



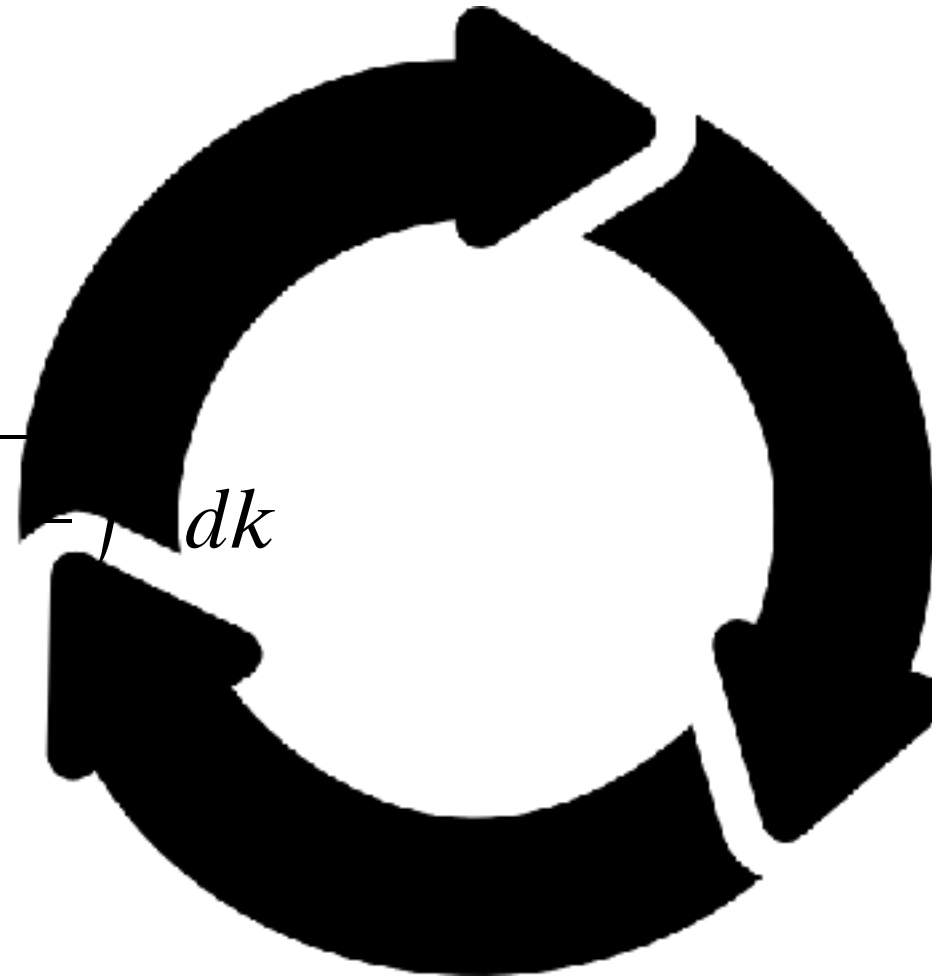
PANGEO

A COMMUNITY-DRIVEN EFFORT FOR
BIG DATA GEOSCIENCE

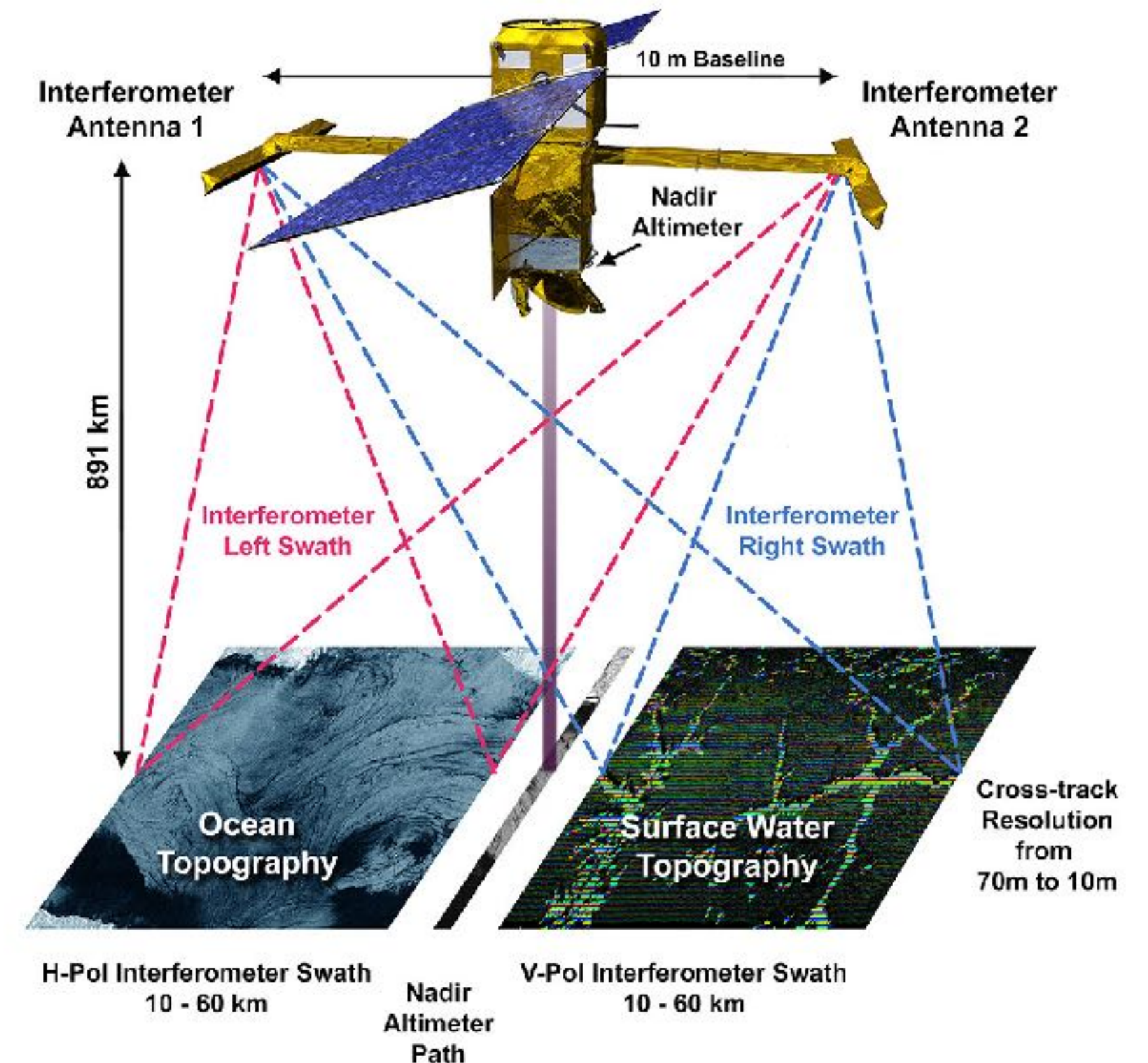
WHAT DRIVES PROGRESS IN OCEANOGRAPHY?

New Ideas

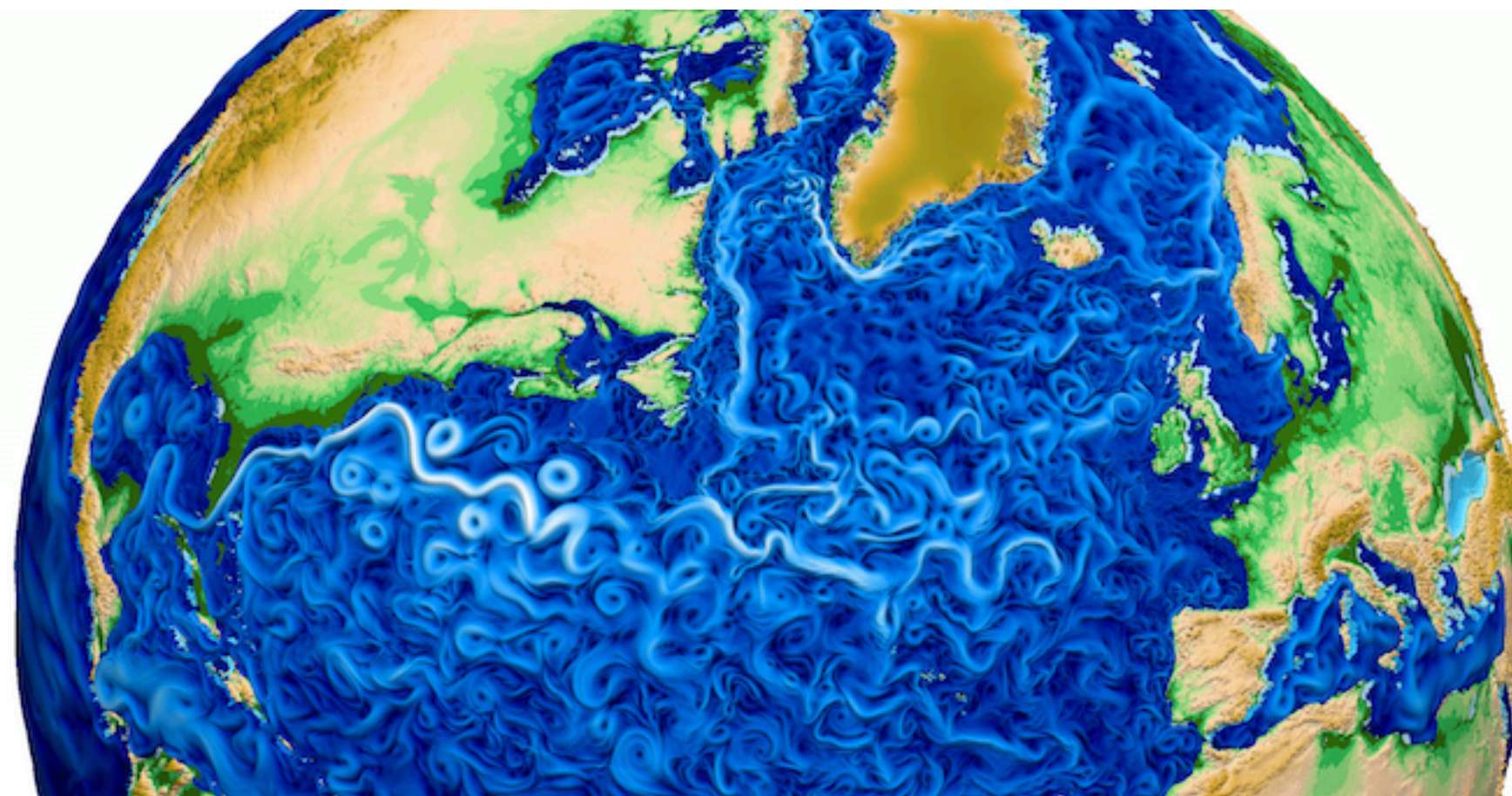
$$E = \frac{\rho_0 |\mathbf{U}|}{\pi} \int_{|f|/|\mathbf{U}|}^{N/|\mathbf{U}|} P_{1D}(k) \sqrt{N^2 - |\mathbf{U}|^2 k^2} \sqrt{|\mathbf{U}|^2 k^2} dk$$

