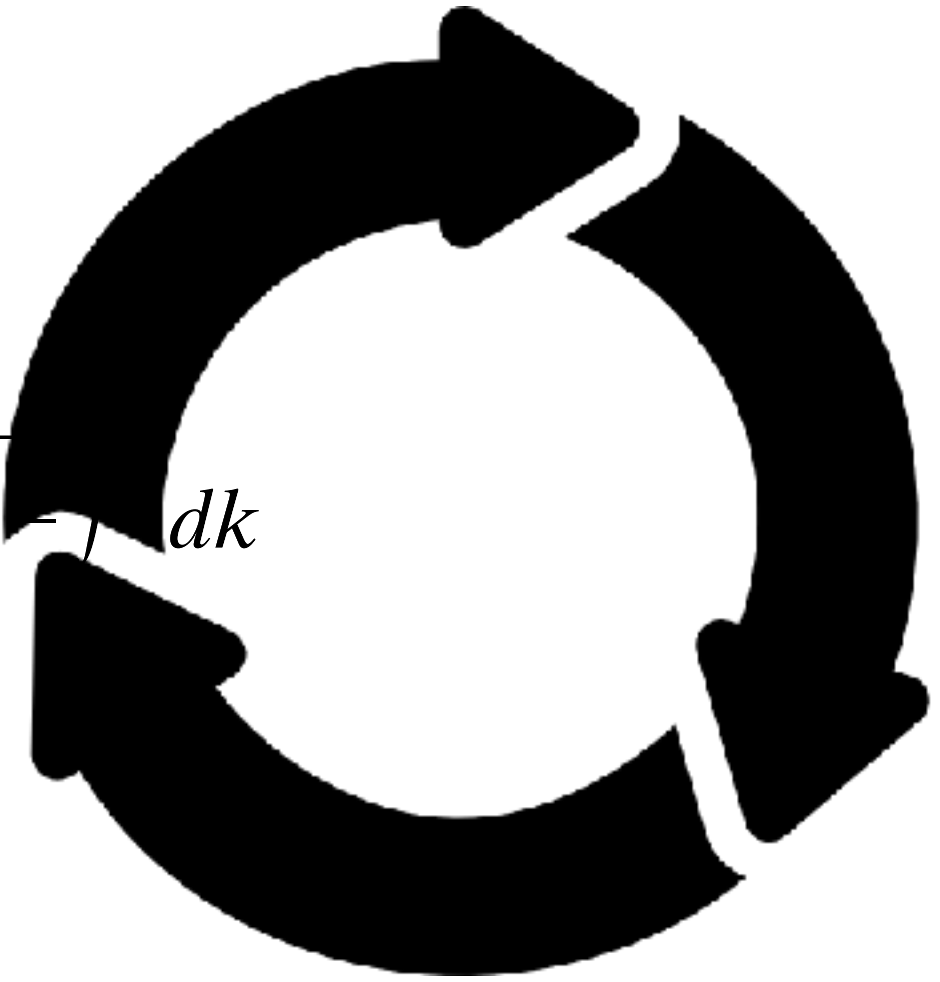
PANGEO

A COMMUNITY-DRIVEN EFFORT FOR
BIG DATA GEOSCIENCE

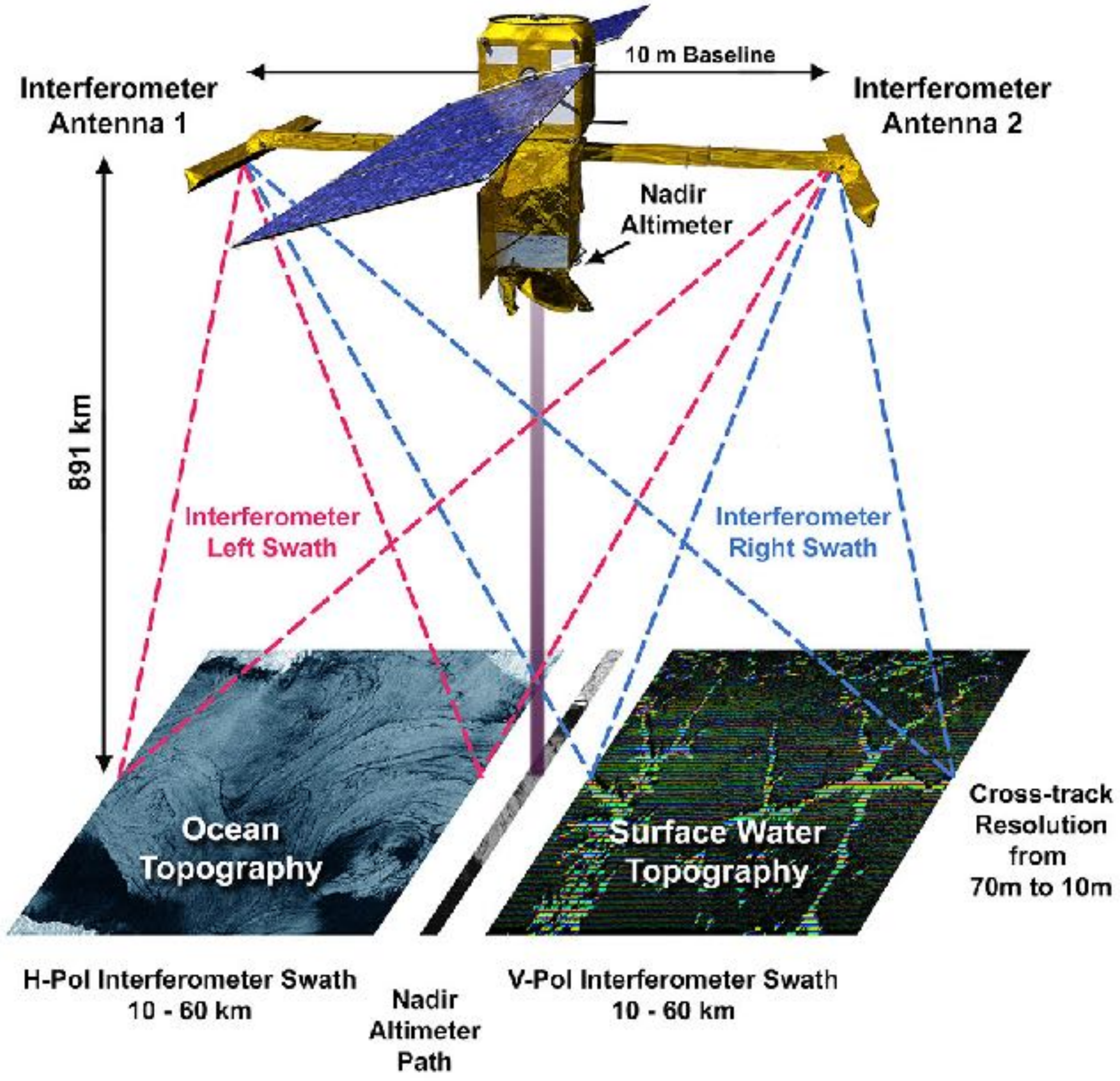
WHAT DRIVES PROGRESS IN OCEANOGRAPHY?

New Ideas

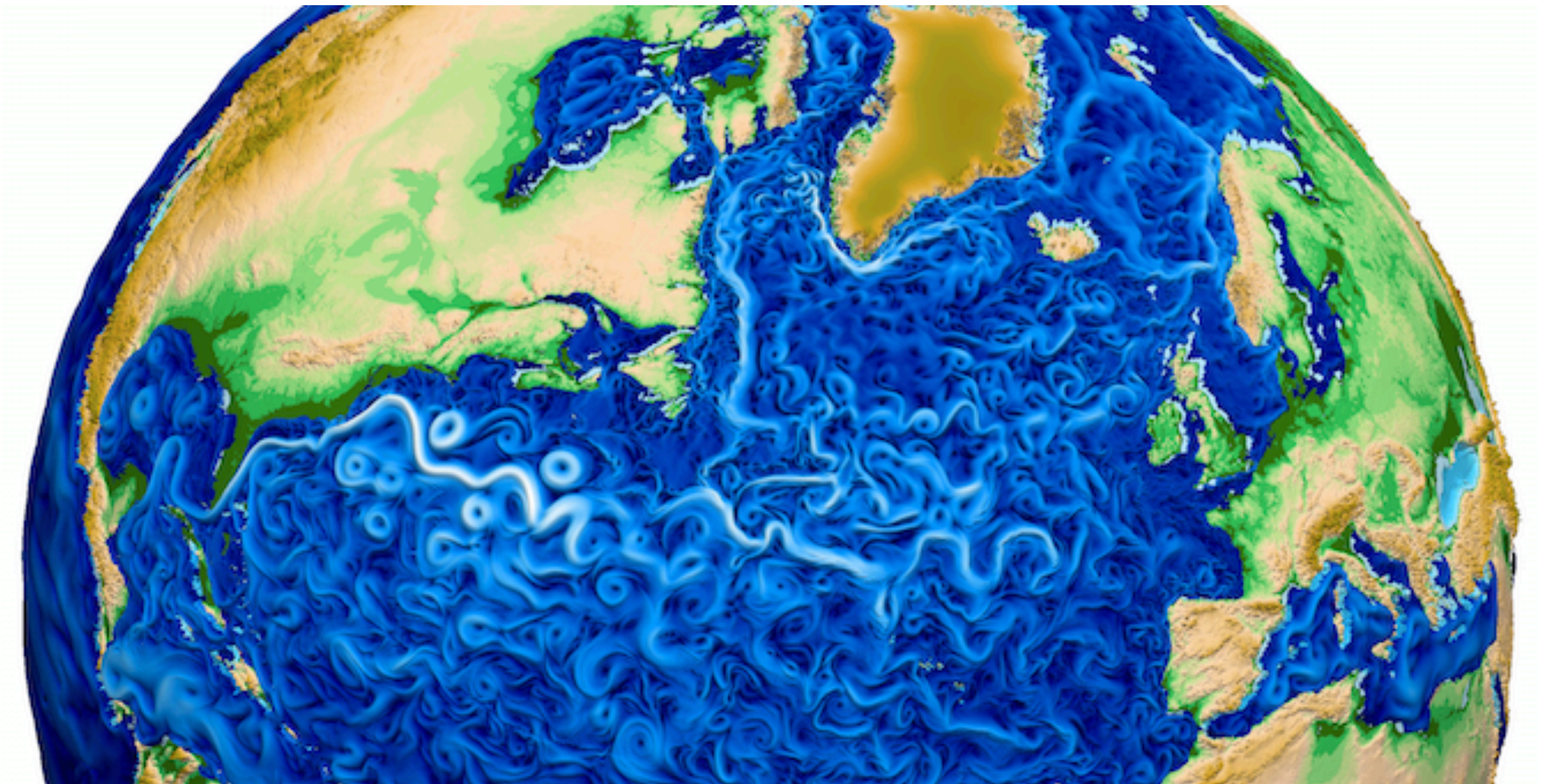
$$E = \frac{\rho_0 |\mathbf{U}|}{\pi} \int_{|f|/|\mathbf{U}|}^{N/|\mathbf{U}|} P_{1\mathrm{D}}(k) \sqrt{N^2 - |\mathbf{U}|^2 k^2} \sqrt{|\mathbf{U}|^2 k^2 - f^2}\, dk$$



New Observations



New Simulations

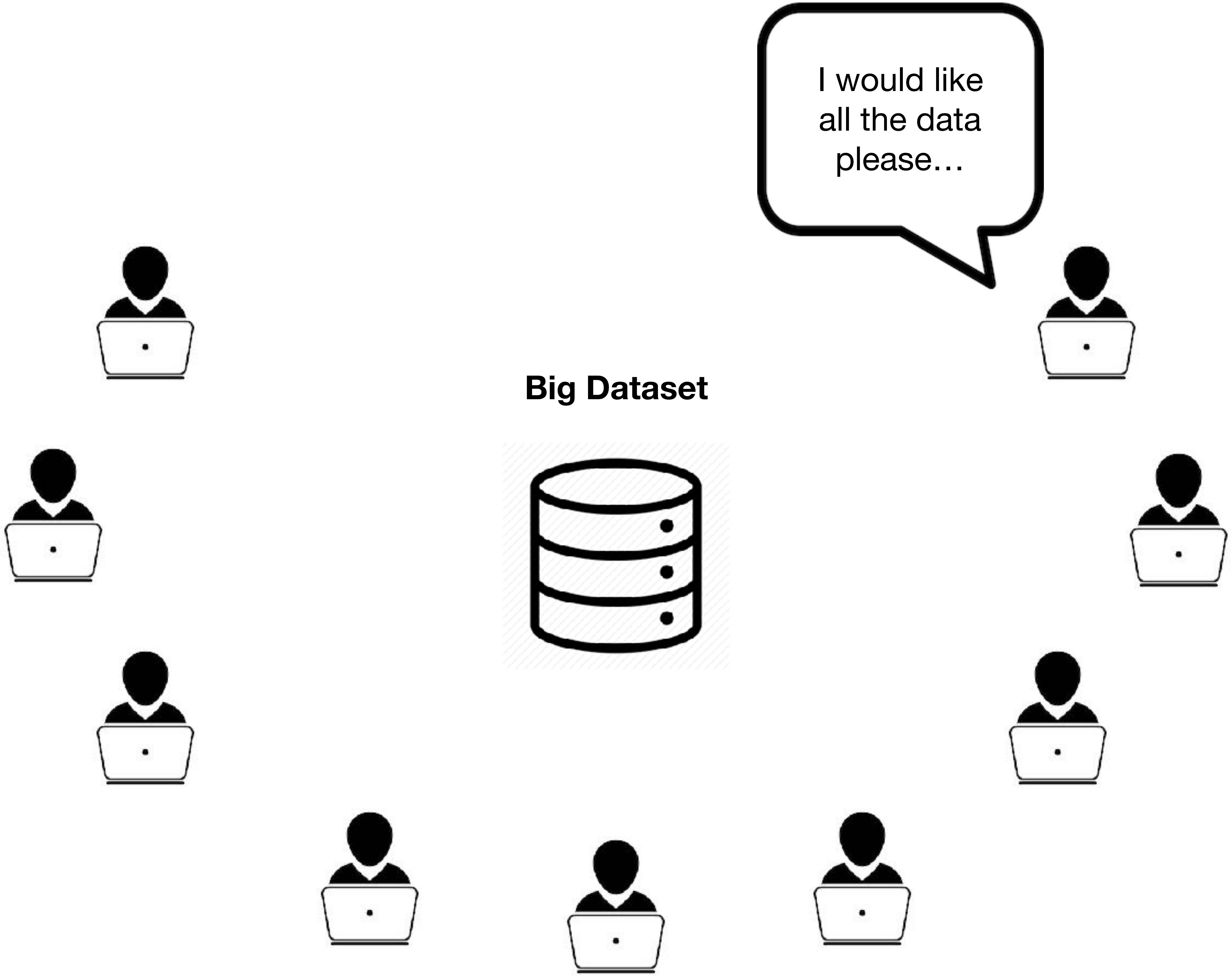


Altimetry Past, Present & Future

Credit: NASA SVS / PODAAC

Flourescence microscopy image of neurons by Jan Schmoranzer (link)

ESA Gaia DR2 ([link])


I would like
all the data
please…
Big Dataset

How should scientific data infrastructure be organized for the petabyte era?

WHAT IS PANGEO?

