo.ini.all.sh
sleep infinity
```

- Make sure your project has enough kSU allocated to complete the job
- For the queue selection (-q), *normal* is recommended if the job is not urgent
- *Walltime* provides the limit of the job run, so make sure you specify enough walltime to run a computationally expensive job
- Watch for the new version of Pangeo in three months time

Add these lines into your notebook or python script!!

```
# start the dask client
 from dask.distributed import
 Client,LocalCluster
  client =
Client(scheduler_file='scheduler.json')

… your work utilizing xarray&dask

# stop the pbs job.
! pangeo.end.sh
```