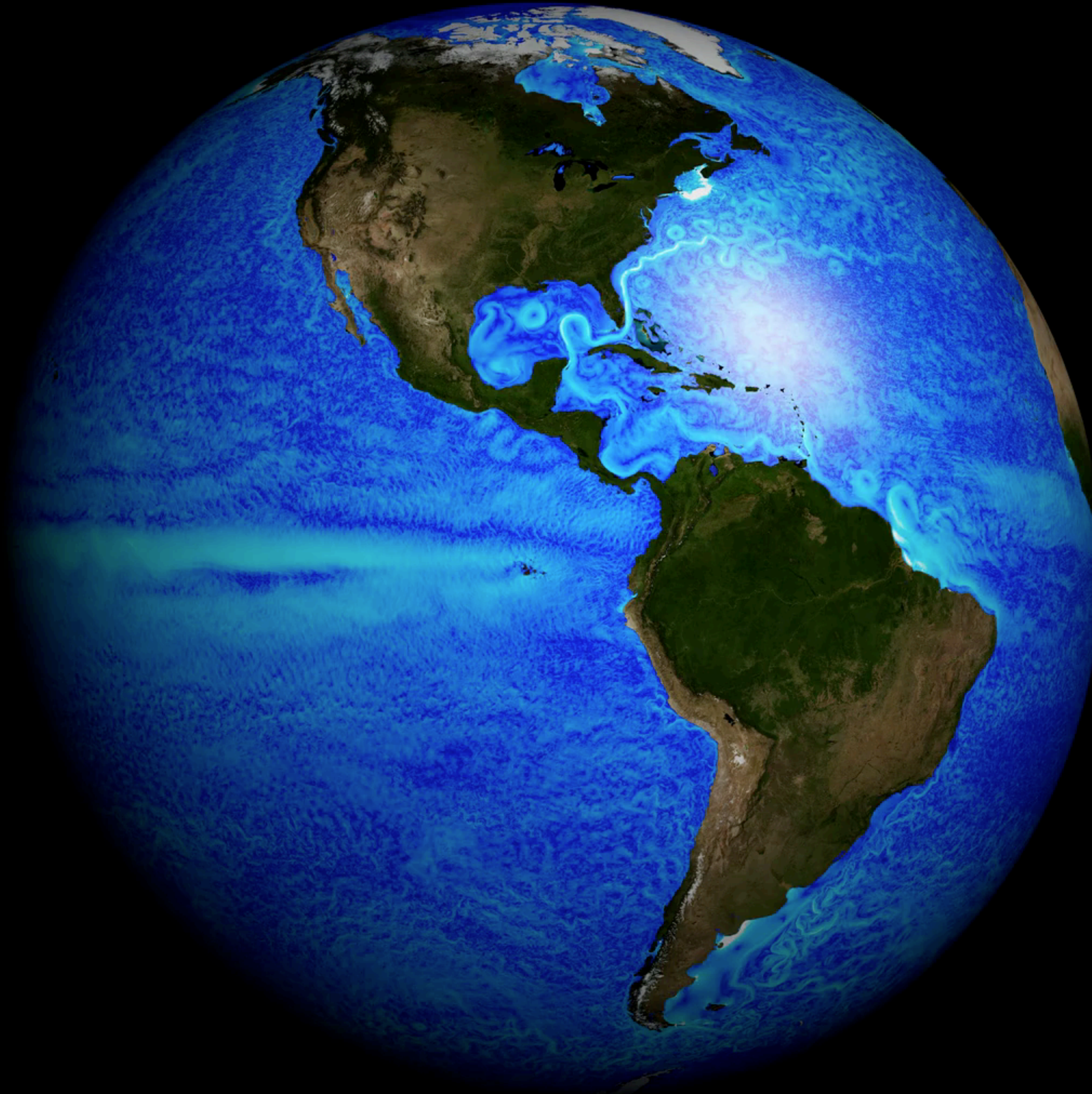


New Observations



New Simulations





MITgcm LLC4320 Simulation

Grid resolution:
1 - 2 km

Single 3D scalar field:
80 GB

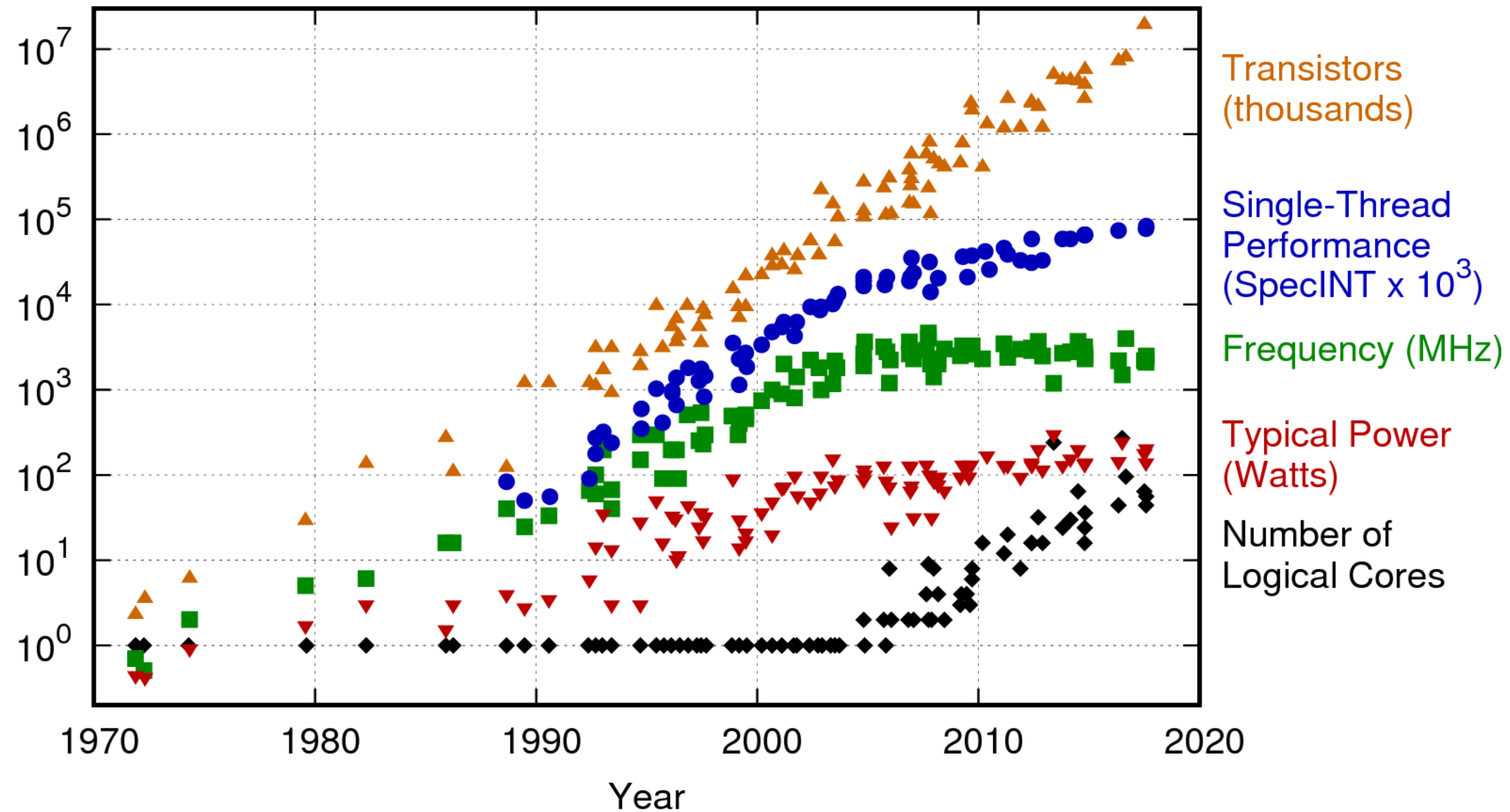
Output frequency:
1 hour

Simulation Length:
1 year

Output data volume:
2 PB

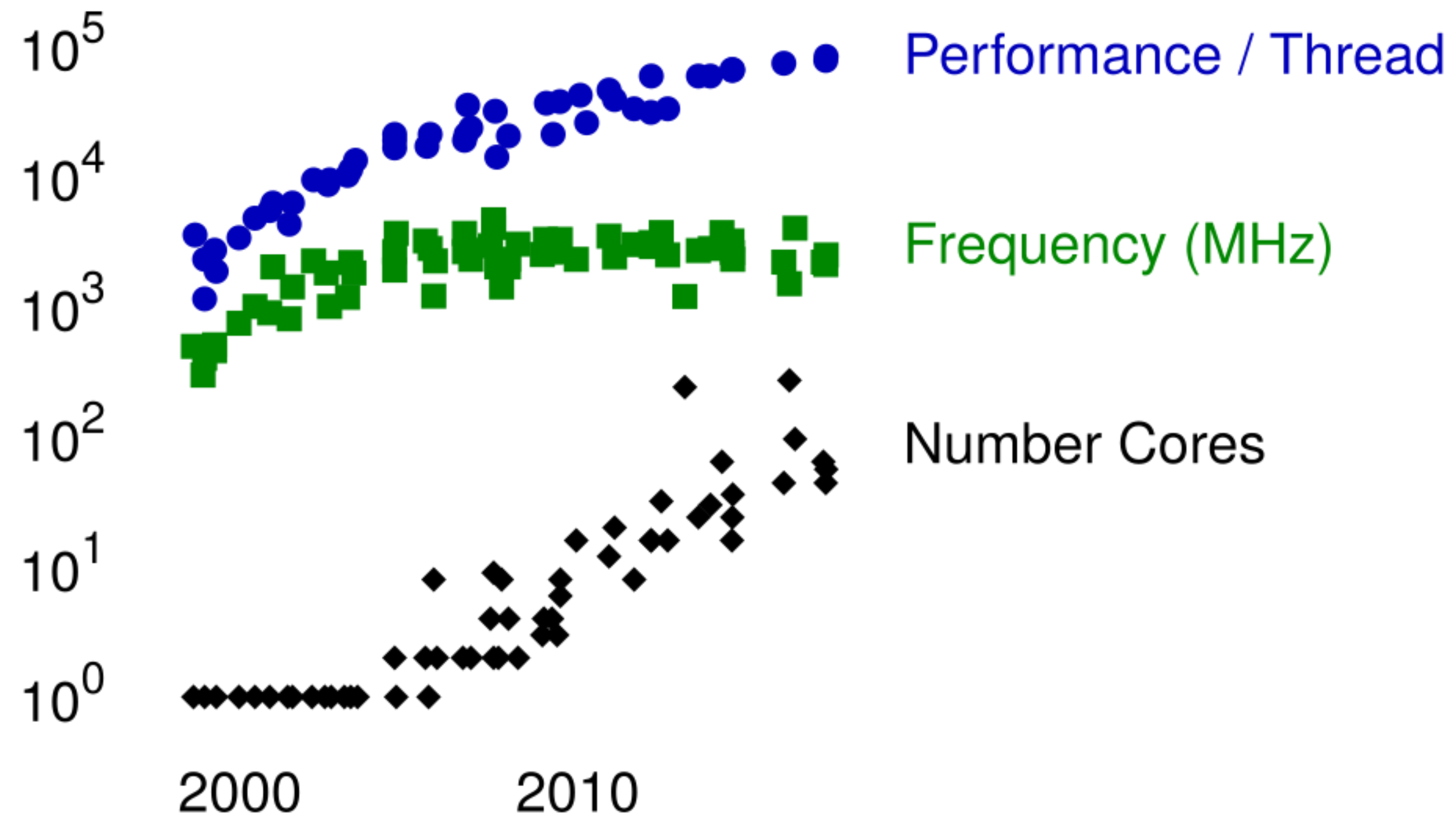
COMPUTER ARCHITECTURE

42 Years of Microprocessor Trend Data



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