"A community platform for Big Data geoscience"

- Open Community
- Open Source Software
- Open Source Infrastructure

PANGEO COMMUNITY

HTTP://PANGEO.IO

Lamont-Doherty Earth Observatory
COLUMBIA UNIVERSITY | EARTH INSTITUTE

NCAR
NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

Met Office

NASA

USGS

GEOMAR

UNIVERSITY of WASHINGTON
eScience Institute

BIDS
BERKELEY INSTITUTE FOR DATA SCIENCE

jupyter

cnes
CENTRE NATIONAL D'ÉTUDES SPATIALES

QUEENS COLLEGE

element84

Geophysical Fluid Dynamics Laboratory
GFDL
NOAA

JUPITER

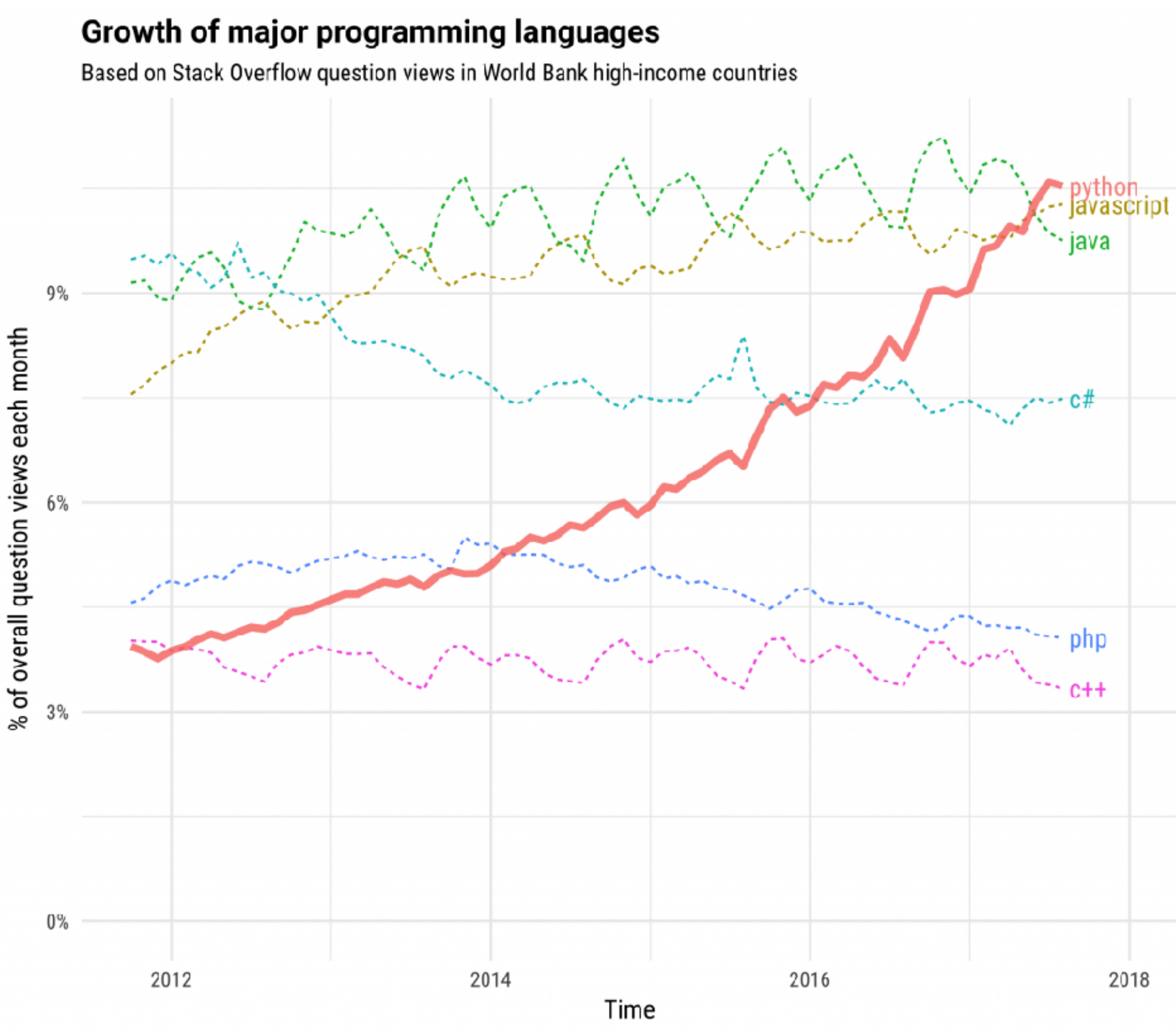
ANACONDA
Powered by Continuum Analytics

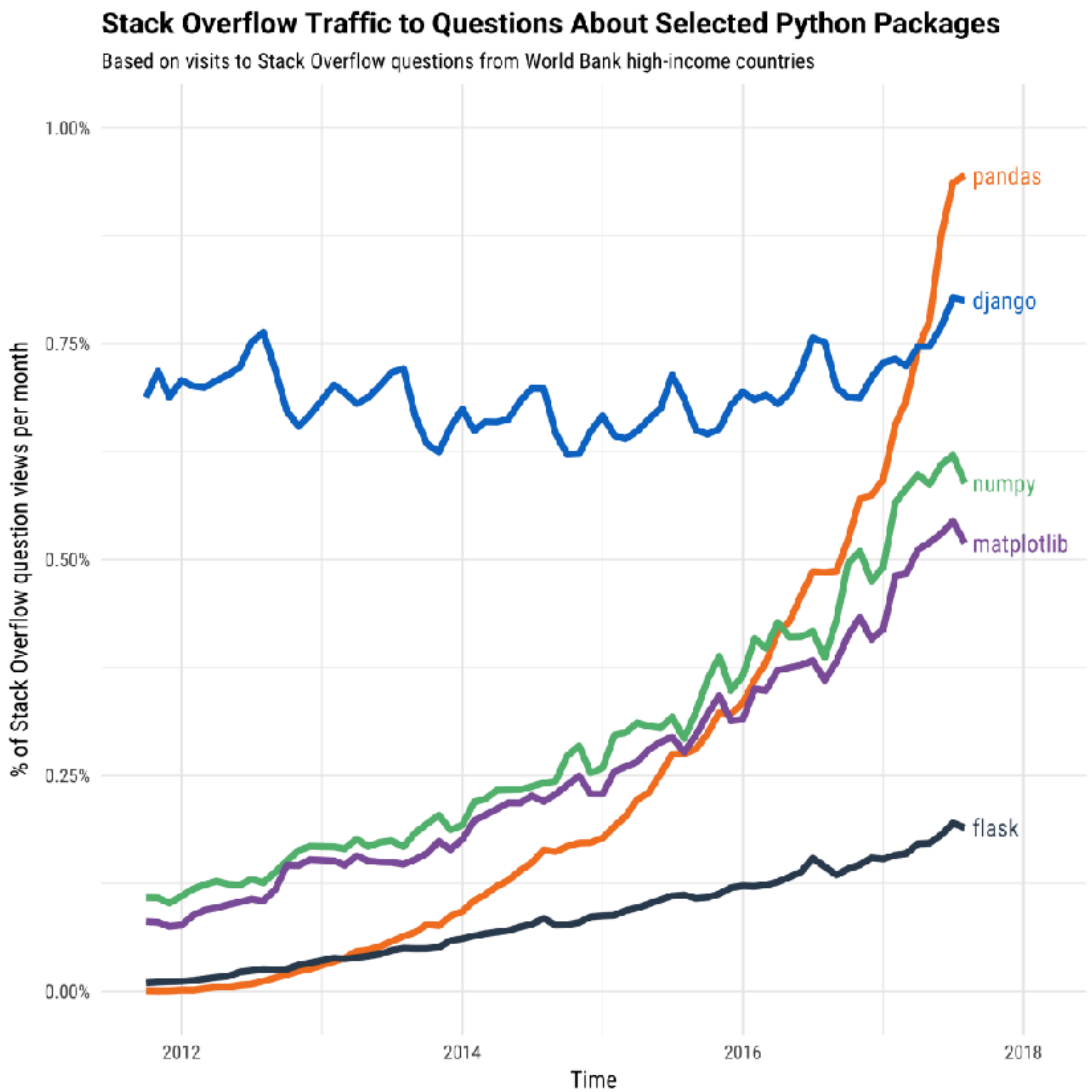
Rh RHODIUM GROUP

CLIMACELL
Weather Revealed

PANGEO TIMELINE

SCIENTIFIC PYTHON FOR DATA SCIENCE





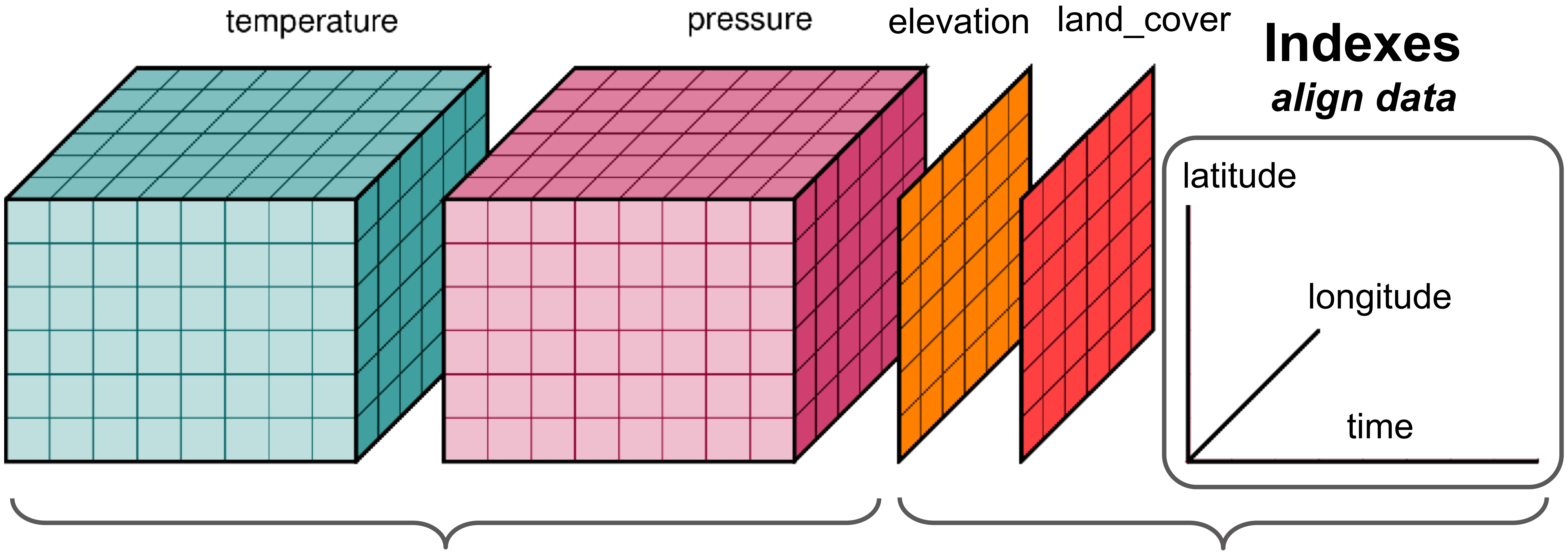
source: stackoverflow.com

PANGEO SOFTWARE

SCIENTIFIC PYTHON FOR GEOSCIENCE

Credit: Stephan Hoyer, Jake Vanderplas (SciPy 2015)

XARRAY DATASET: MULTIDIMENSIONAL VARIABLES WITH COORDINATES AND METADATA



"netCDF meets pandas.DataFrame"

Credit: Stephan Hoyer

XARRAY MAKES SCIENCE EASY

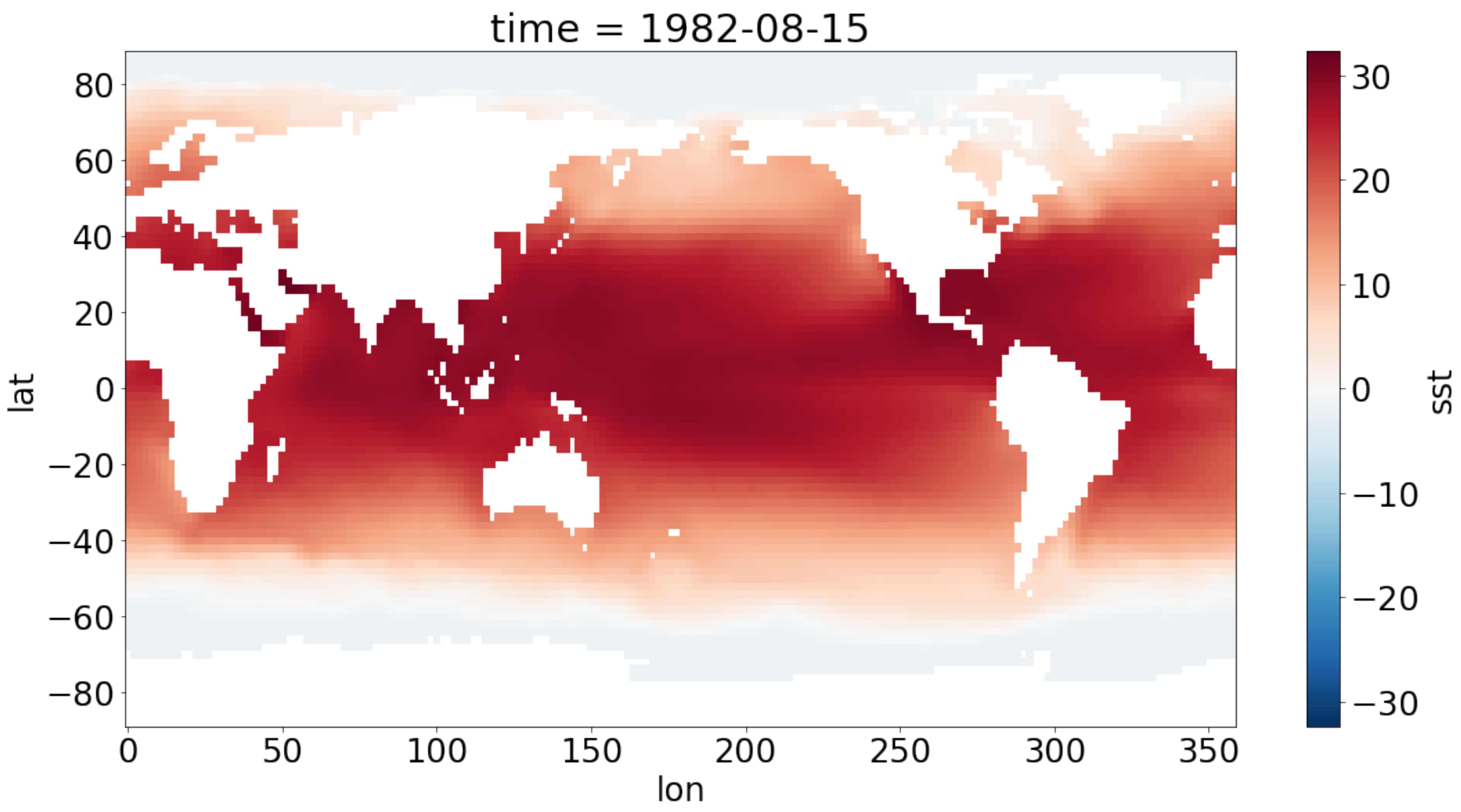
```python
import xarray as xr
ds = xr.open_dataset('NOAA_NCDC_ERSST_v3b_SST.nc')
ds
```

```
<xarray.Dataset>
Dimensions:  (lat: 89, lon: 180, time: 684)
Coordinates:
  * lat      (lat) float32 -88.0 -86.0 -84.0 -82.0 -80.0 -78.0 -76.0 -74.0 ...
  * lon      (lon) float32 0.0 2.0 4.0 6.0 8.0 10.0 12.0 14.0 16.0 18.0 20.0 ...
  * time     (time) datetime64[ns] 1960-01-15 1960-02-15 1960-03-15 ...
Data variables:
    sst      (time, lat, lon) float64 nan nan nan nan nan nan nan nan nan ...
Attributes:
    Conventions:  IRIDL
    source:       https://iridl.ldeo.columbia.edu/SOURCES/.NOAA/.NCDC/.ERSST/...
```