COMPUTER ARCHITECTURE



COMPUTER ARCHITECTURE: SIMULATION

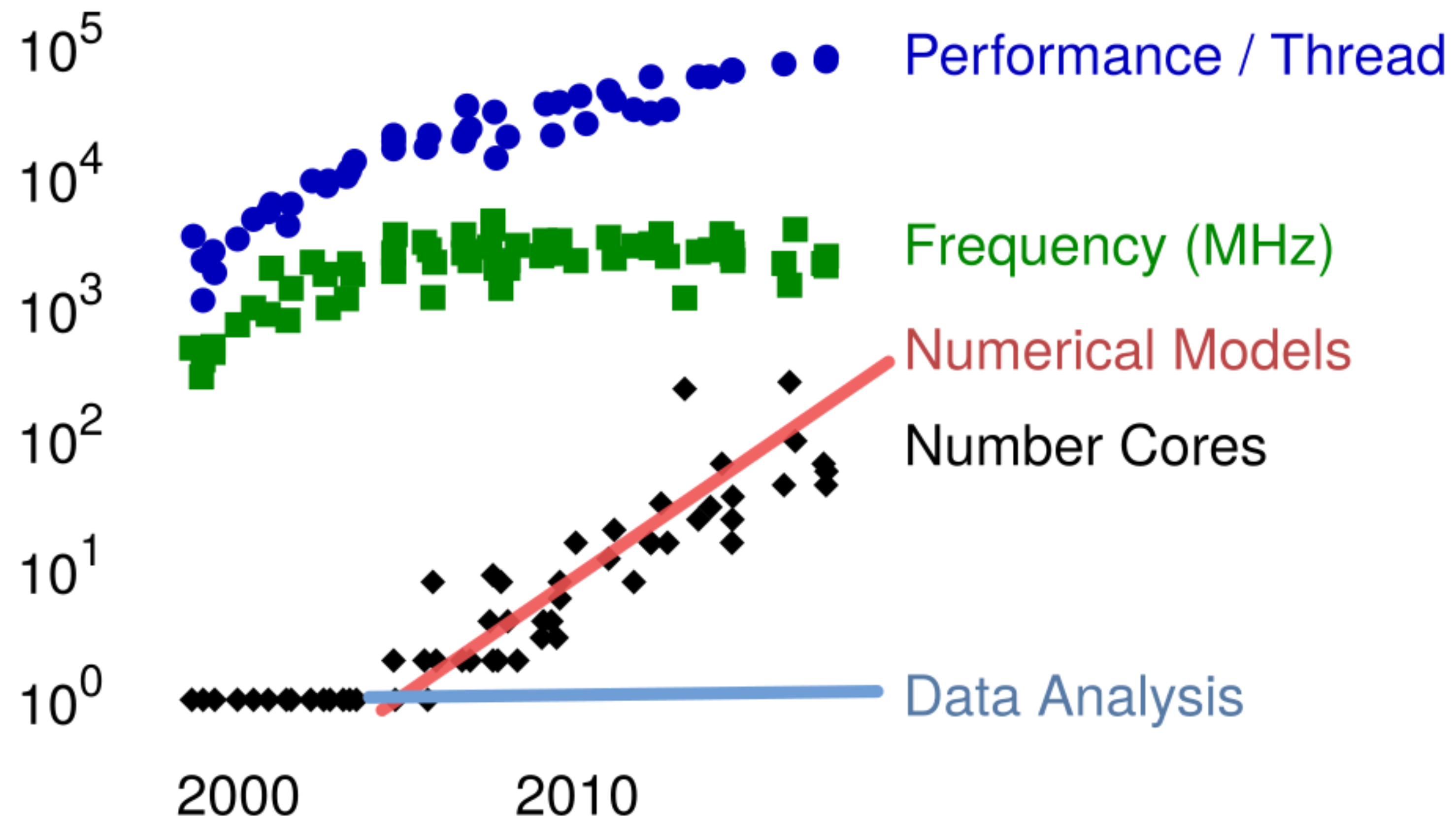


NASA Pleiades Supercomputer

COMPUTER ARCHITECTURE: ANALYSIS AND VISUALIZATION

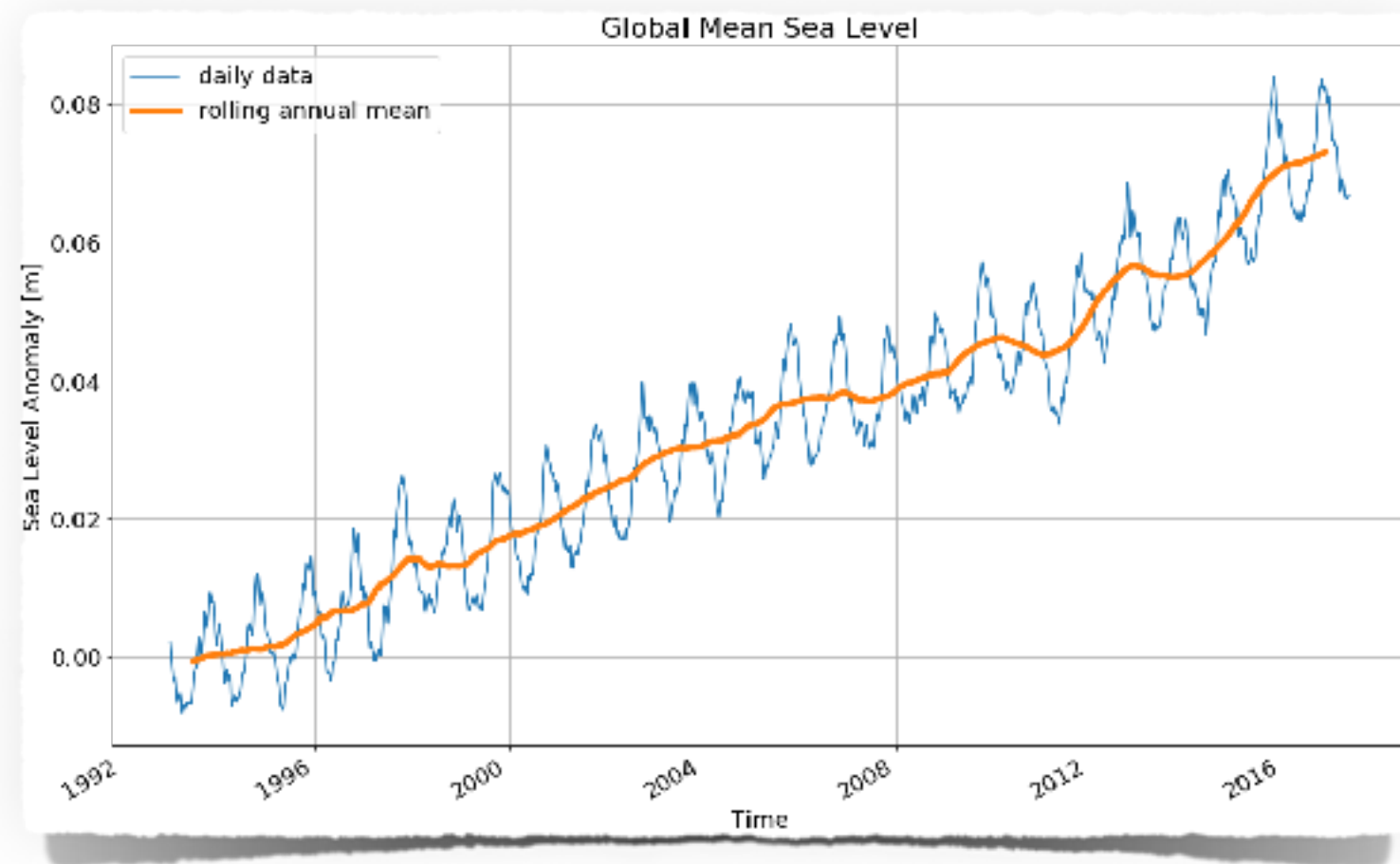


COMPUTER ARCHITECTURE

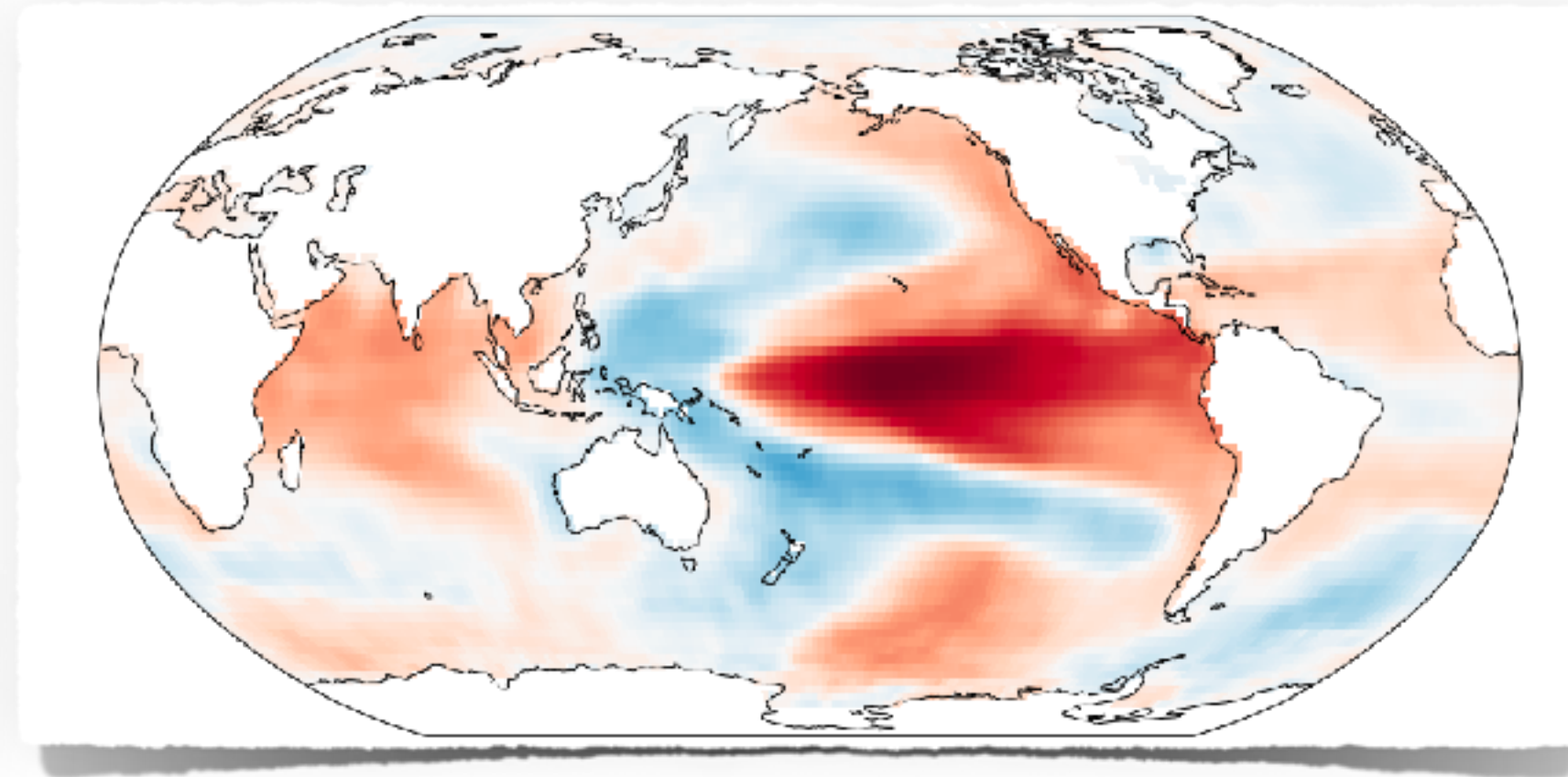


WHAT SCIENCE DO WE WANT TO DO WITH CLIMATE DATA?

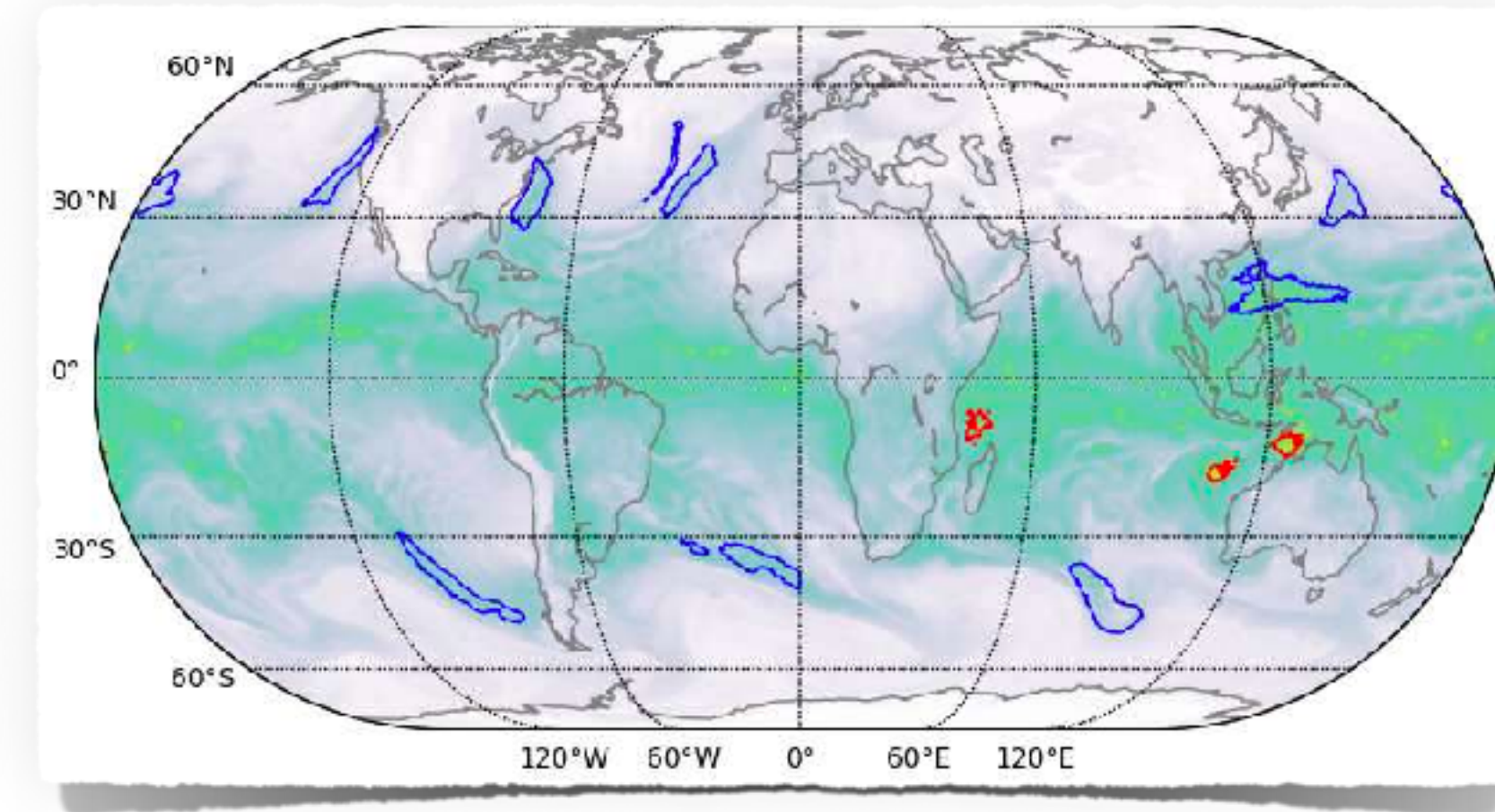
Take the mean!