PANGEO

XARRAY: LABEL-BASED SELECTION

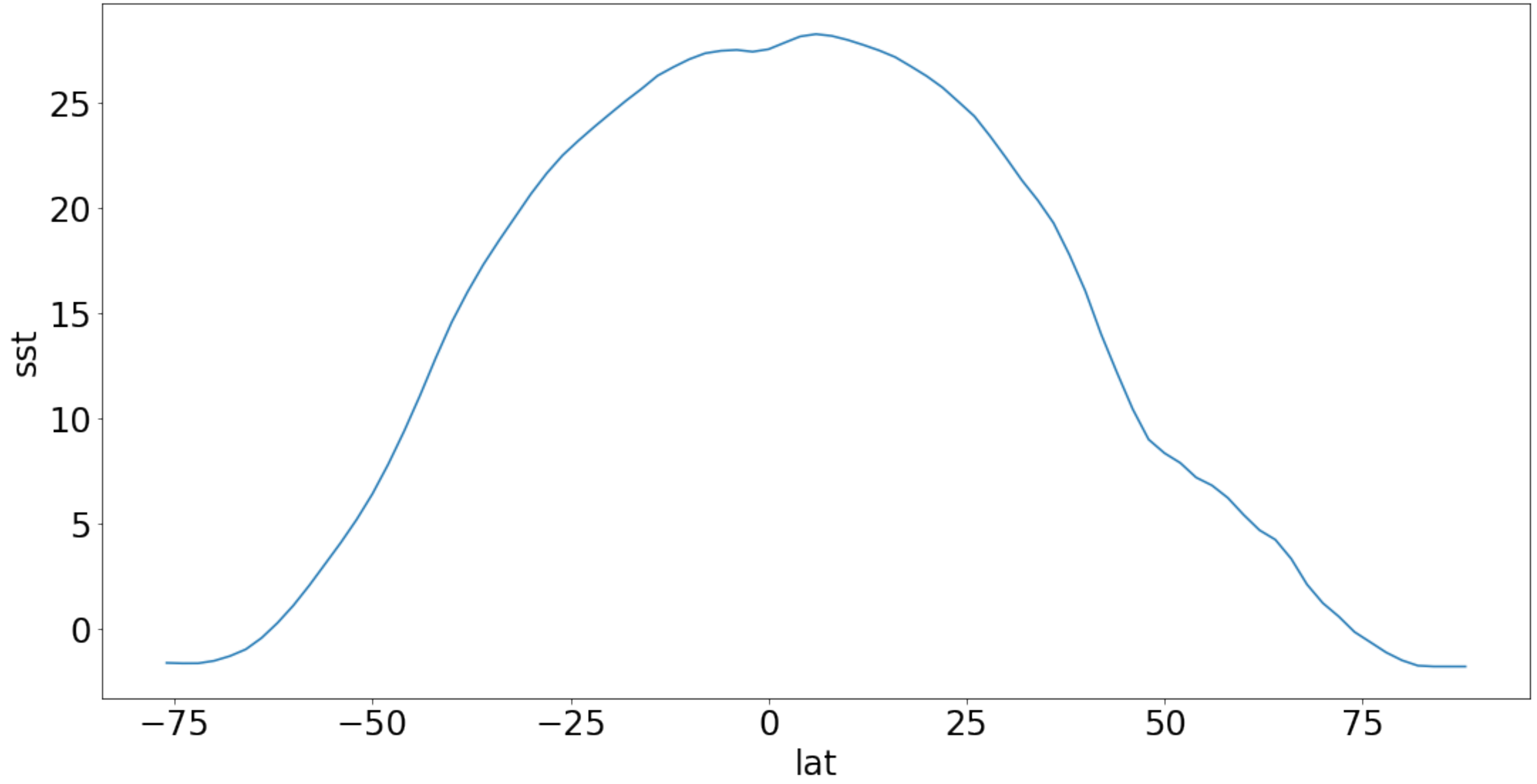
```
# select and plot data from my birthday
ds.sst.sel(time='1982-08-07', method='nearest').plot()
```



PANGEO

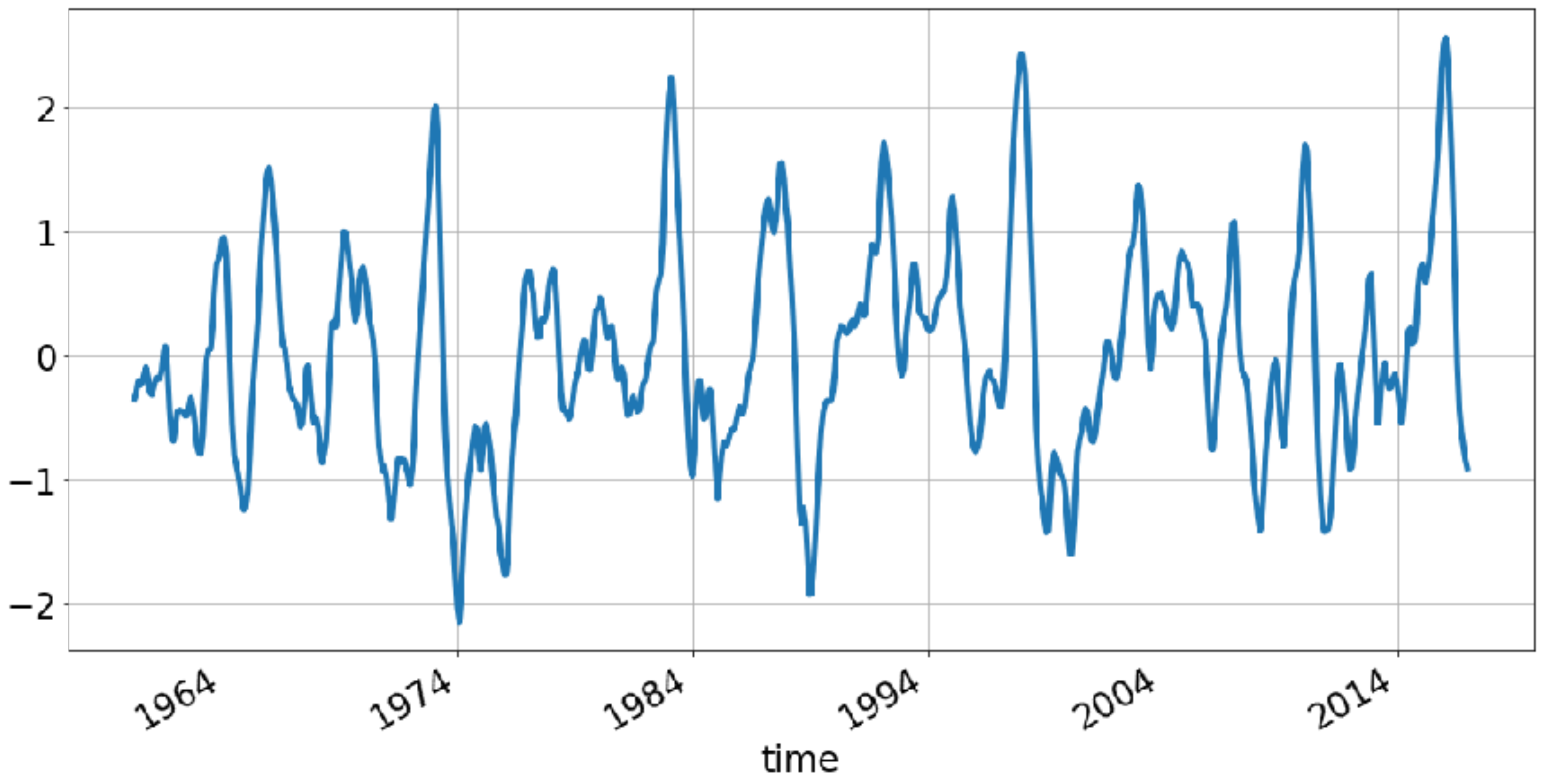
XARRAY: LABEL-BASED OPERATIONS

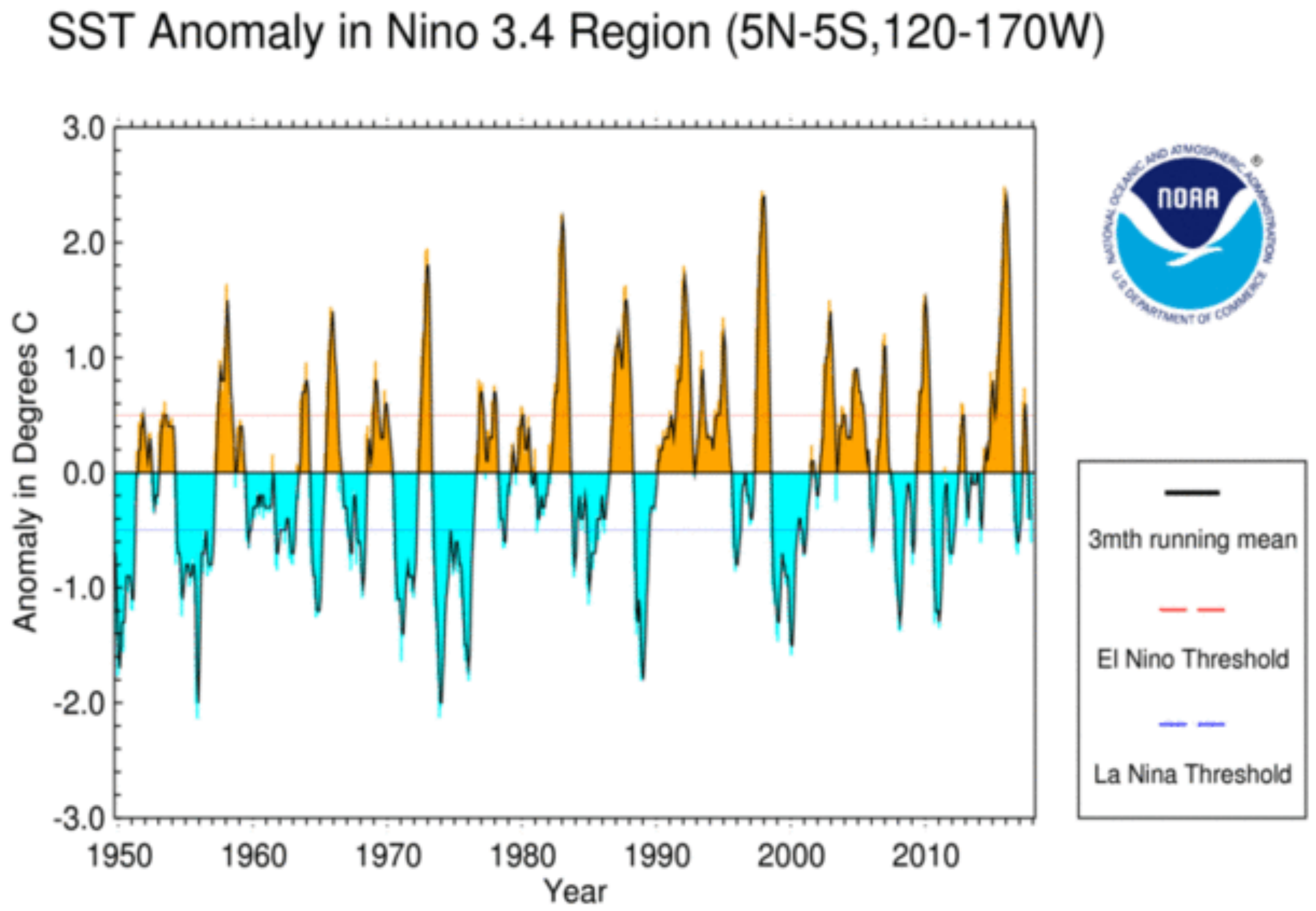
```
# zonal and time mean temperature
ds.sst.mean(dim=('time', 'lon')).plot()
```



XARRAY: GROUPING AND AGGREGATION

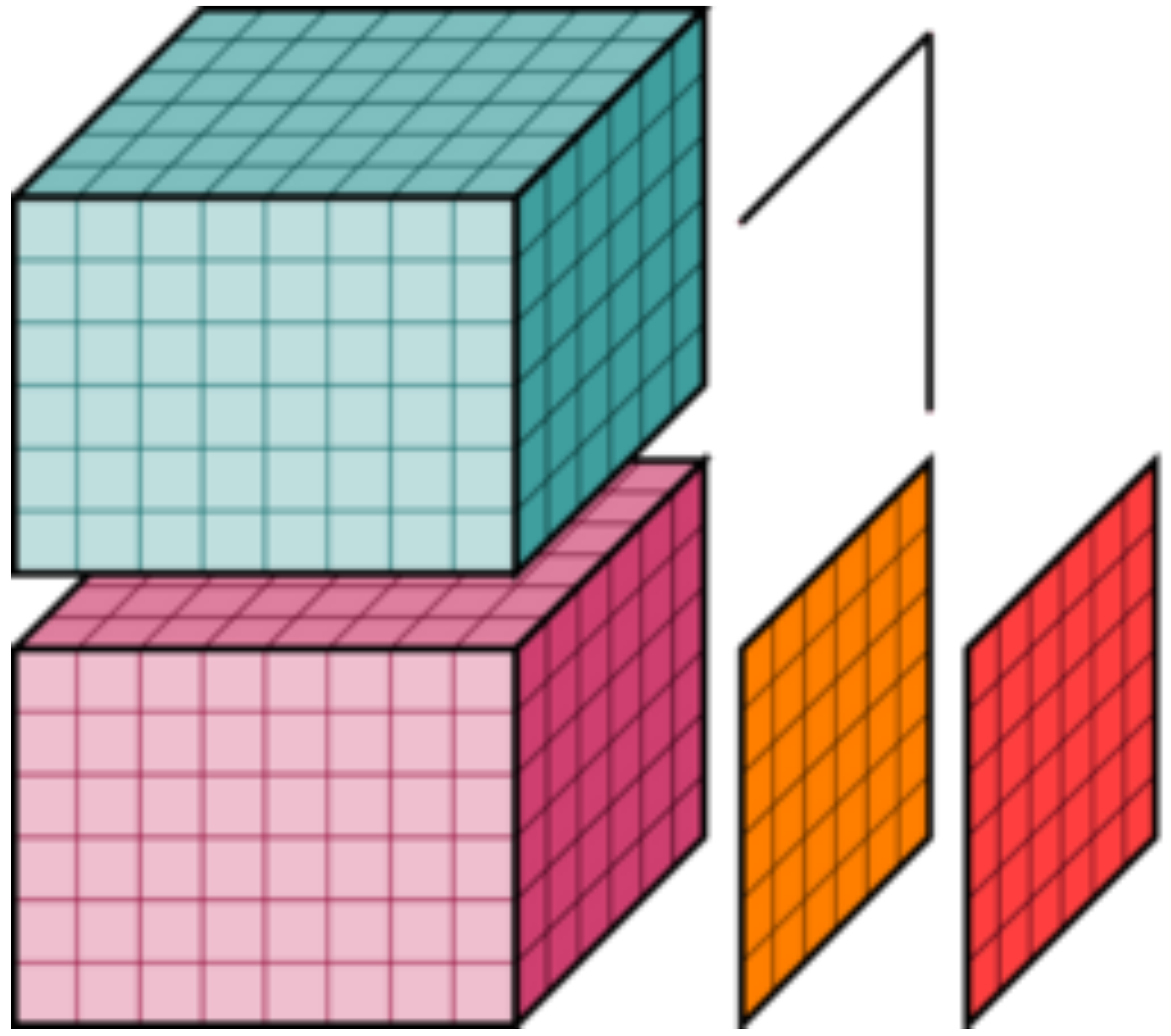
```
sst_clim = sst.groupby('time.month').mean(dim='time')
sst_anom = sst.groupby('time.month') - sst_clim
nino34_index = (sst_anom.sel(lat=slice(-5, 5), lon=slice(190, 240))
                        .mean(dim=('lon', 'lat'))
                        .rolling(time=3).mean(dim='time'))
nino34_index.plot()
```





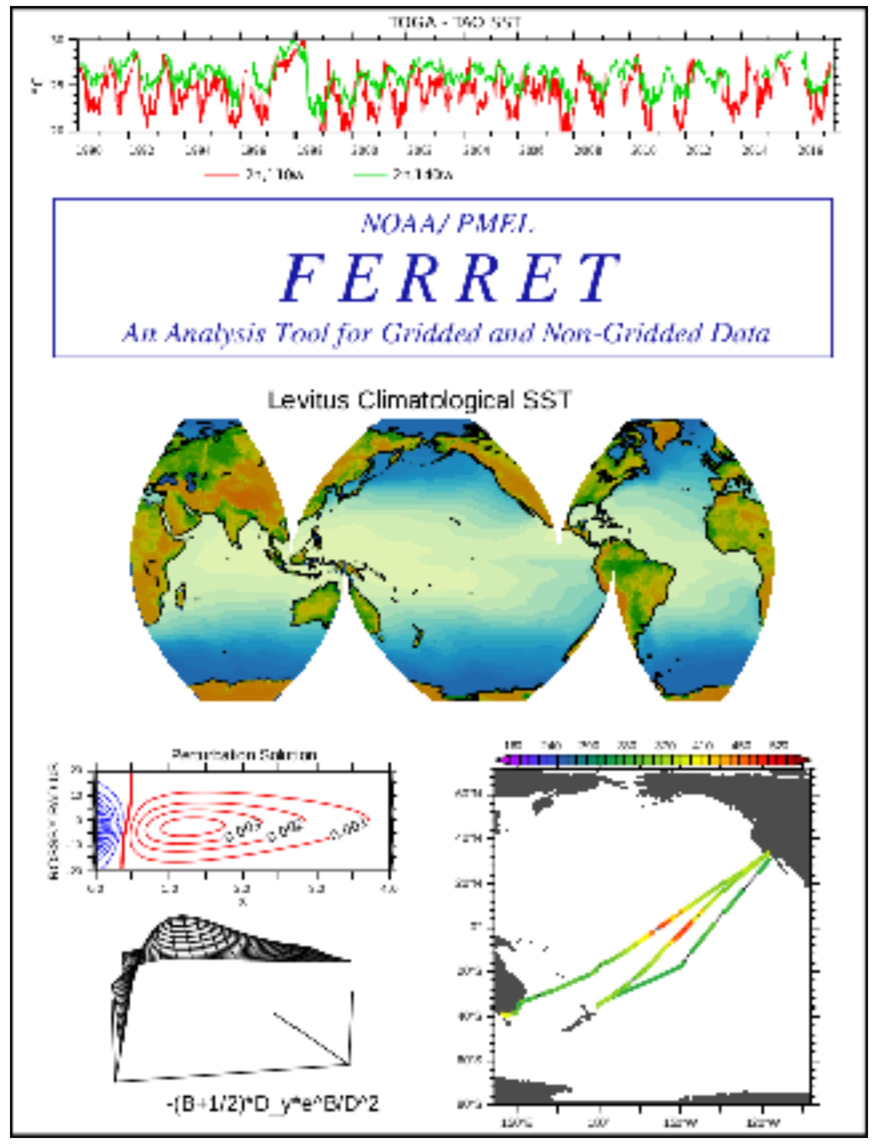
XARRAY

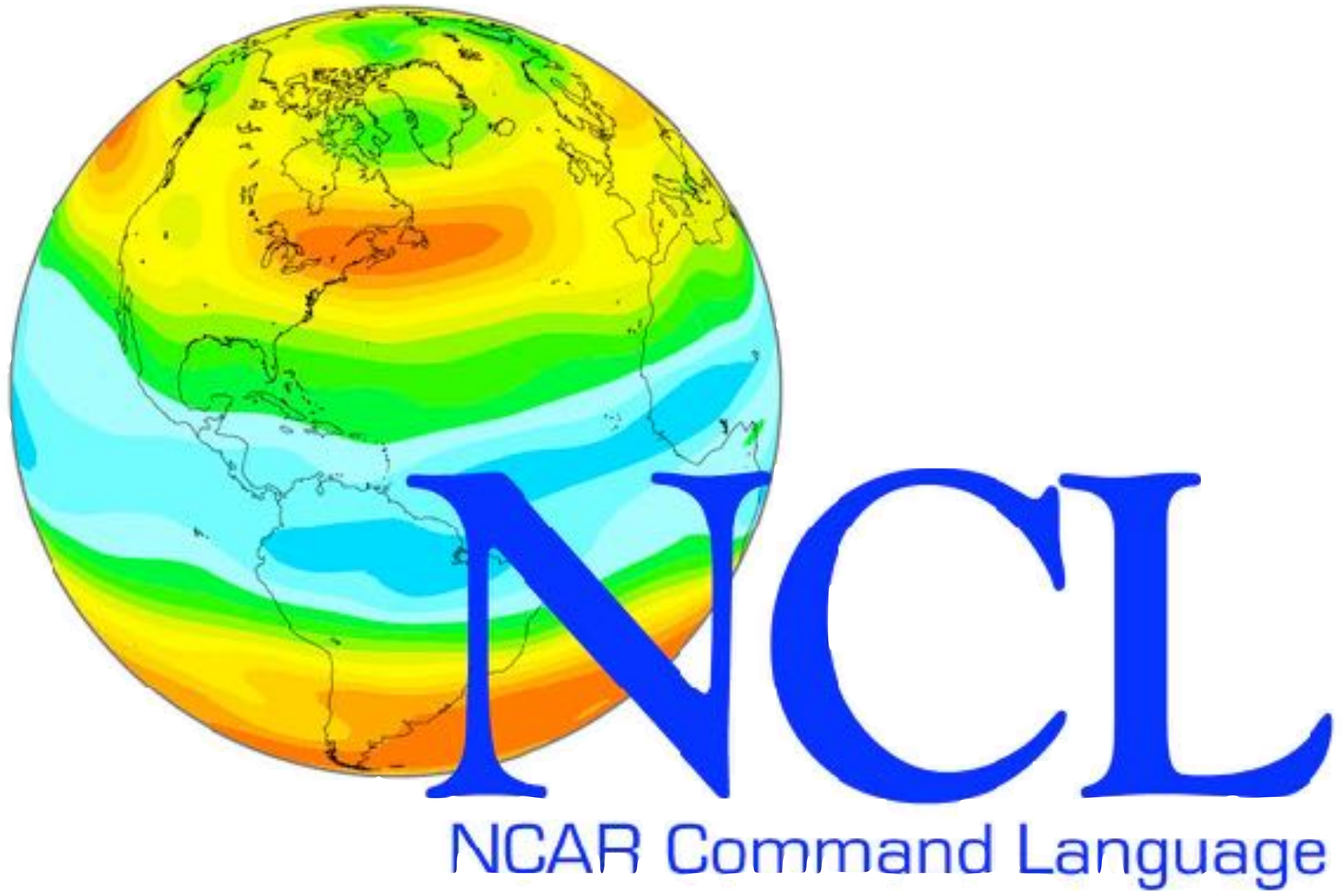
https://github.com/pydata/xarray

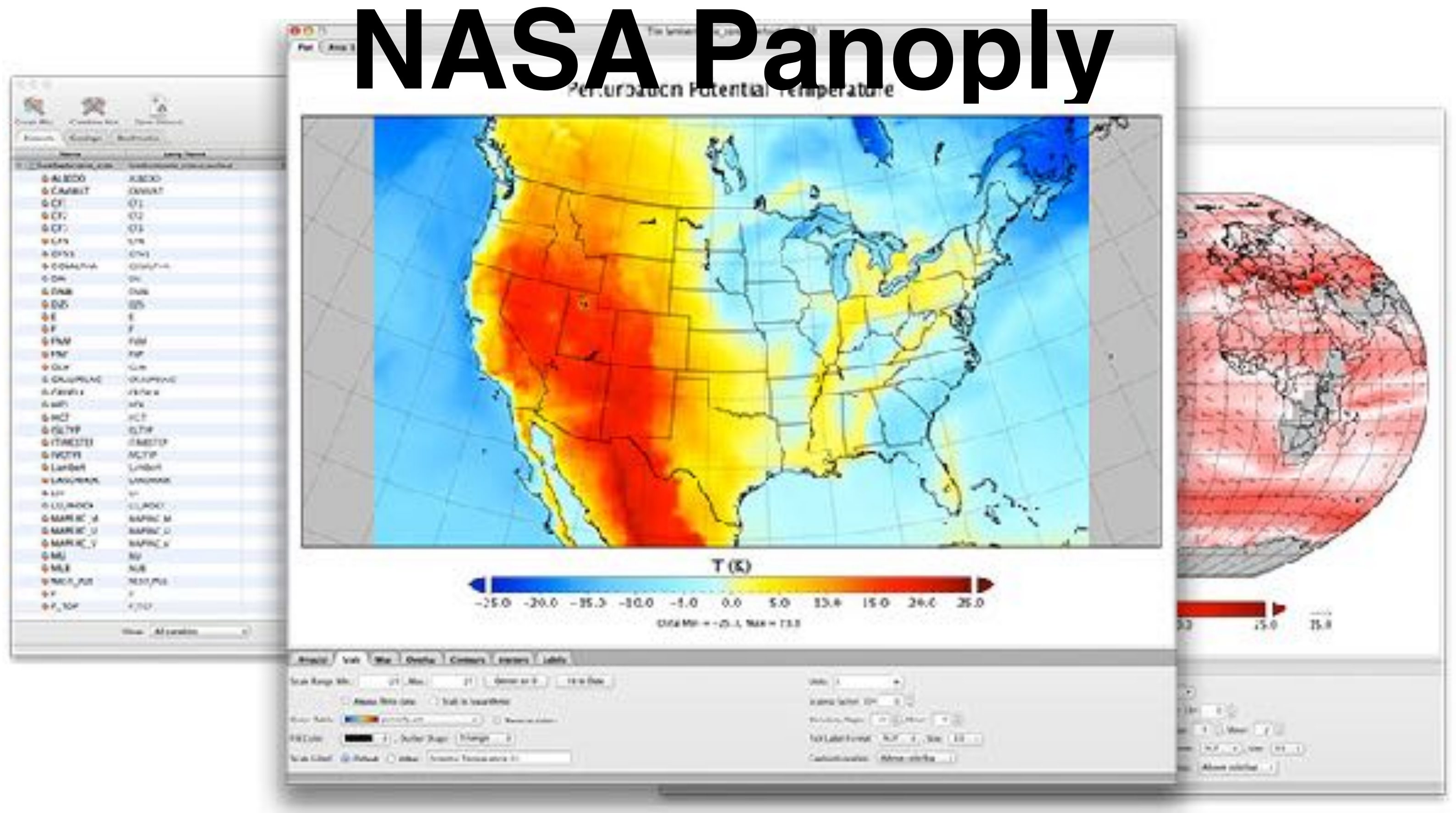


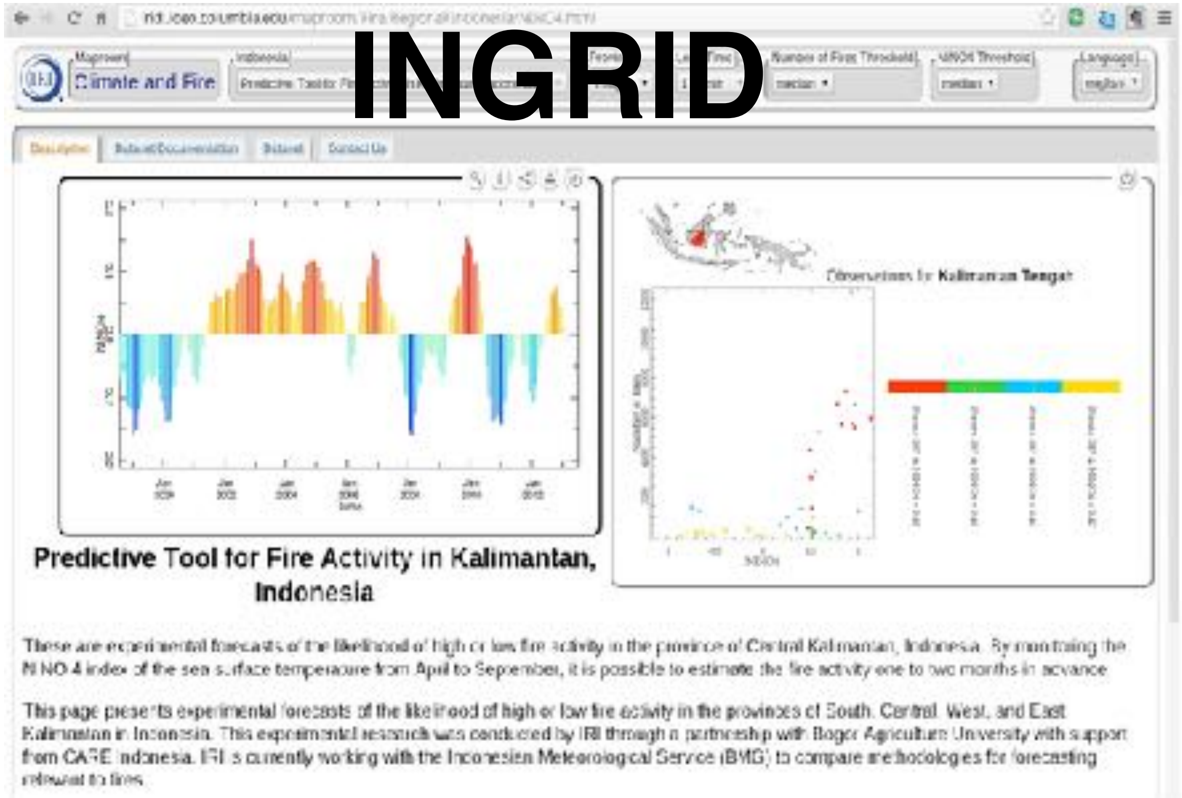
- label-based indexing and arithmetic
- interoperability with the core scientific Python packages (e.g., pandas, NumPy, Matplotlib)
- out-of-core computation on datasets that don't fit into memory (thanks dask!)
- wide range of input/output (I/O) options: netCDF, HDF, geoTIFF, zarr
- advanced multi-dimensional data manipulation tools such as group-by and resampling