Analyze spatiotemporal variability



Machine learning!



Need to process all the data!

PANGEO CHALLENGE

*How can we develop **flexible** analysis tools that meet our community's **diverse needs** and **scale** to Petabyte-sized datasets?*

WHAT IS PANGEO?

“A community platform for Big Data geoscience”

- Open Community
- Open Source Software
- Open Source Infrastructure

PANGEO COMMUNITY



Lamont-Doherty Earth Observatory
COLUMBIA UNIVERSITY | EARTH INSTITUTE



Met Office



developmentSEED



RHODIUM GROUP



CLIMACELL
Weather Revealed

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

PANGEO FUNDING



Google Cloud Platform



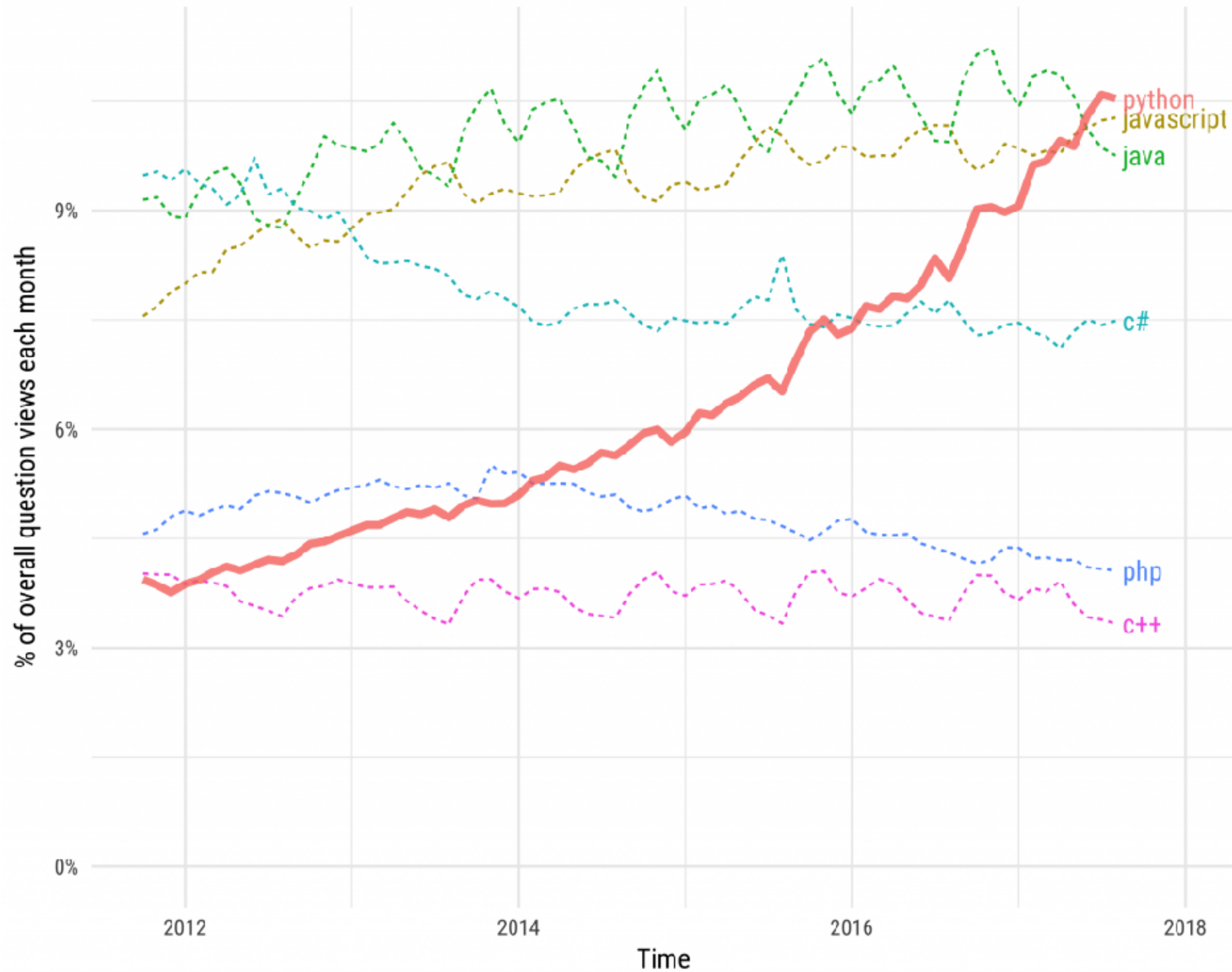
[HTTP://PANGEO.IO](http://PANGEO.IO)

PANGEO SOFTWARE

SCIENTIFIC PYTHON FOR DATA SCIENCE

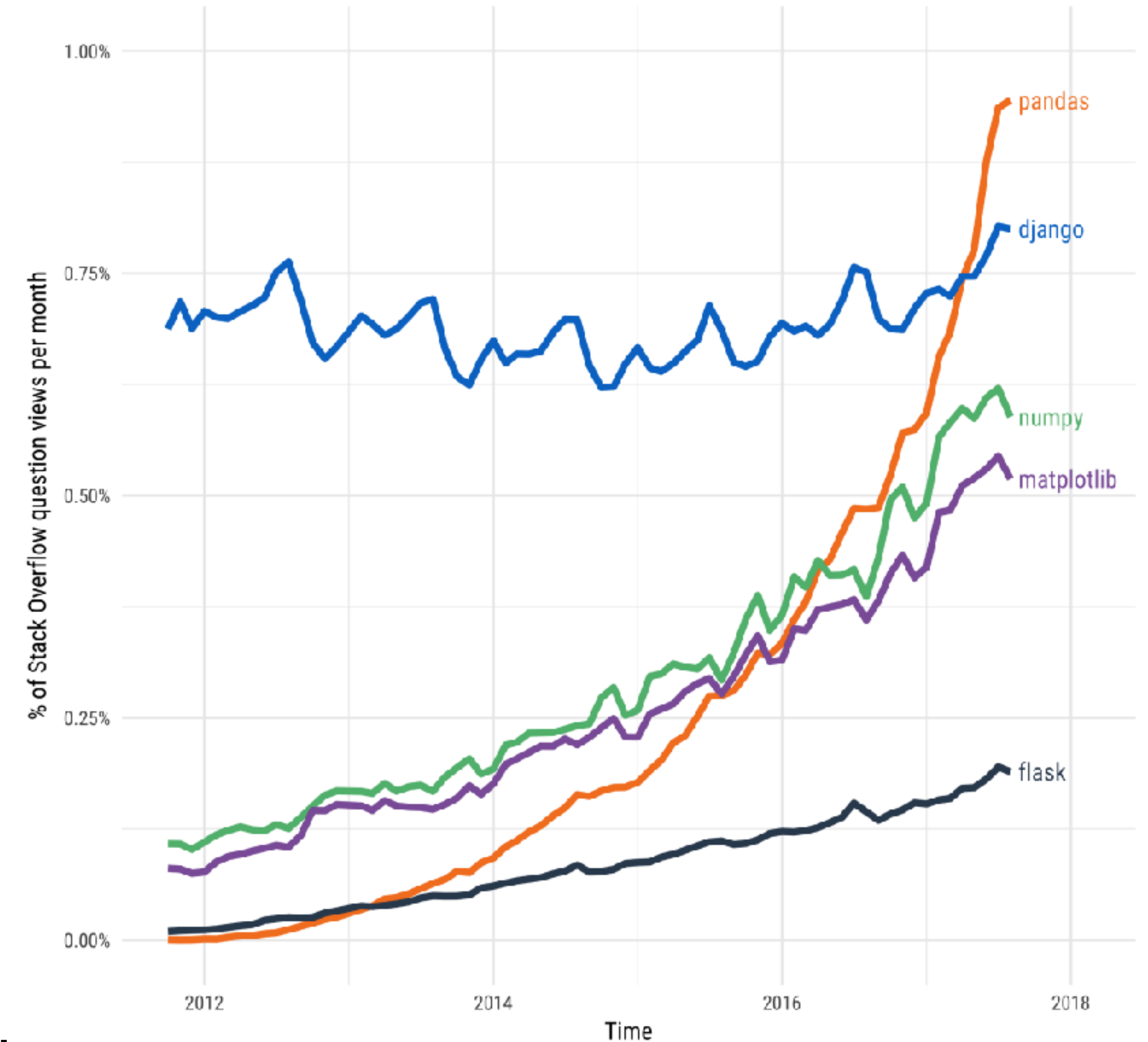
Growth of major programming languages

Based on Stack Overflow question views in World Bank high-income countries



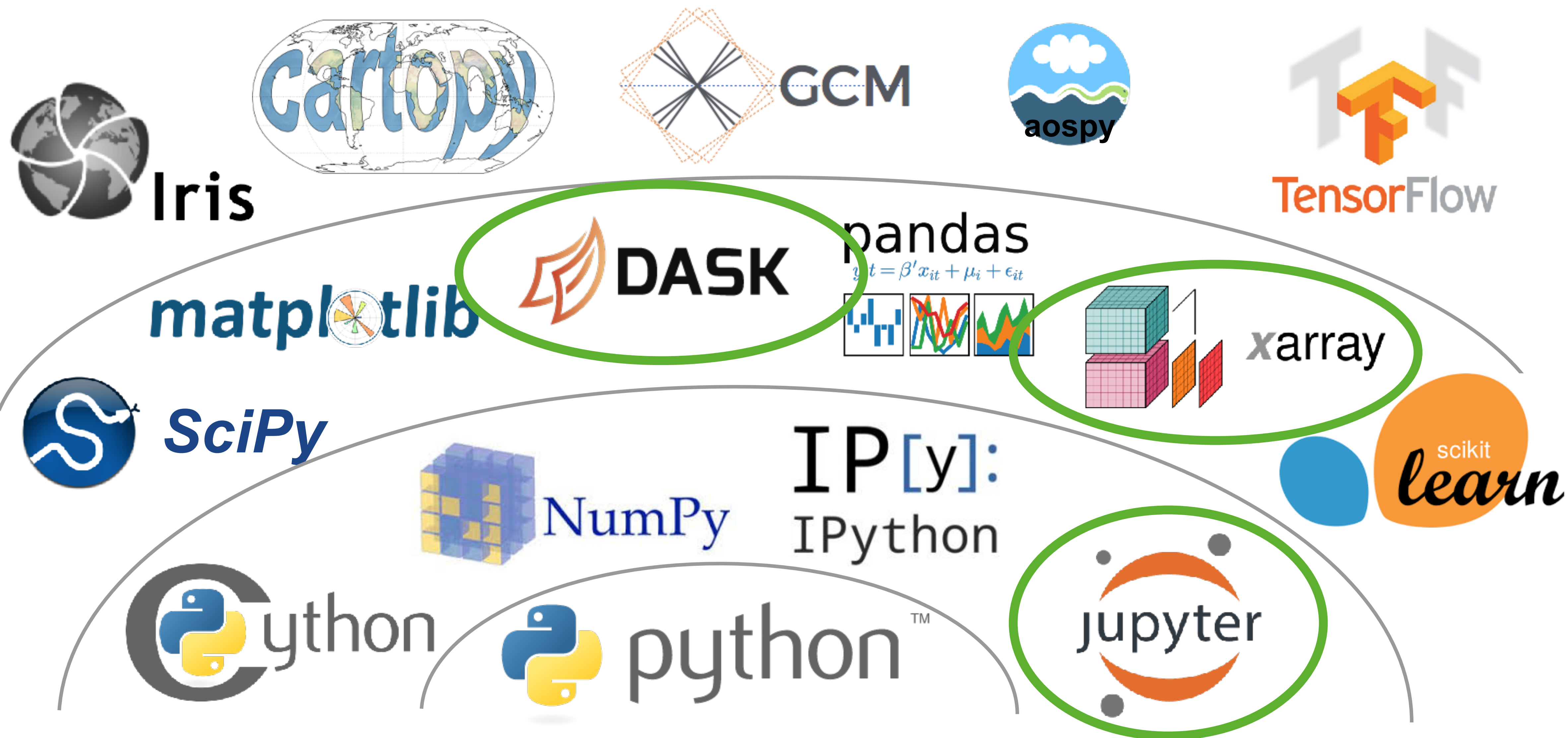
Stack Overflow Traffic to Questions About Selected Python Packages