LEGACY SOFTWARE



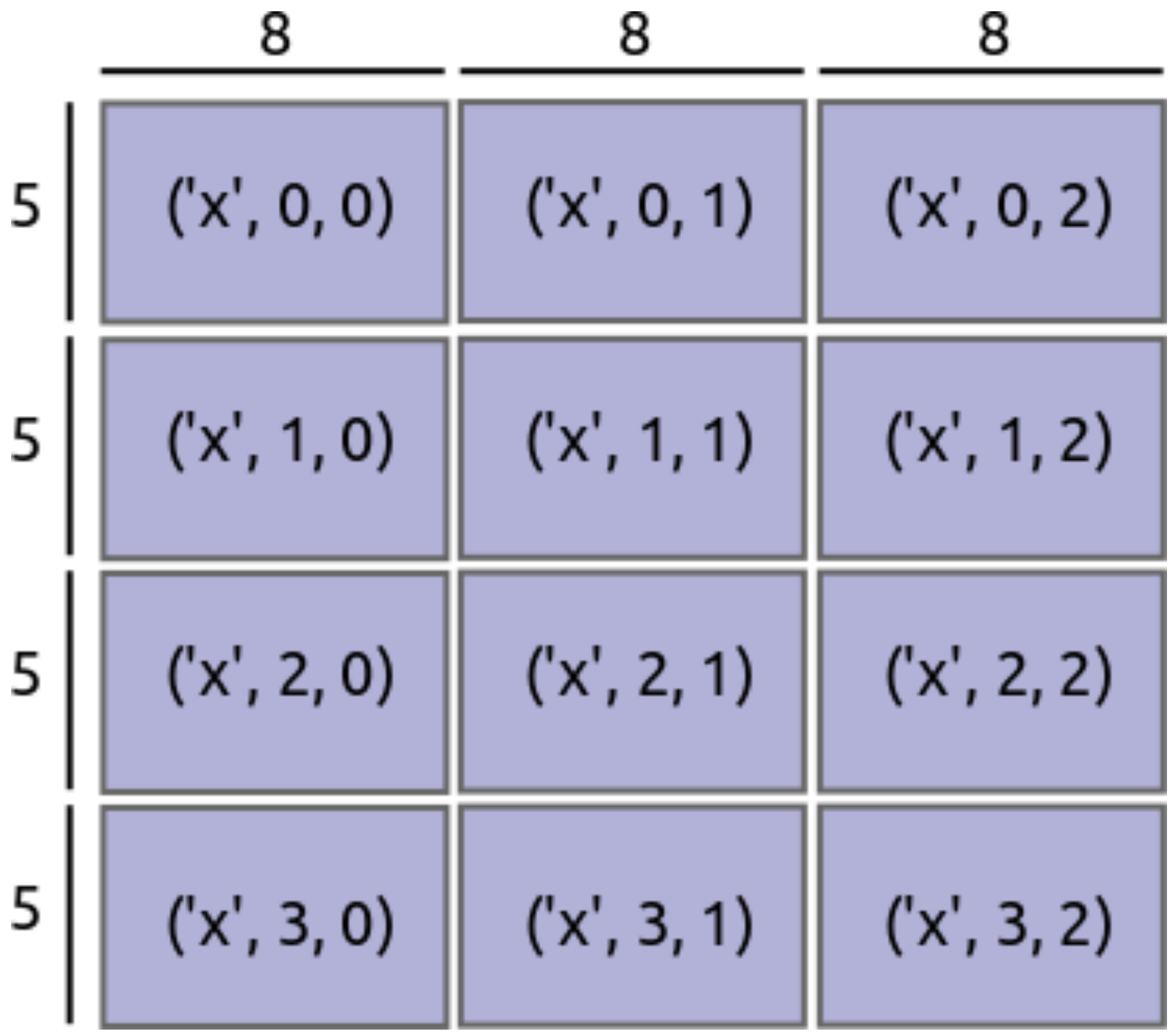




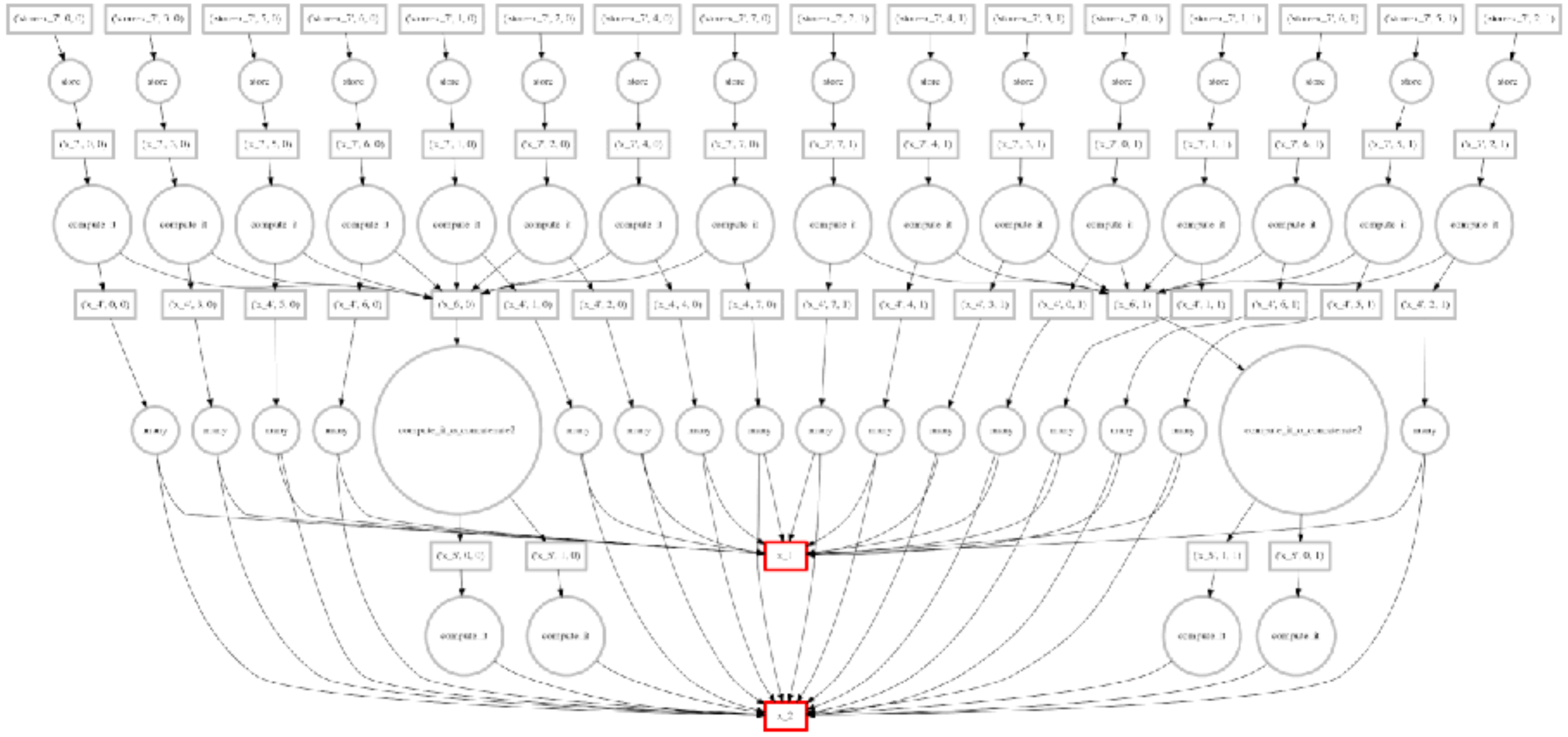


DASK

https://github.com/dask/dask/



ND-Arrays are split into chunks that comfortably fit in memory

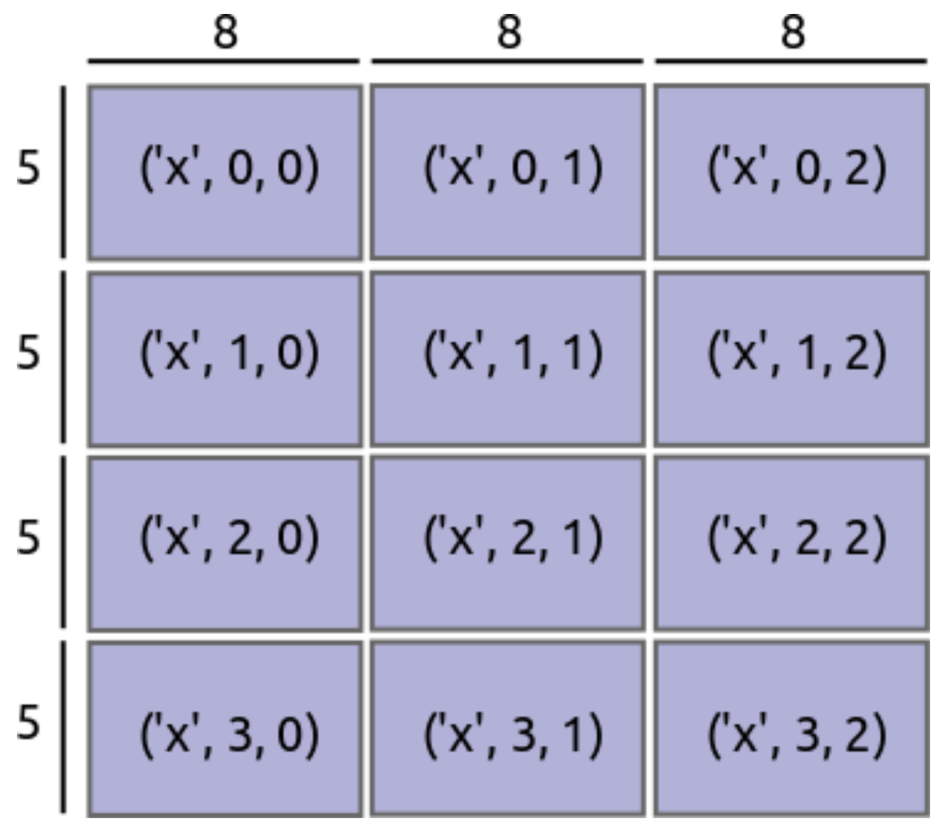


Complex computations represented as a graph of individual tasks.

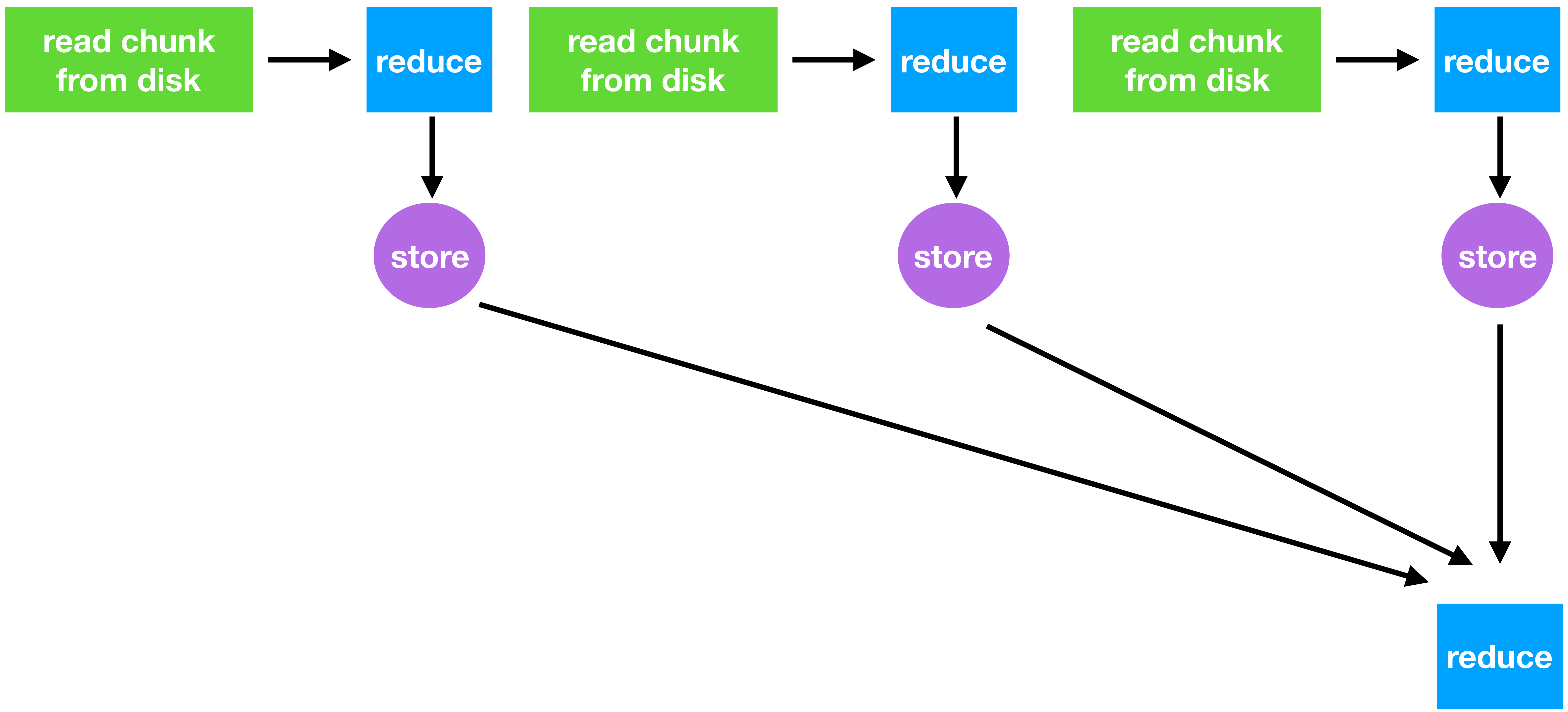
Scheduler optimizes execution of graph.