Based on visits to Stack Overflow questions from World Bank high-income countries



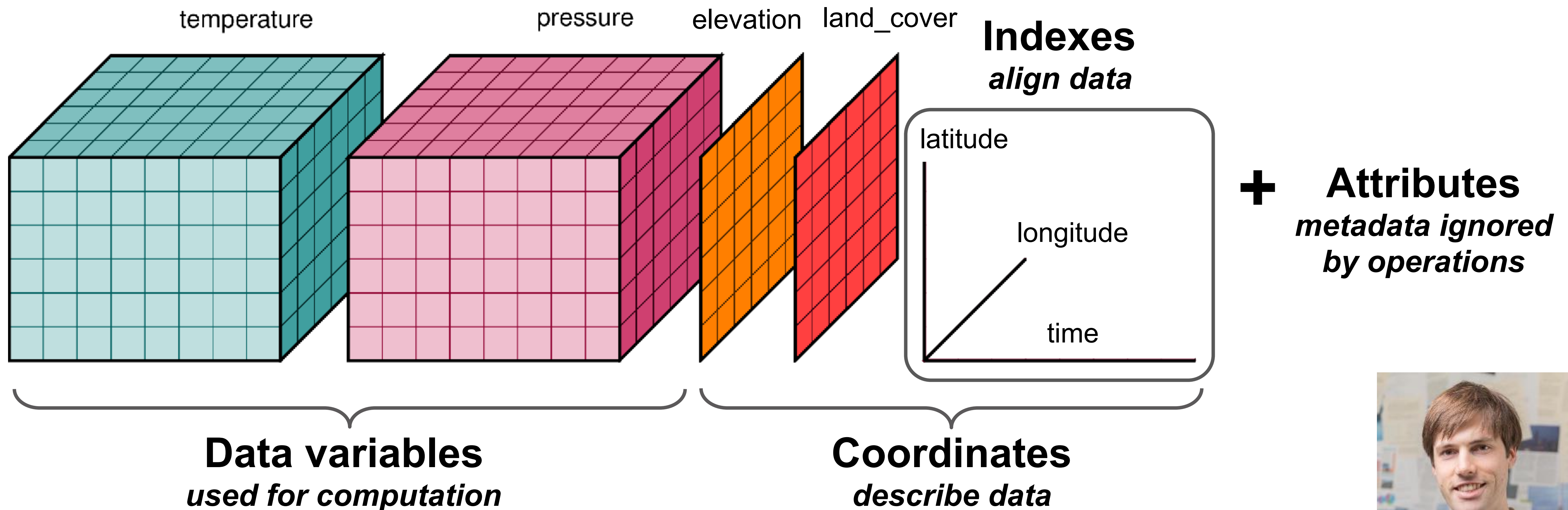
source: stackoverflow.com

SCIENTIFIC PYTHON FOR CLIMATE



XARRAY

<https://github.com/pydata/xarray>



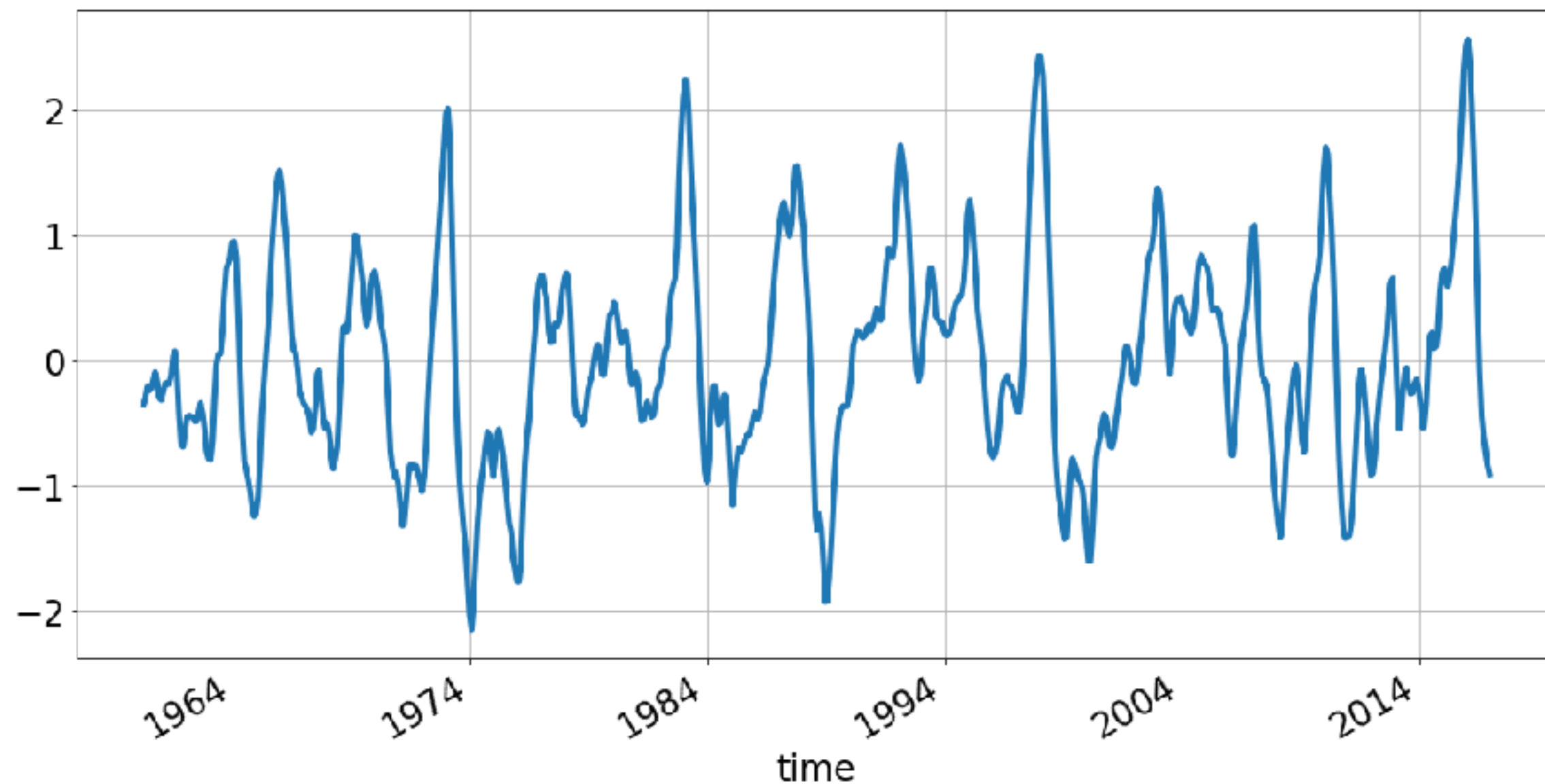
“netCDF meets pandas.DataFrame”



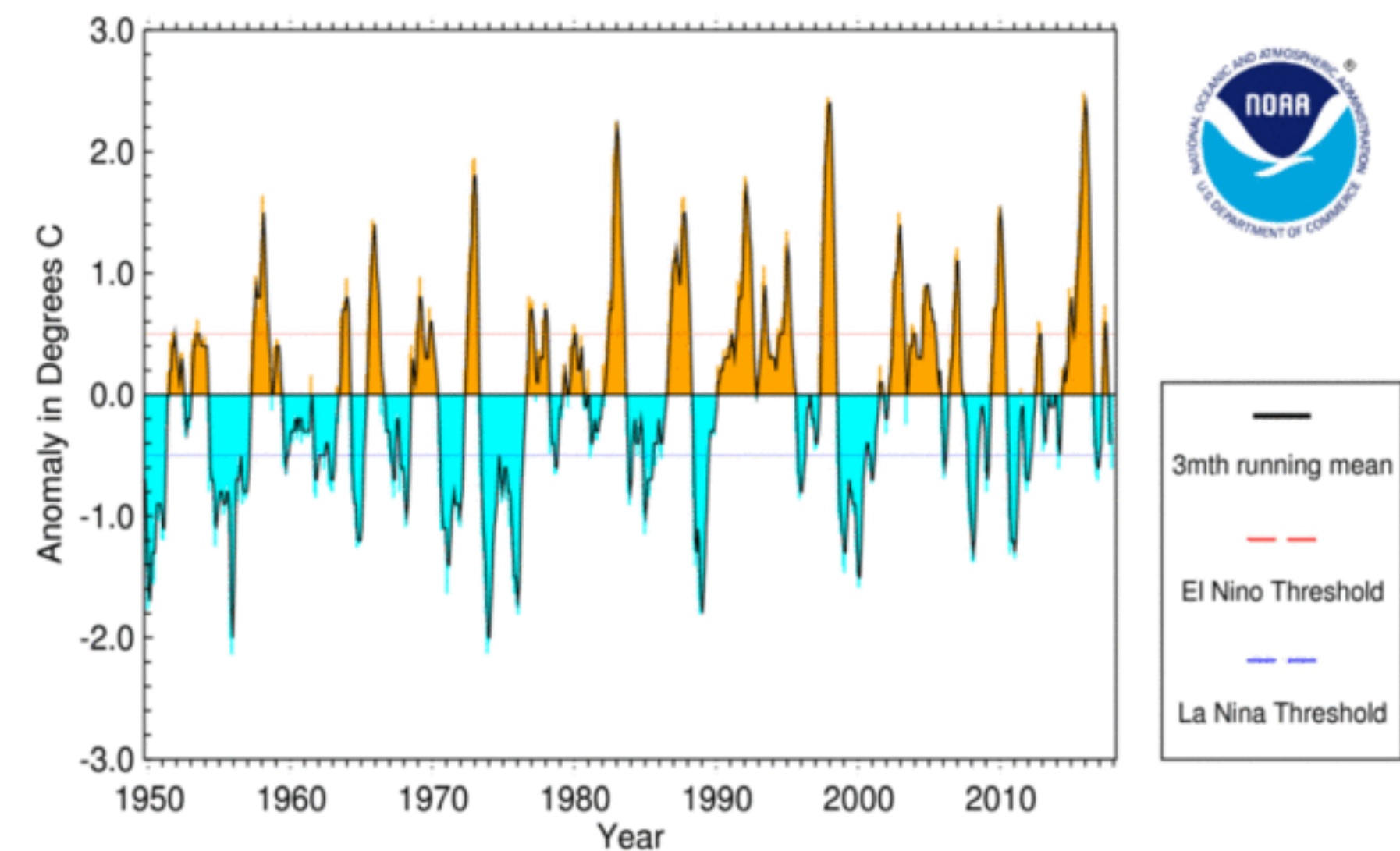
Stephan Hoyer
(Google Research)

XARRAY: EXPRESSIVE & HIGH-LEVEL

```
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()
```

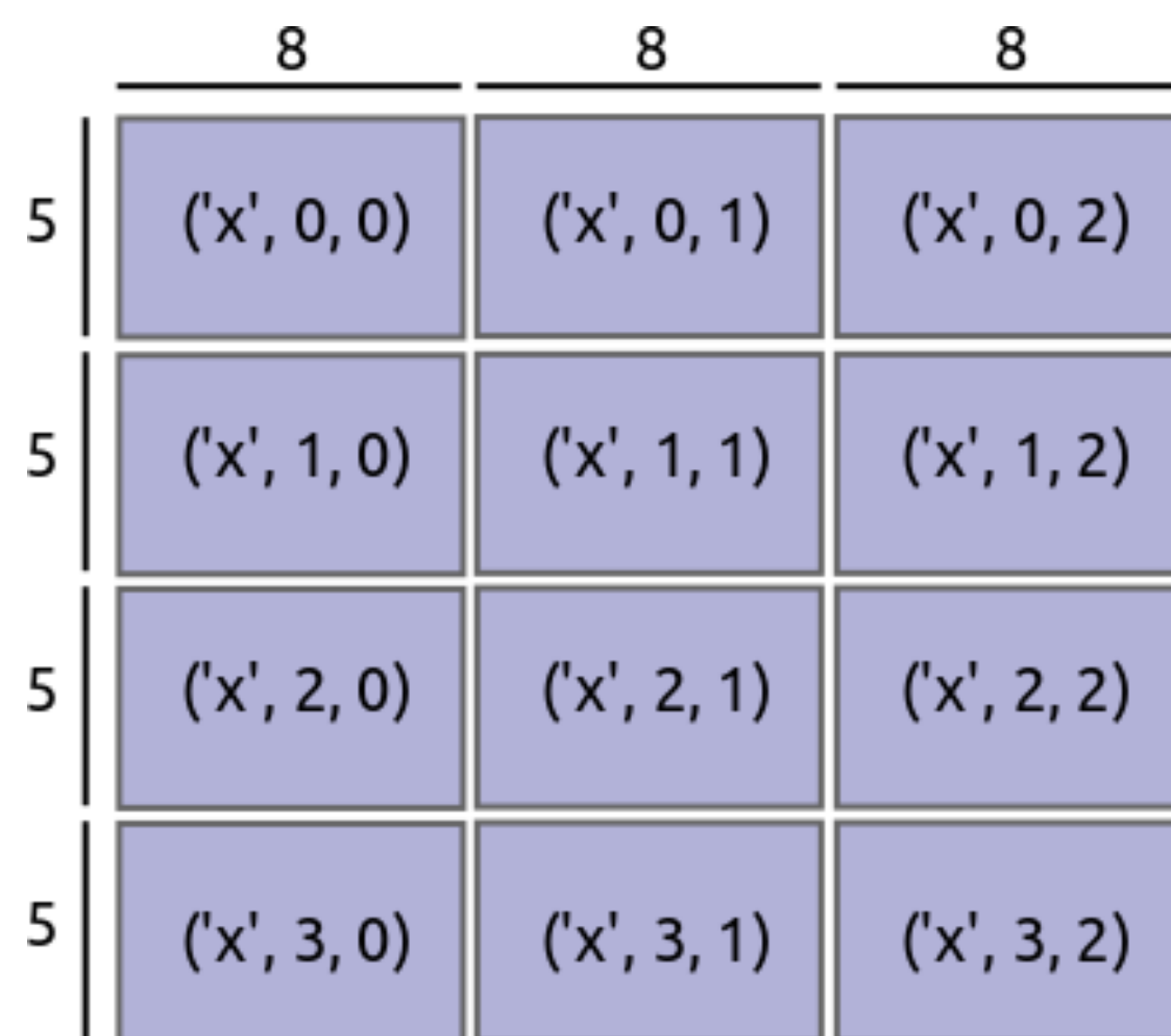


SST Anomaly in Nino 3.4 Region (5N-5S,120-170W)

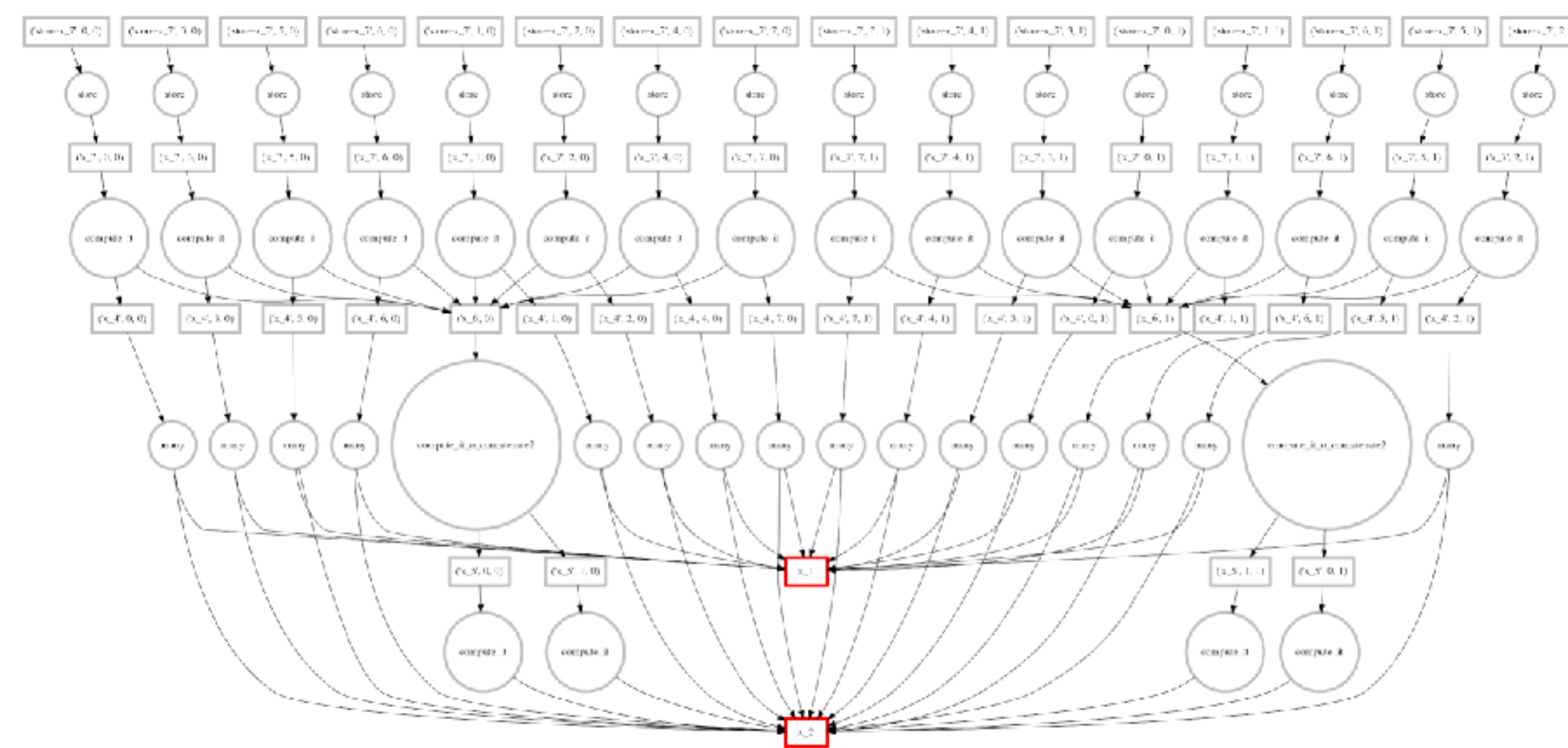


DASK

<https://github.com/dask/dask/>



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



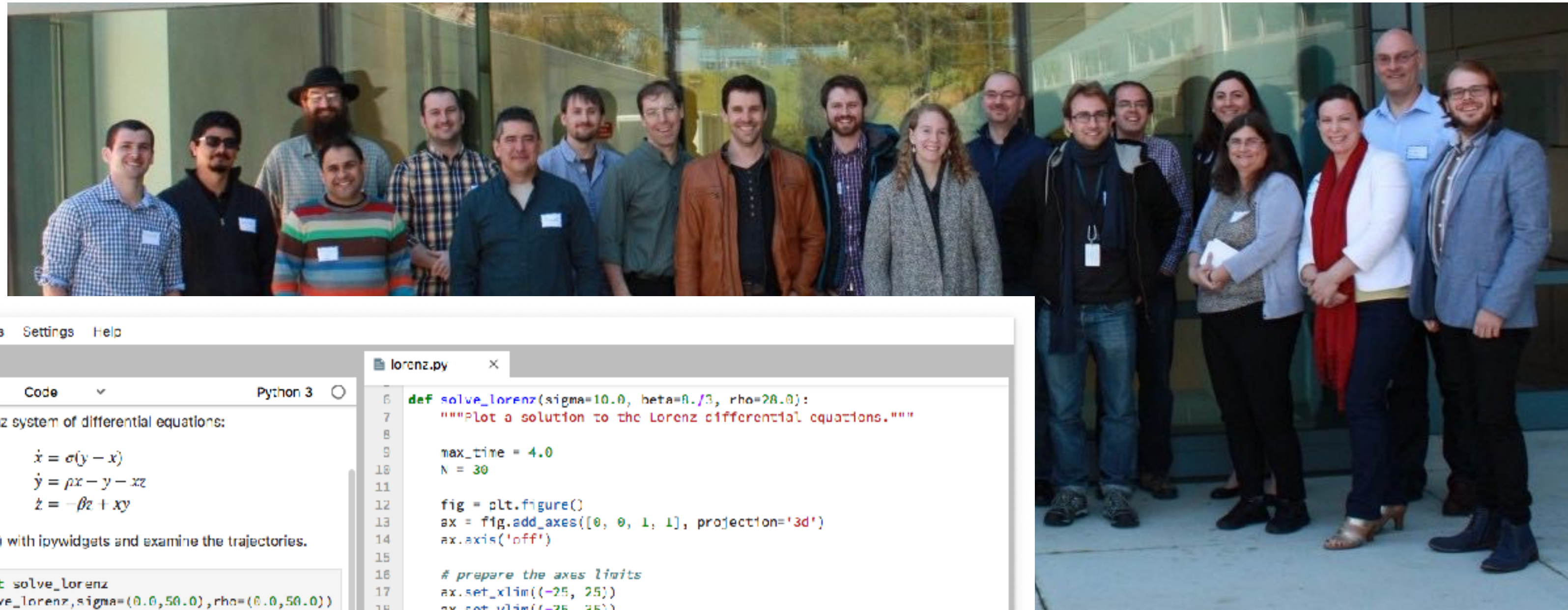
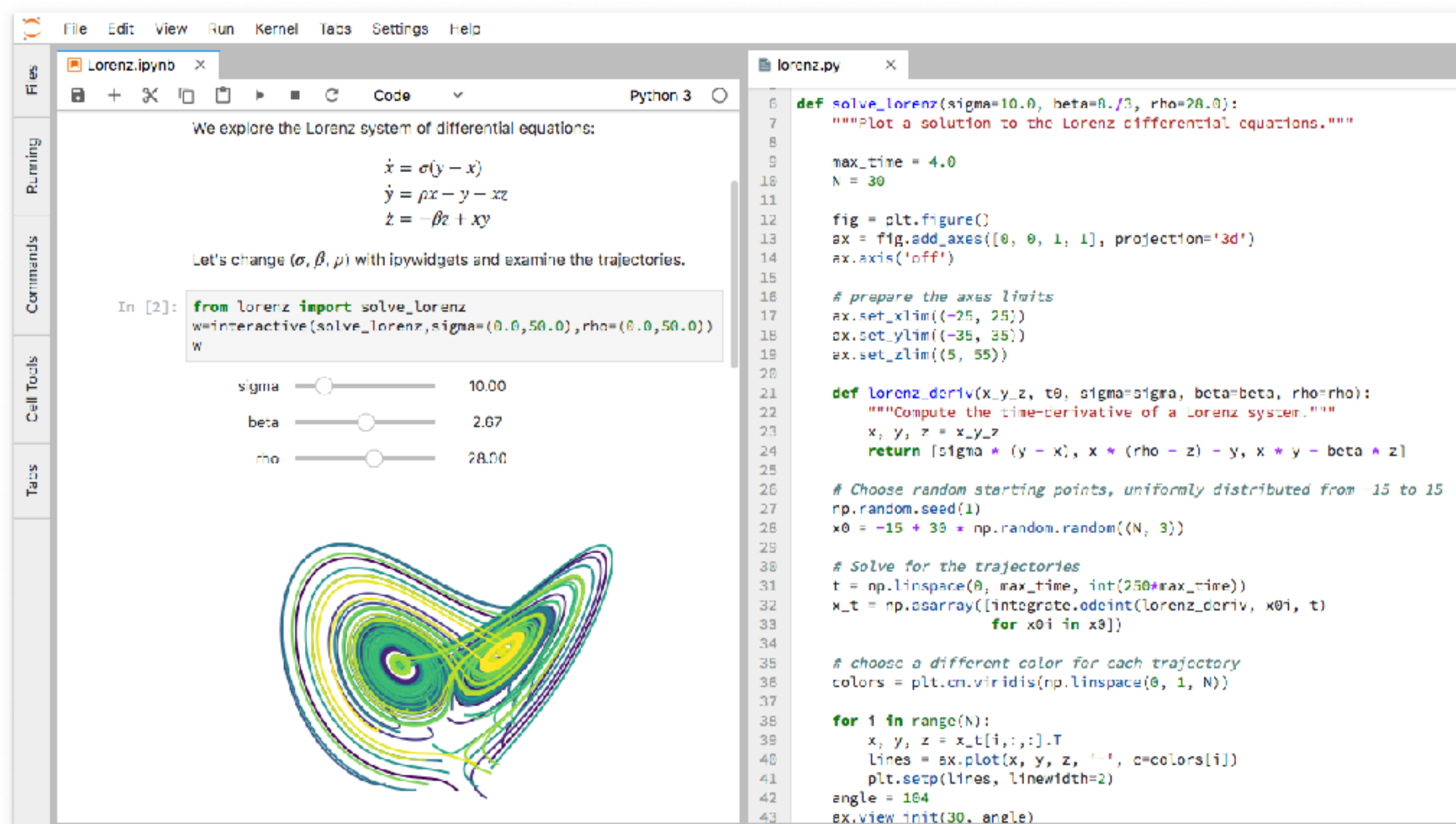
Complex computations represented as a graph of individual tasks.