EXAMPLE CALCULATION: TAKE THE MEAN!

multidimensional array

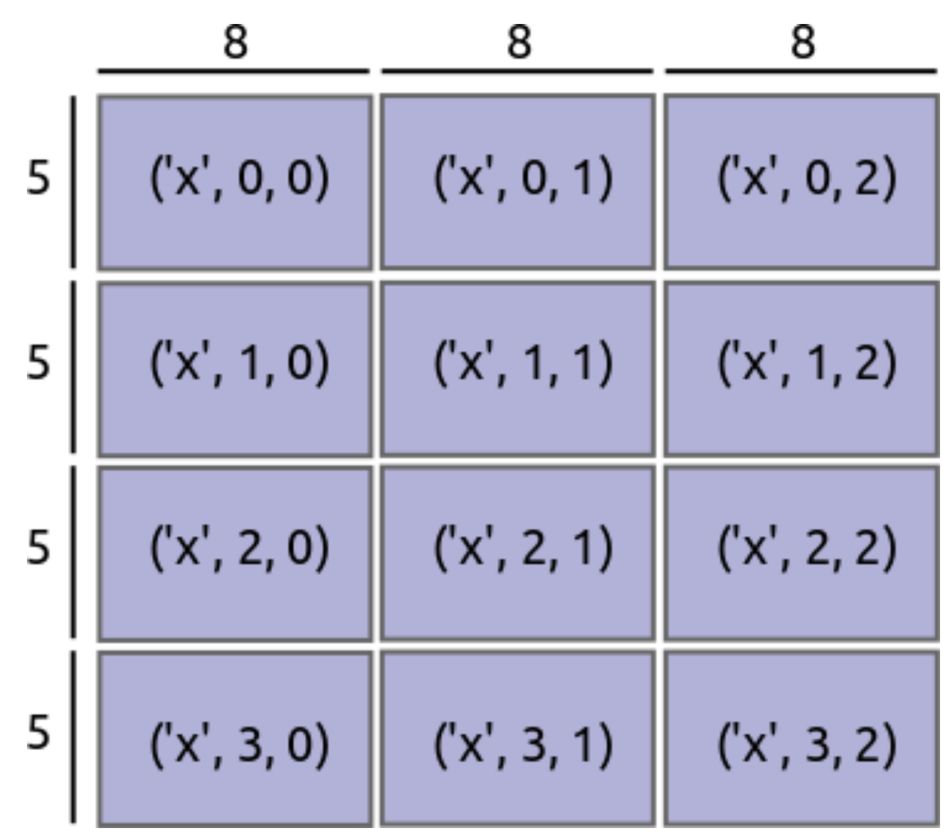


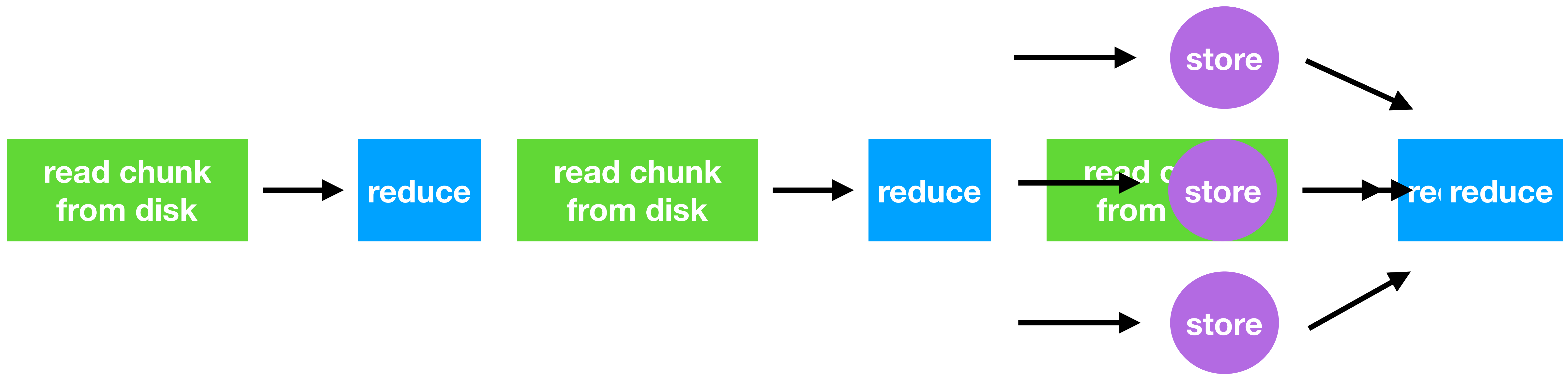
serial execution (a loop)



EXAMPLE CALCULATION: TAKE THE MEAN!

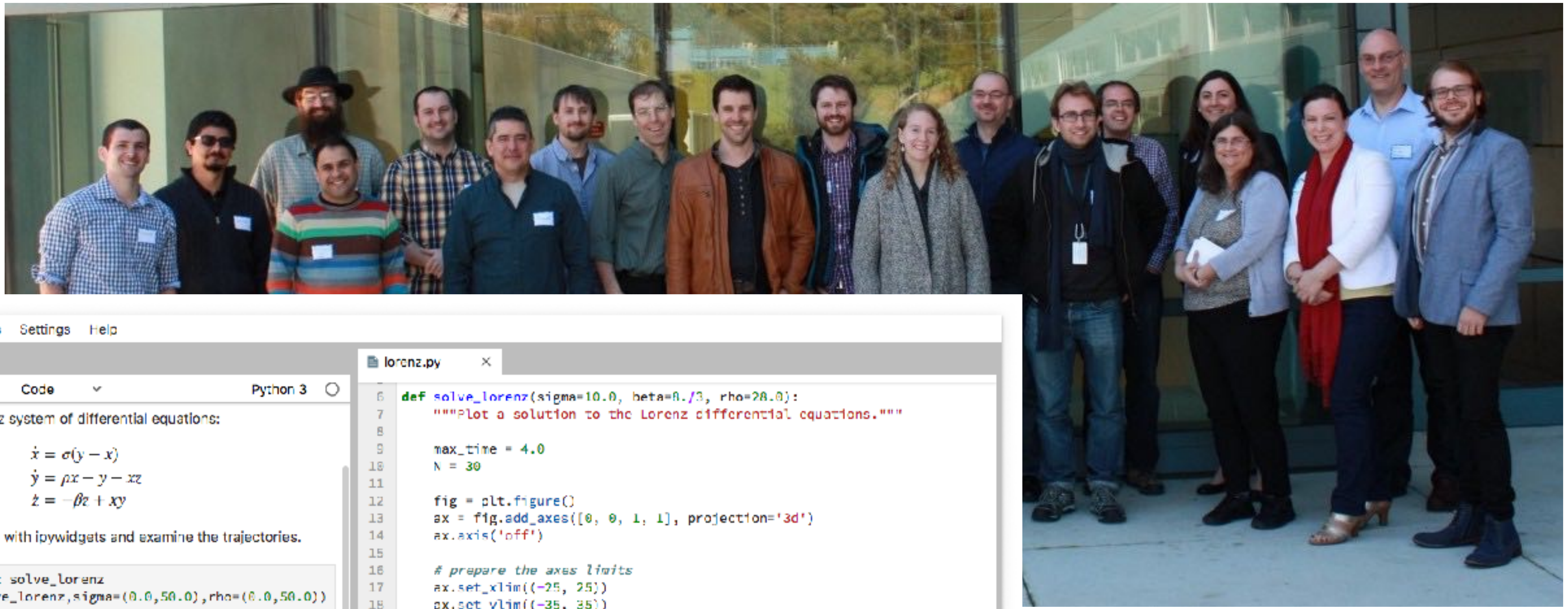
multidimensional array

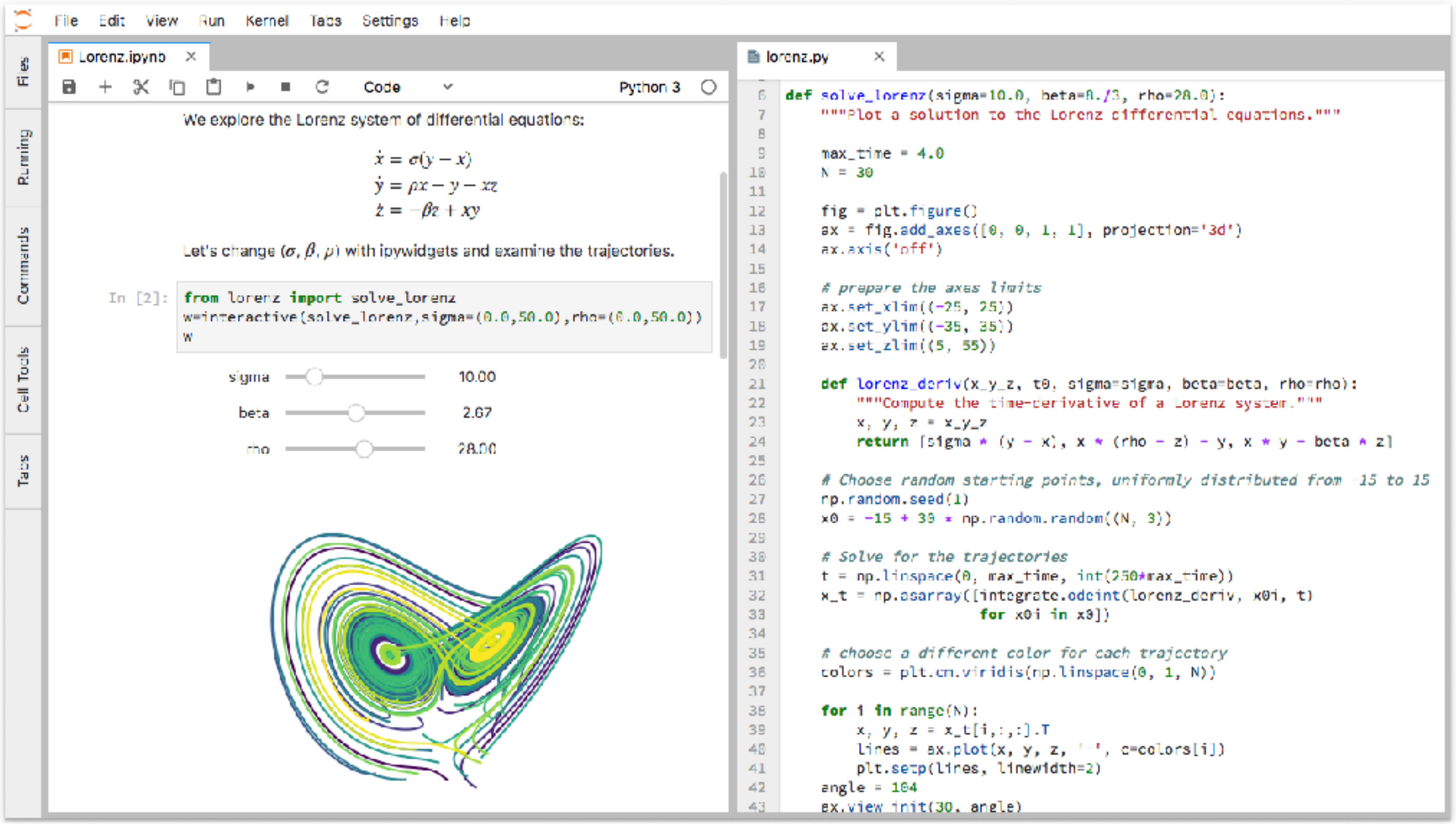




parallel execution (dask graph)

JUPYTER

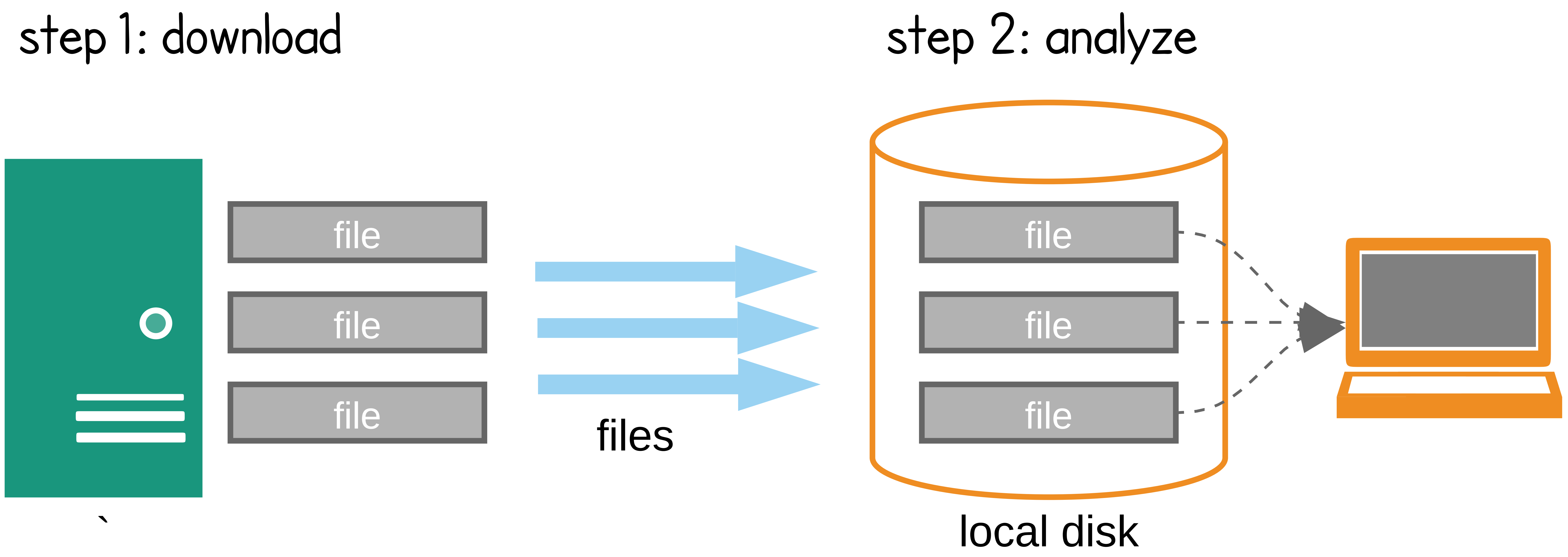




“Project Jupyter exists to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages.”

PANGEO INFRASTRUCTURE

FILE-BASED APPROACH



Data provider's responsibilities

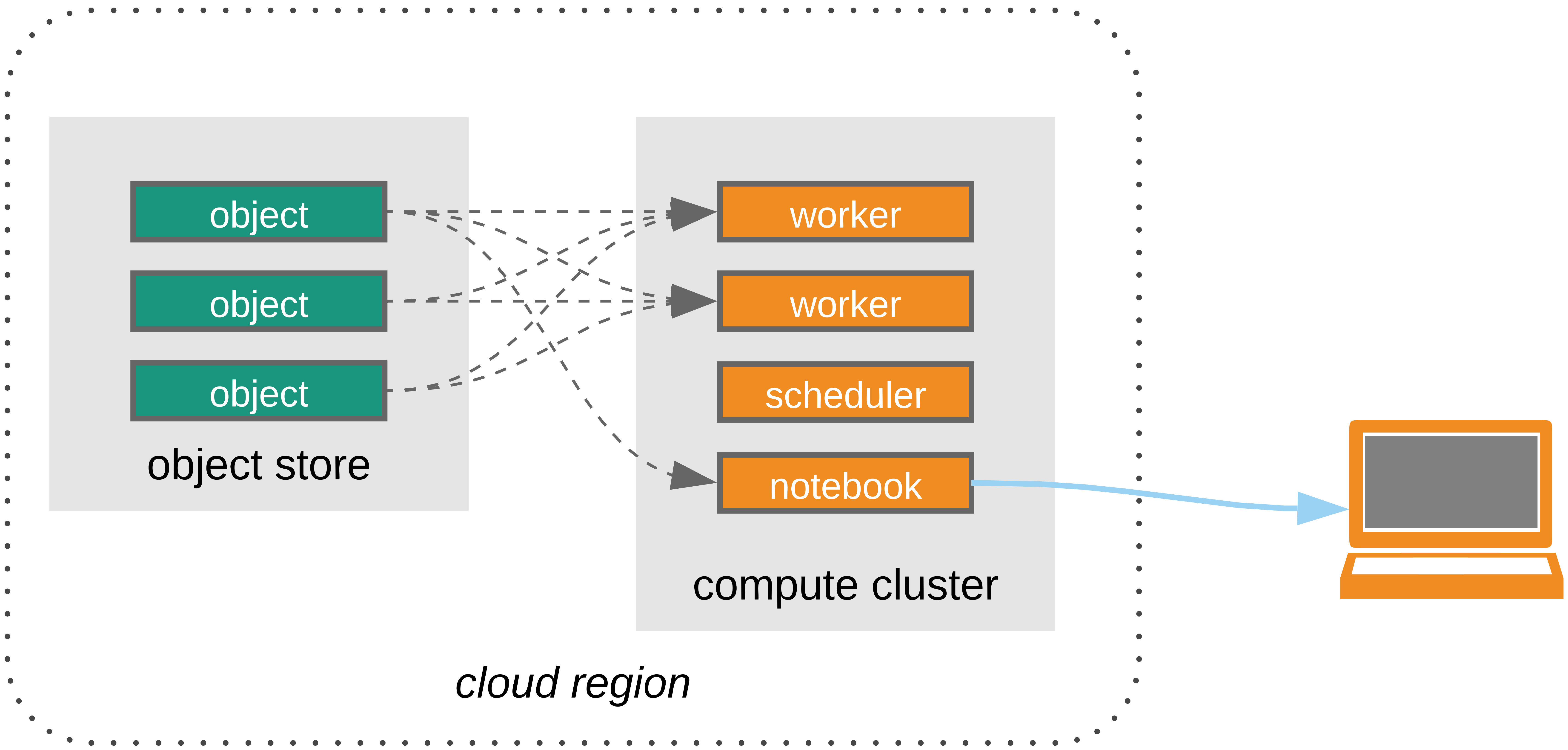
End user's responsibilities

SERVER-SIDE DATABASE

Data provider's responsibilities

End user's responsibilities

CLOUD-NATIVE APPROACH



Data provider's responsibilities

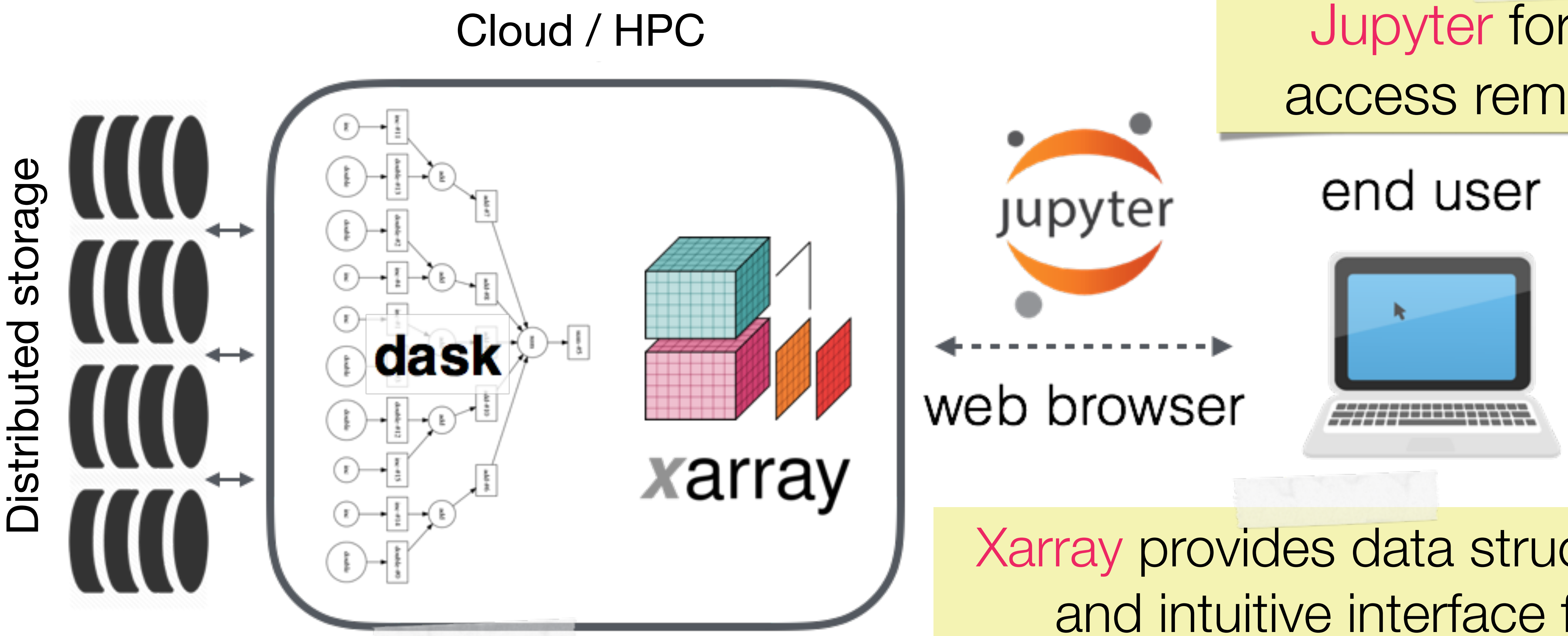
End user's responsibilities

PANGEO PRINCIPLES FOR CLOUD-NATIVE SCIENCE INFRASTRUCTURE

- **Community-driven** - Our needs are no different from those of our peer institutions. By developing infrastructure collaboratively, we can accomplish much more than any one institution can alone.
- **Open source** - Because infrastructure is code, the code should be licensed in a way that enables the entire research community to reuse and build upon it.
- **Modular** - "all in one" solutions are impossible to maintain long term. Separation of concerns is a key principle of good software and systems engineering.
- **Vendor neutral** - Academic research infrastructure should use only vendor-neutral services APIs. If this principle is followed, it means we can redeploy our infrastructure anywhere.

PANGEO ARCHITECTURE

“Analysis Ready Data” stored on globally-available distributed storage.



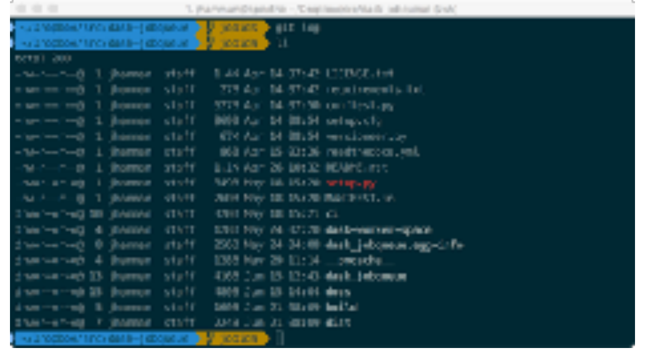
Jupyter for interactive access remote systems

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

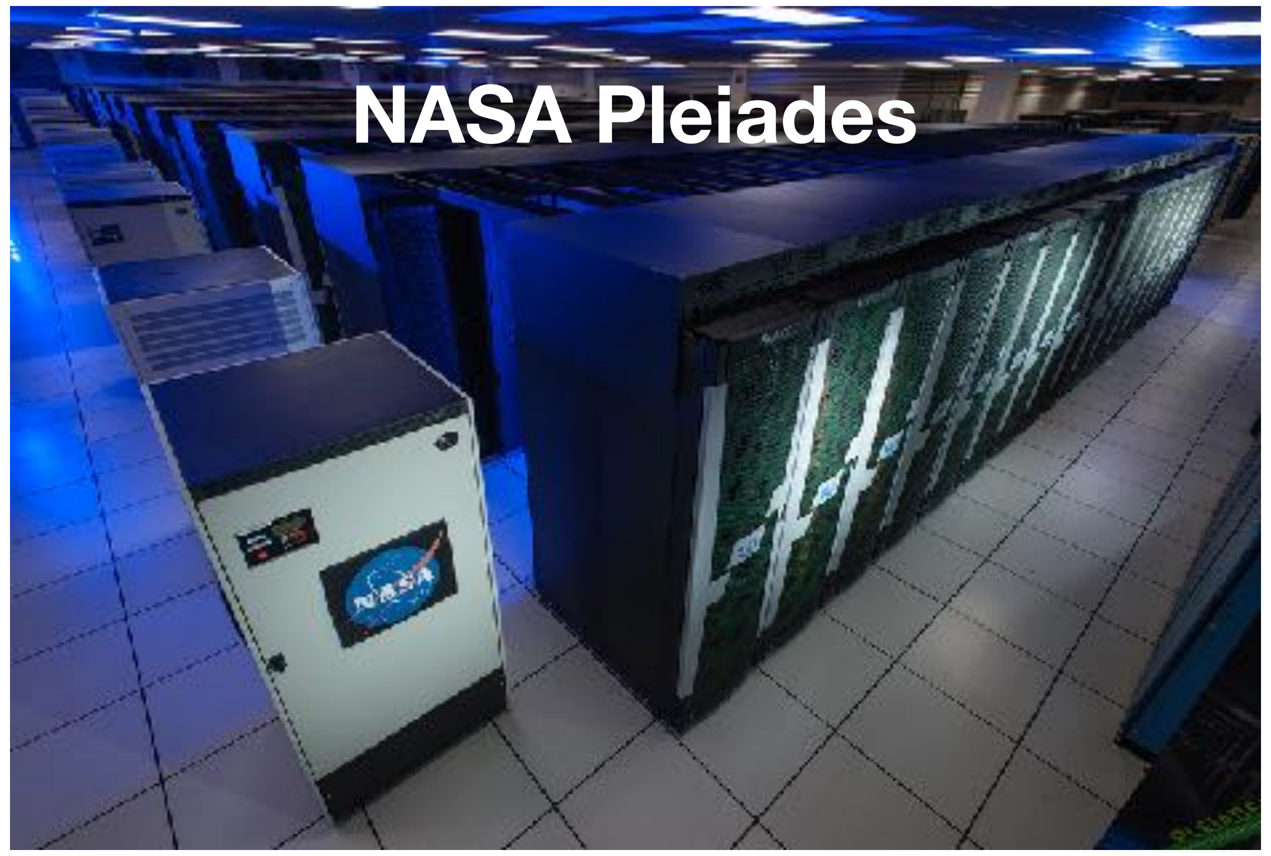
Dask tells the nodes what to do.

Xarray provides data structures and intuitive interface for interacting with datasets

BUILD YOUR OWN PANGEO

Storage Formats	HDF	OPeNDAP	Cloud Optimized COG/Zarr/Parquet/etc.
ND-Arrays	NumPy	DASK	More coming…
Data Models	xarray	Iris	pandas $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$
Processing Mode	jupyter Interactive	Batch 	Serverless
Compute Platform	HPC 	Cloud Google Cloud Platform	Local 

PANGEO DEPLOYMENTS



NCAR Cheyenne

kubernetes

Google Cloud Platform

Jetstream

Microsoft Azure

aws

HTTP://PANGEO.IO/DEPLOYMENTS.HTML

CONTINUOUS DEPLOYMENT

GitHub, Inc. [US] | https://github.com/pangeo-data/pangeo-cloud-federation

README.md

This repository manages the continuous deployment of the Pangeo Cloud Federation JupyterHub Kubernetes clusters using hubploy. It contains scripts to automatically redeploy when the image definition or chart parameters are changed.

Changing the image will typically take ~20 minutes, and changing a Helm config variable ~1 minute.

Clusters

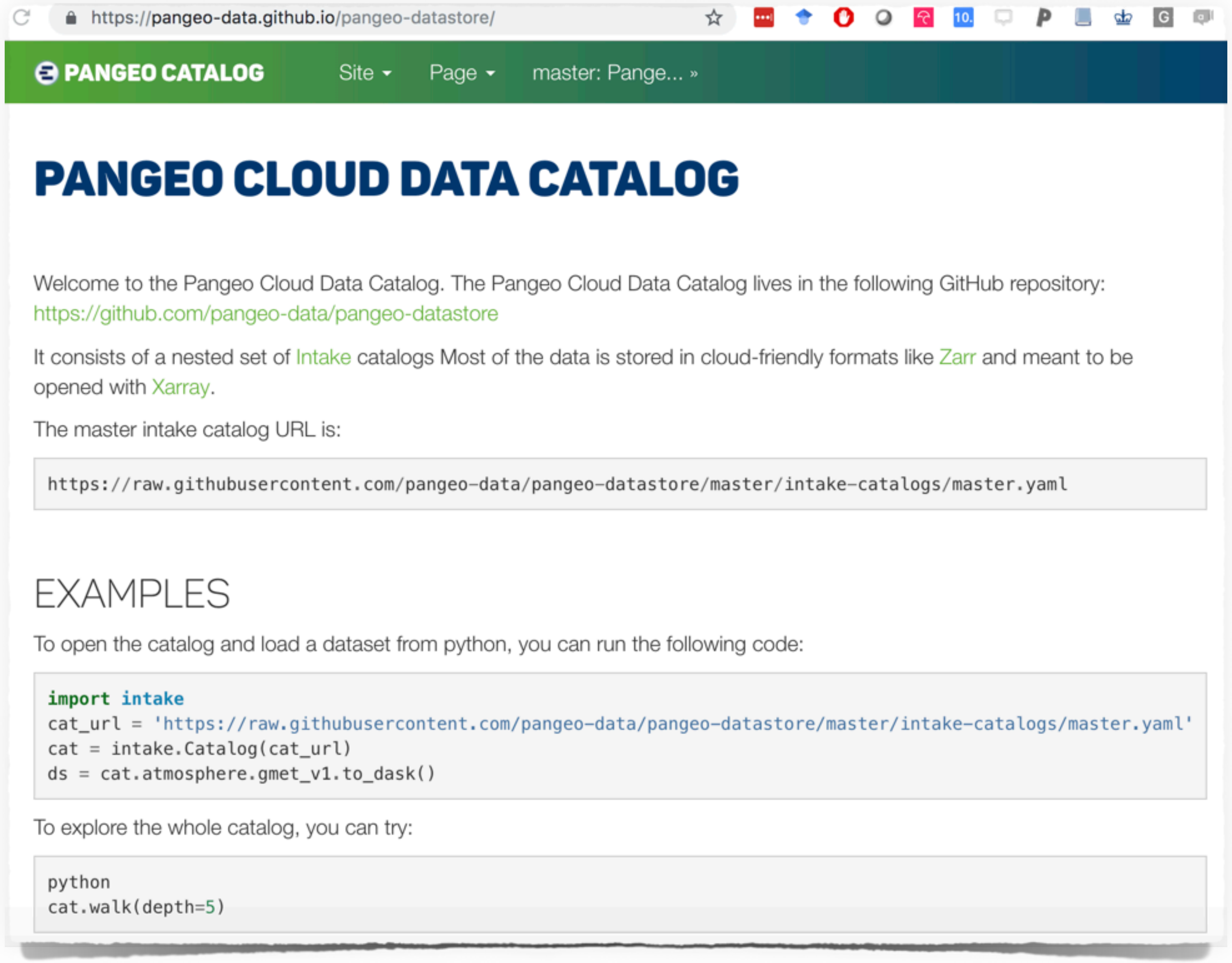
Name	Cloud: region	Staging URL	Production URL
dev	GCP: us-central1-b	https://staging.hub.pangeo.io	https://hub.pangeo.io
ocean	GCP: us-central1-b	https://staging.ocean.pangeo.io	https://ocean.pangeo.io
hydro	GCP: us-central1-b	https://staging.hydro.pangeo.io	https://hydro.pangeo.io
nasa	AWS: us-east-1	https://staging.nasa.pangeo.io	https://nasa.pangeo.io
icesat	AWS: us-west-2	https://staging.icesat.pangeo.io	https://icesat.pangeo.io

Build Status

Branch	Build
staging	FAILED
prod	PASSED

- https://github.com/pangeo-data/pangeo-cloud-federation
- Cloud-based clusters managed with helm /kubernetes
- Deployment is completely automated via GitHub / circleci
- Resources scale elastically with demand

CLOUD DATA CATALOG



https://pangeo-data.github.io/pangeo-datastore/

PANGEO CATALOG Site Page master: Pange... »

PANGEO CLOUD DATA CATALOG

Welcome to the Pangeo Cloud Data Catalog. The Pangeo Cloud Data Catalog lives in the following GitHub repository: https://github.com/pangeo-data/pangeo-datastore

It consists of a nested set of Intake catalogs Most of the data is stored in cloud-friendly formats like Zarr and meant to be opened with Xarray.