Scheduler optimizes execution of graph.



Matt Rocklin (NVIDIA)

JUPYTER

The screenshot displays the JupyterLab environment. On the left, a notebook titled 'Lorenz.ipynb' is open, showing the following text:

We explore the Lorenz system of differential equations:

$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy\end{aligned}$$

Let's change (σ, β, ρ) with ipywidgets and examine the trajectories.

In [2]: `from lorenz import solve_lorenz`
`w=interactive(solve_lorenz, sigma=(0.0, 50.0), rho=(0.0, 50.0))`
`w`

Below the code, three interactive sliders are shown for the parameters:

- sigma: 10.00
- beta: 2.07
- rho: 28.00

At the bottom of the notebook, a 3D plot of the Lorenz attractor is displayed, showing its characteristic butterfly-like shape with multiple trajectories in various colors.

On the right, a code editor window titled 'lorenz.py' shows the following Python code:

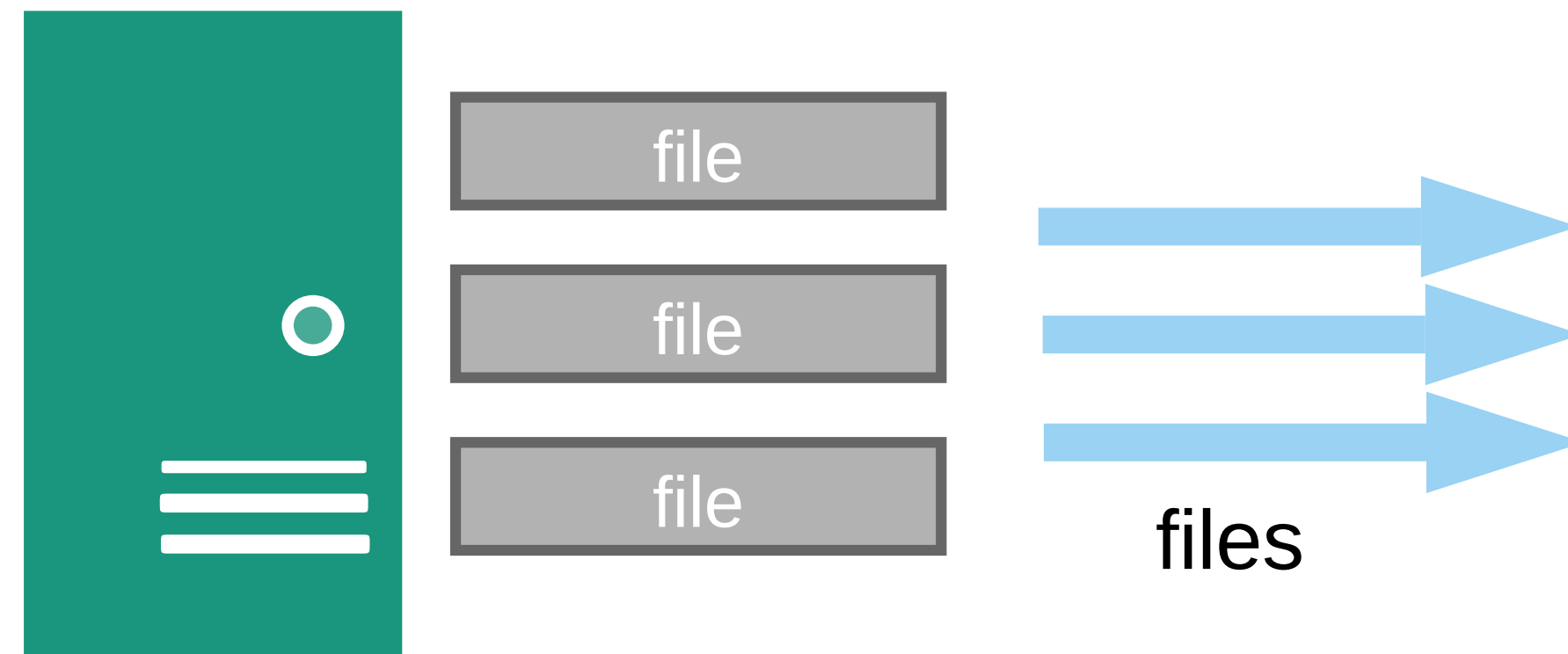
```
6 def solve_lorenz(sigma=10.0, beta=8./3, rho=28.0):
7     """Plot a solution to the Lorenz differential equations."""
8
9     max_time = 4.0
10    N = 30
11
12    fig = plt.figure()
13    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
14    ax.axis('off')
15
16    # prepare the axes limits
17    ax.set_xlim((-25, 25))
18    ax.set_ylim((-35, 35))
19    ax.set_zlim((5, 55))
20
21    def Lorenz_deriv(x,y,z, t0, sigma=sigma, beta=beta, rho=rho):
22        """Compute the time-derivative of a Lorenz system."""
23        x, y, z = x,y,z
24        return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]
25
26    # Choose random starting points, uniformly distributed from -15 to 15
27    np.random.seed(1)
28    x0 = -15 + 30 * np.random.random((N, 3))
29
30    # Solve for the trajectories
31    t = np.linspace(0, max_time, int(250*max_time))
32    x_t = np.asarray([integrate.odeint(Lorenz_deriv, x0i, t)
33                      for x0i in x0])
34
35    # choose a different color for each trajectory
36    colors = plt.cm.viridis(np.linspace(0, 1, N))
37
38    for i in range(N):
39        x, y, z = x_t[i, :, :].T
40        lines = ax.plot(x, y, z, '-', c=colors[i])
41        plt.setp(lines, linewidth=2)
42    angle = 104
43    ax.view_init(30, angle)
```