The master intake catalog URL is:

```
https://raw.githubusercontent.com/pangeo-data/pangeo-datastore/master/intake-catalogs/master.yaml
```

EXAMPLES

To open the catalog and load a dataset from python, you can run the following code:

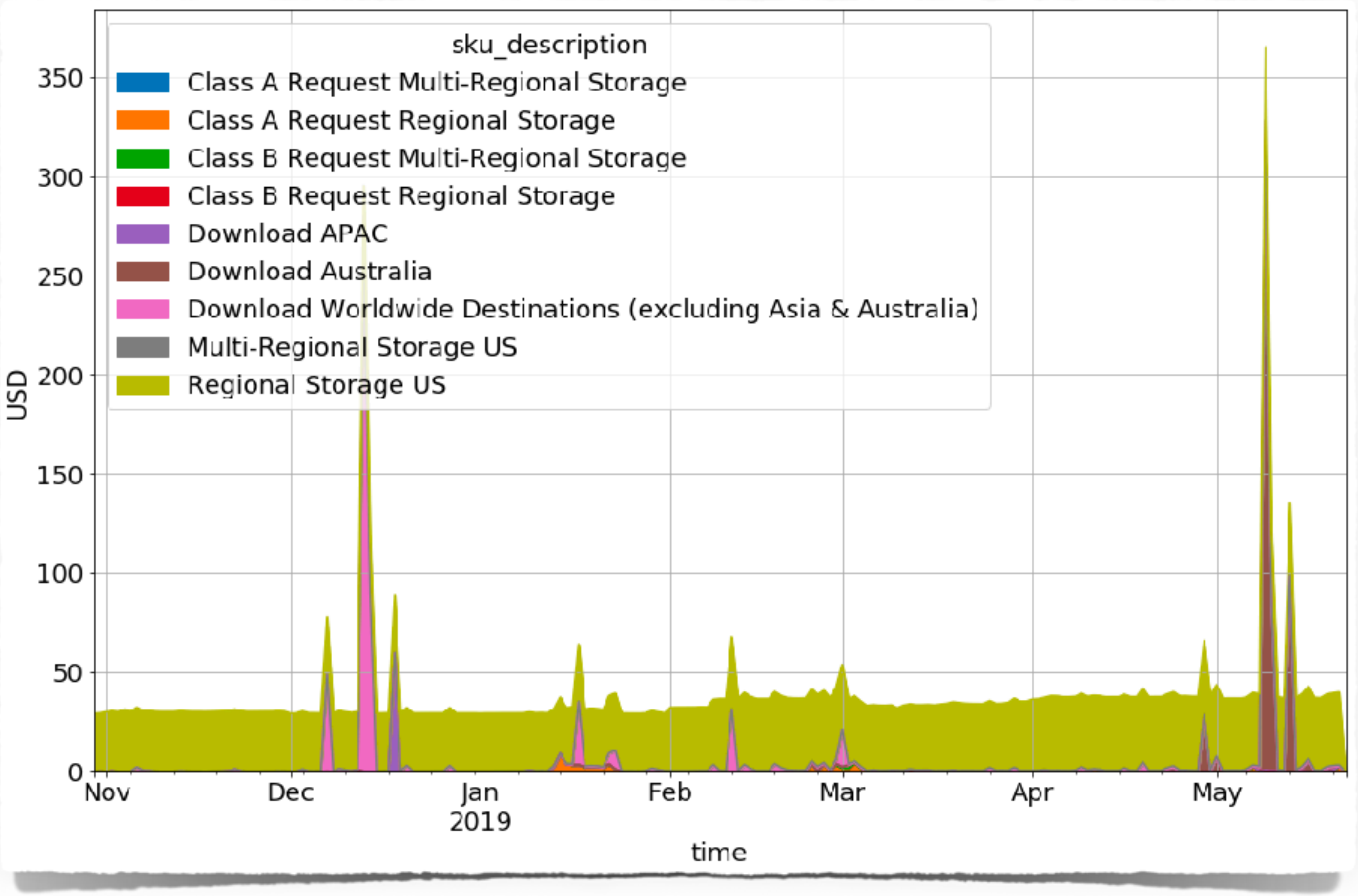
```
import intake
cat_url = 'https://raw.githubusercontent.com/pangeo-data/pangeo-datastore/master/intake-catalogs/master.yaml'
cat = intake.Catalog(cat_url)
ds = cat.atmosphere.gmet_v1.to_dask()
```

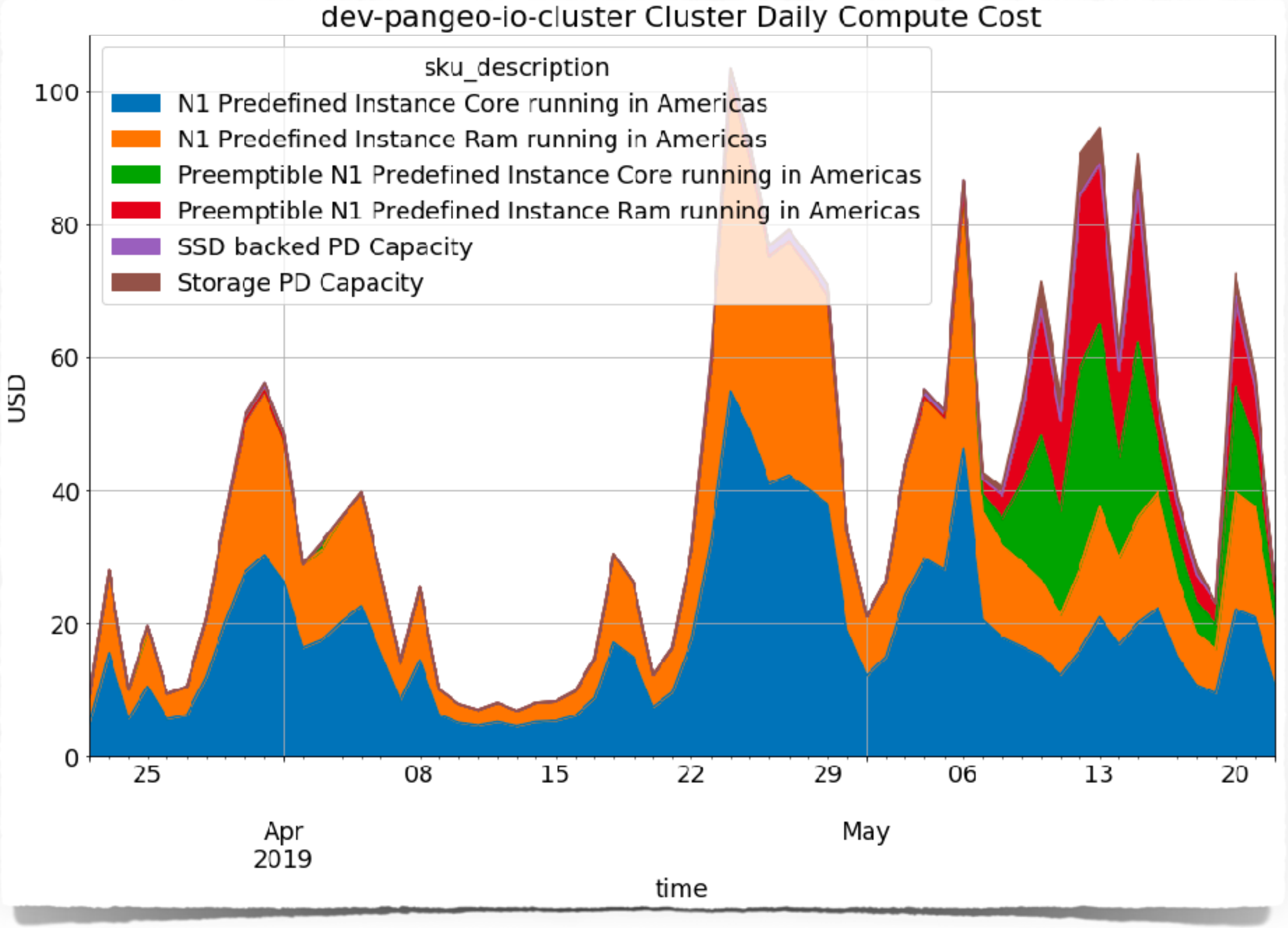
To explore the whole catalog, you can try:

```
python
cat.walk(depth=5)
```

- https://pangeo-data.github.io/pangeo-datastore/
- Datasets stored in **zarr** format (cloud-native HDF-replacement)
- Cataloged using **intake**
- Automated testing of datasets

CLOUD COSTS


sku_description
Class A Request Multi-Regional Storage
Class A Request Regional Storage
Class B Request Multi-Regional Storage
Class B Request Regional Storage
Download APAC
Download Australia
Download Worldwide Destinations (excluding Asia & Australia)
Multi-Regional Storage US
Regional Storage US
USD
0
50
100
150
200
250
300
350
Nov
Dec
Jan
2019
Feb
Mar
Apr
May
time


dev-pangeo-io-cluster Cluster Daily Compute Cost
sku_description
N1 Predefined Instance Core running in Americas
N1 Predefined Instance Ram running in Americas
Preemptible N1 Predefined Instance Core running in Americas
Preemptible N1 Predefined Instance Ram running in Americas
SSD backed PD Capacity
Storage PD Capacity
USD
0
20
40
60
80
100
25
08
15
22
29
06
13
20
Apr
2019
May
time

DEMO

https://tinyurl.com/pangeo-ocean

	Government HPC	Commercial Cloud
Access	✅ Available to all federally funded projects ❌ Available *only* to federally funded projects	✅ Available globally to anyone with a credit card ❌ Authentication is not integrated with existing research infrastructure
Cost	✅ Cost is hidden from researchers and billed by funding agencies ❌ Allocations, quotas, limits	❌ Cost is borne by individual researchers and hidden from funding agencies ✅ Economics of scale, unlimited resources
Compute	✅ Homogeneous, high performance nodes ❌ Queues, batch scheduling, ssh access ❌ Fixed-size compute	✅ Flexible hardware (big, small, GPU) ✅ Instant provisioning of unlimited resources ✅ Spot market: burstable, volatile
Storage	✅ Fast parallel filesystems (e.g. GPFS)	✅ Fast object storage

HOW TO GET INVOLVED

[HTTP://PANGEO.IO](http://pangeo.io)

- Use and contribute to xarray, dask, zarr, jupyterhub, etc.
- Access an existing Pangeo deployment on an HPC cluster, or cloud resources (http://pangeo.io/deployments.html)
- Adapt Pangeo elements to meet your projects needs (data portals, etc.) and give feedback via github: github.com/pangeo-data/pangeo
- Provide data in a cloud-optimized format

The following slide is a backup in case live demo is not possible.

JupyterLab

https://ocean.pangeo.io/user/rabernat/lab?redirects=1

File Edit View Run Kernel Hub Tabs Settings Help

cesm-pop-highres-ocean

Code

Python 3

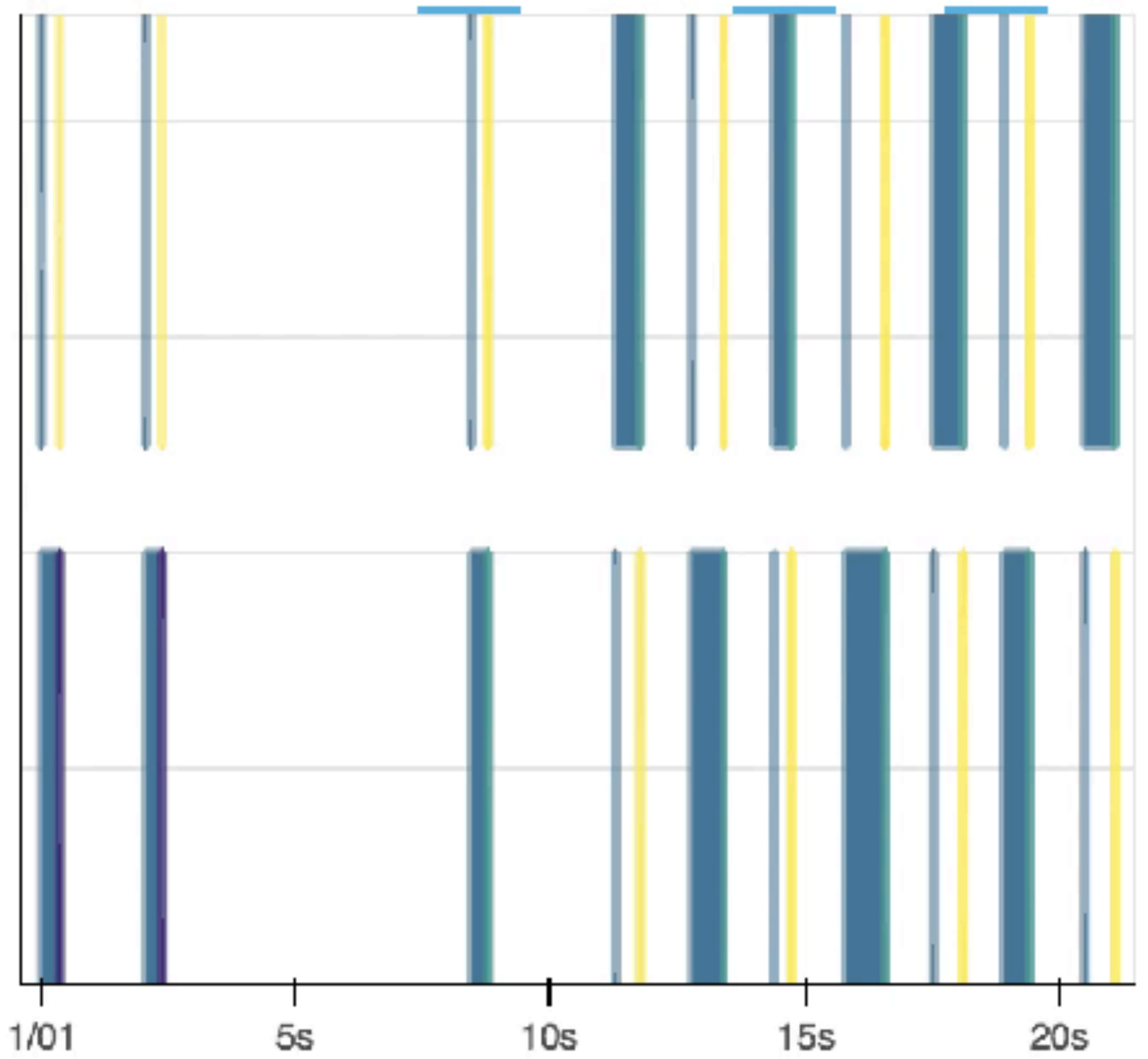
```
[5]: <xarray.DataArray 'SST' (time: 14965, nlat: 2400, nlon: 3600)>
dask.array<shape=(14965, 2400, 3600), dtype=float32, chunksize=(1, 2400, 3600)>
Coordinates:
  * time     (time) float64 1.679e+04 1.679e+04 ... 3.175e+04 3.176e+04
  * nlon     (nlon) int64 0 1 2 3 4 5 6 7 ... 3593 3594 3595 3596 3597 3598 3599
  * nlat     (nlat) int64 0 1 2 3 4 5 6 7 ... 2393 2394 2395 2396 2397 2398 2399
    lon      (nlat, nlon) float64 dask.array<shape=(2400, 3600), chunksize=(2400, 3600)>
    lat      (nlat, nlon) float64 dask.array<shape=(2400, 3600), chunksize=(2400, 3600)>
Attributes:
    cell_methods:  time: mean
    grid_loc:      2110
    long_name:     Surface Potential Temperature
    units:         degC
```

```
[ ]: %%output holomap='scrubber' fps=1

sst_ds = hv.Dataset(sst, kdims=['time', 'nlon', 'nlat'])
hv_sst = sst_ds.to(hv.Image, kdims=["nlon", "nlat"], dynamic=True)

%opts Image [width=700 height=400] (cmap='magma')
regrid(hv_sst, precompute=True)
```

```
[ ]:
```

Dask Task Stream

Task Stream



Dask Progress

Progress -- total: 0, in-memory: 0, processing: 0, erred: 0

- 12 GB / 24 hours -> must be a mistake