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

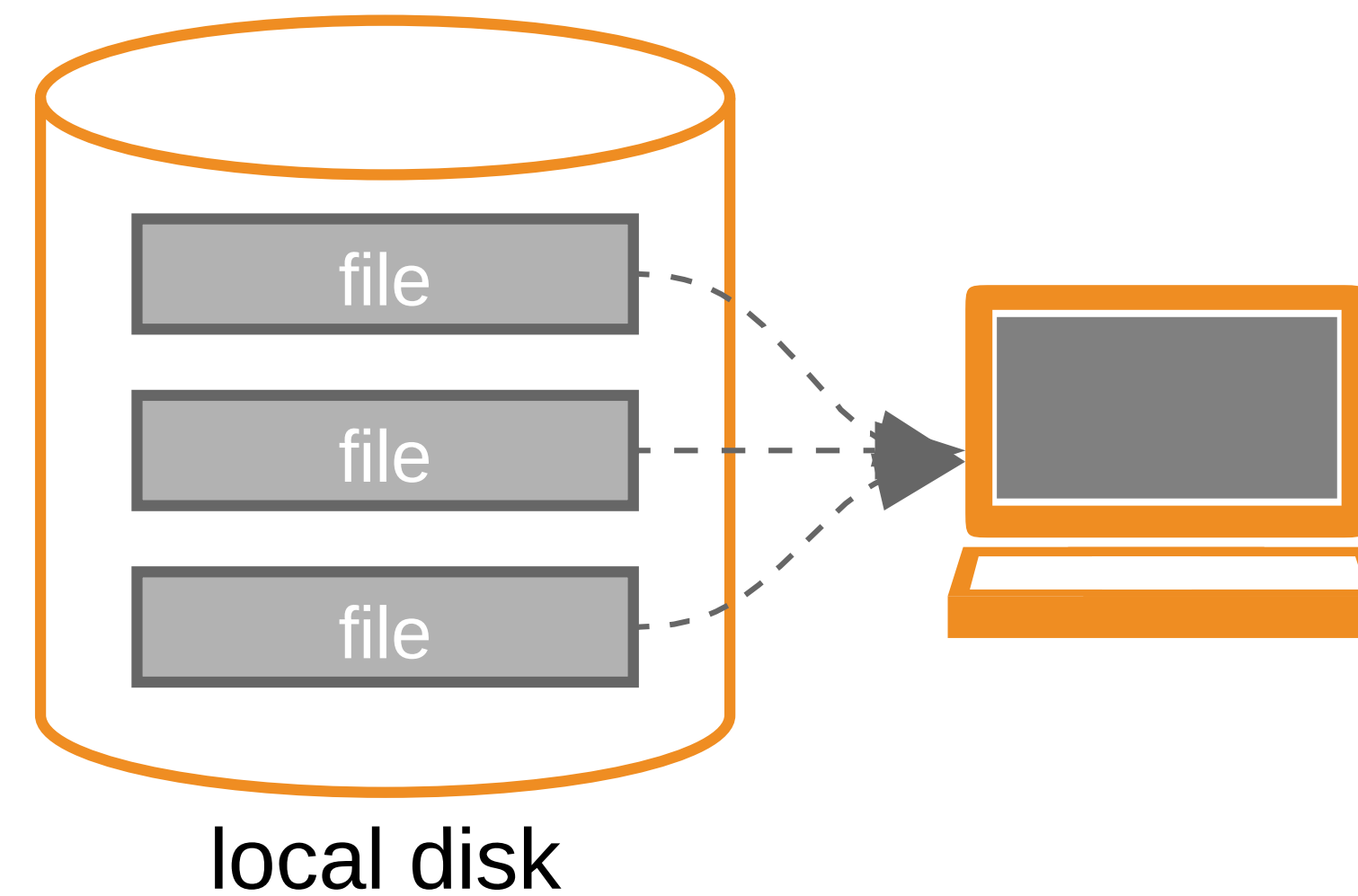
PANGEO INFRASTRUCTURE

FILE-BASED APPROACH

step 1: download



step 2: analyze



Data provider's responsibilities

End user's responsibilities

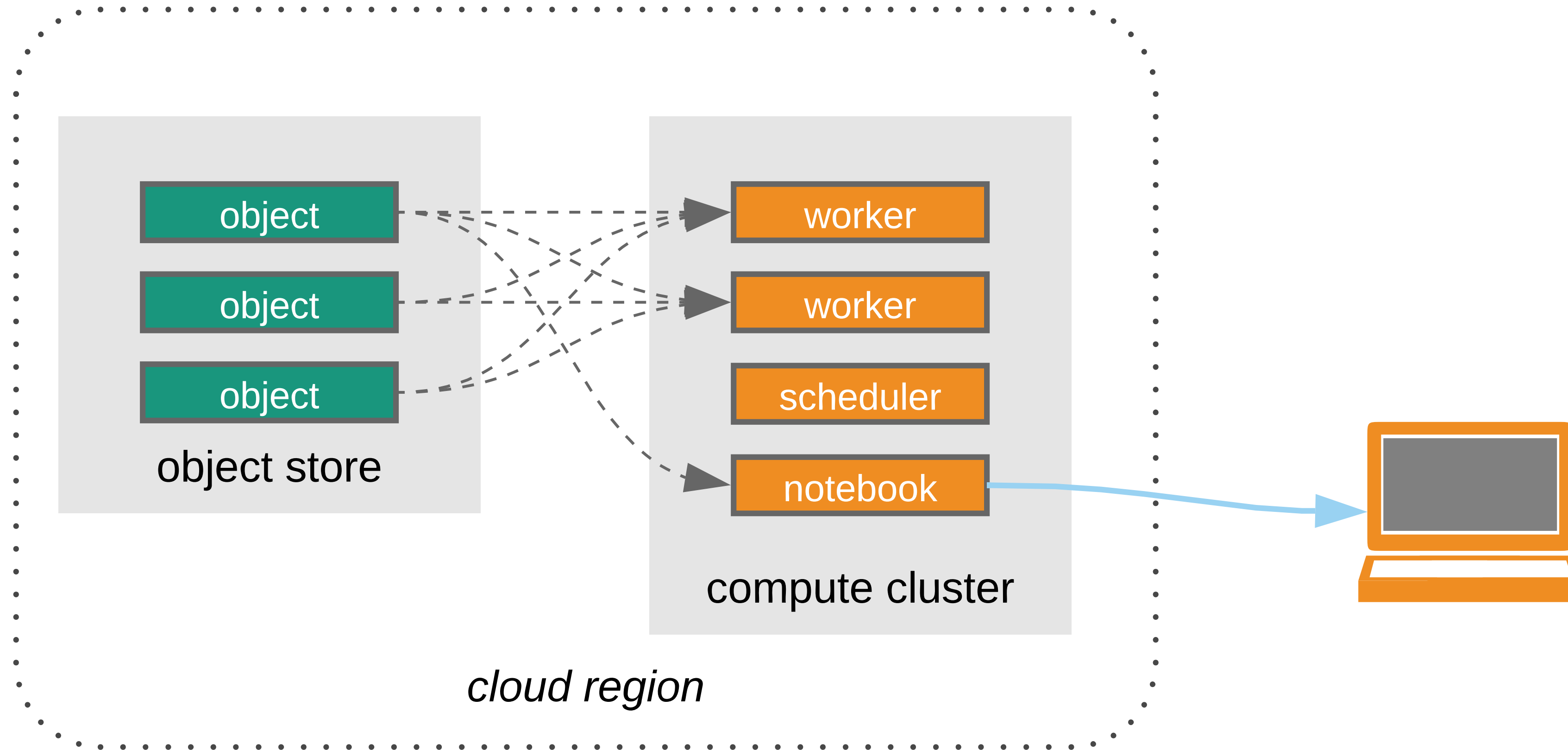
SERVER-SIDE DATABASE



Data provider's responsibilities

End user's responsibilities

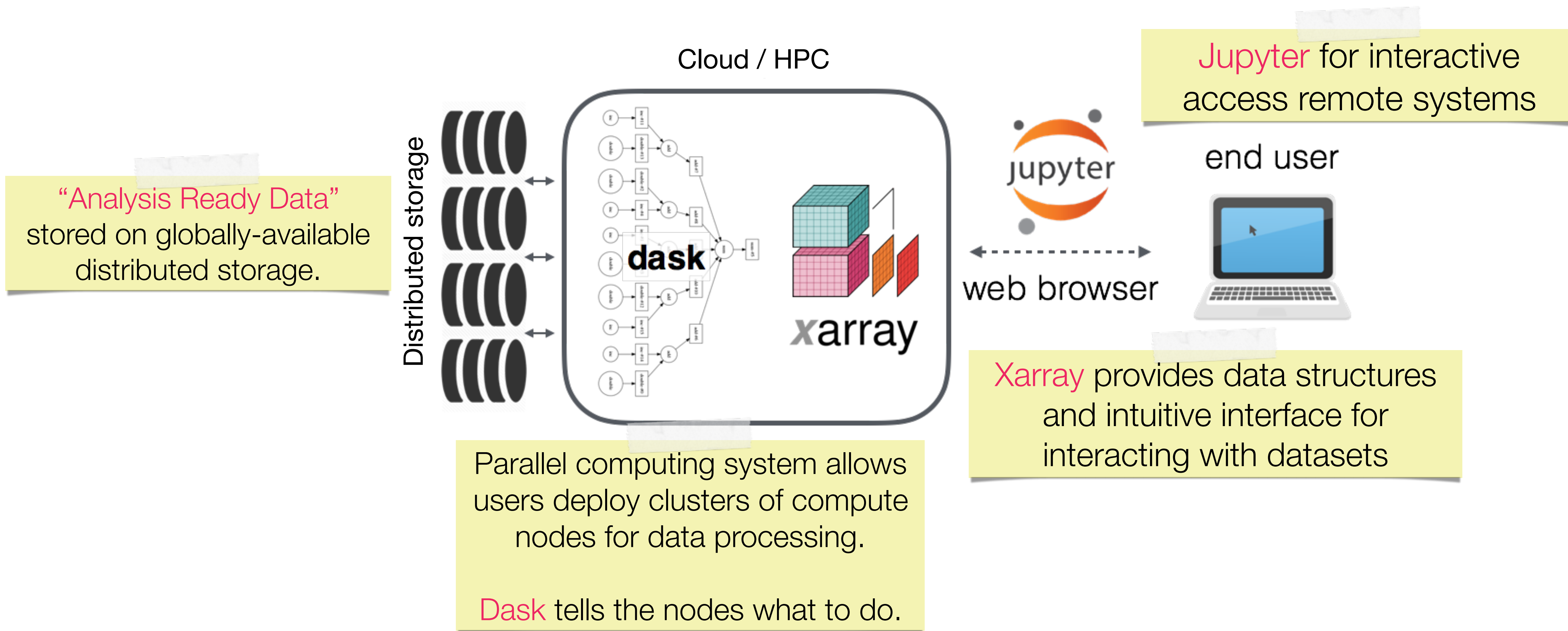
CLOUD-NATIVE APPROACH



Data provider's responsibilities

End user's responsibilities

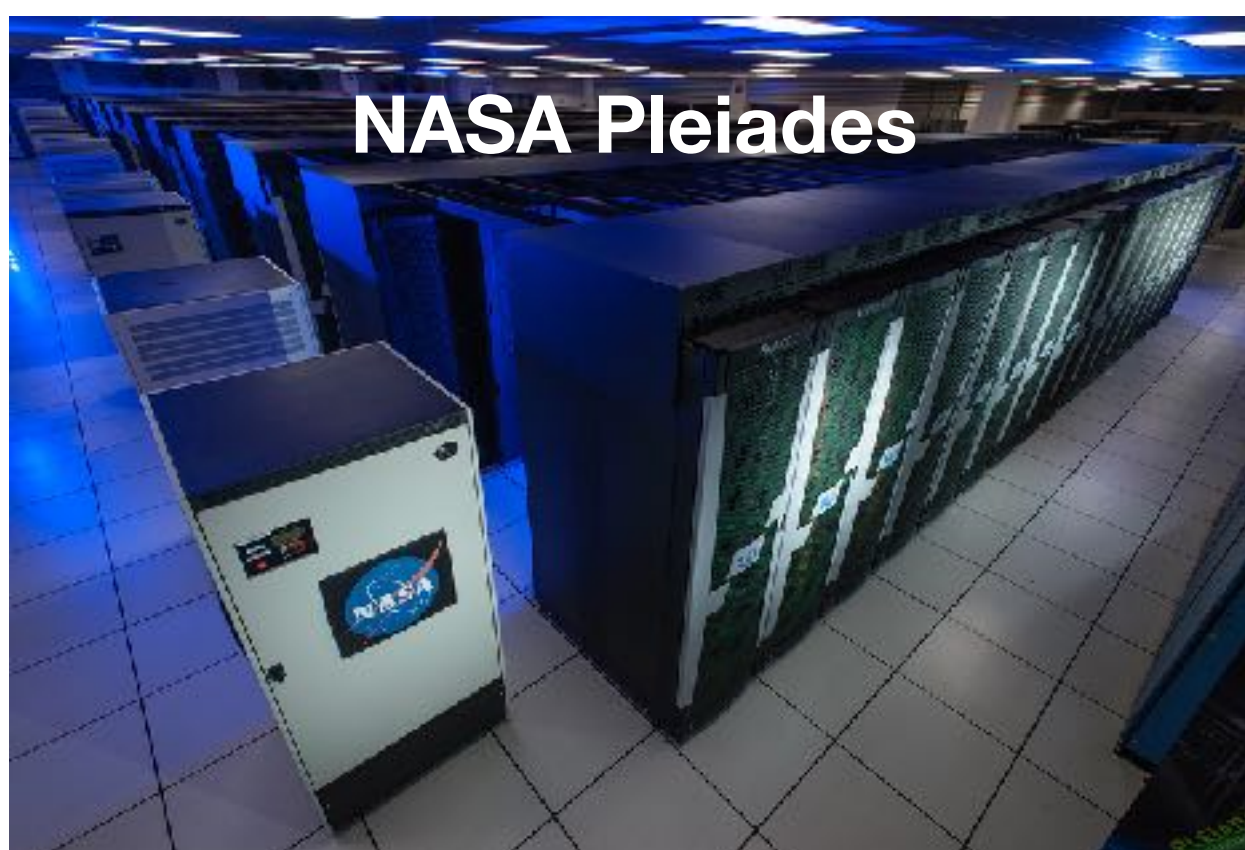
PANGEO ARCHITECTURE



BUILD YOUR OWN PANGEO

Storage Formats			Cloud Optimized COG/Zarr/Parquet/etc.
ND-Arrays			More coming...
Data Models			<p>pandas</p> $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$ 
Processing Mode	 <p>Interactive</p>	<p>Batch</p> 	<p>Serverless</p> 
Compute Platform	<p>HPC</p> 	<p>Cloud</p> 	<p>Local</p> 

PANGEO DEPLOYMENTS



NCAR Cheyenne



PANGEO.PYDATA.ORG



Over 1000 unique
users since March

Google Cloud Platform



[HTTP://PANGEO.IO/DEPLOYMENTS.HTML](http://PANGEO.IO/DEPLOYMENTS.HTML)



Government HPC



Commercial Cloud

	Government HPC	Commercial Cloud
Access	<ul style="list-style-type: none"> ✓ Available to all federally funded projects ✗ Available <i>only</i> to federally funded projects 	<ul style="list-style-type: none"> ✓ Available globally to anyone with a credit card ✗ Authentication is not integrated with existing research infrastructure
Cost	<ul style="list-style-type: none"> ✓ Cost is hidden from researchers and billed by funding agencies ✗ Allocations, quotas, limits 	<ul style="list-style-type: none"> ✗ Cost is borne by individual researchers and hidden from funding agencies ✓ Economics of scale, unlimited resources
Compute	<ul style="list-style-type: none"> ✓ Homogeneous, high performance nodes ✗ Queues, batch scheduling, ssh access ✗ Fixed-size compute 	<ul style="list-style-type: none"> ✓ Flexible hardware (big, small, GPU) ✓ Instant provisioning of unlimited resources ✓ Spot market: burstable, volatile
Storage	<ul style="list-style-type: none"> ✓ Fast parallel filesystems (e.g. GPFS) 	<ul style="list-style-type: none"> ✓ Fast object storage

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

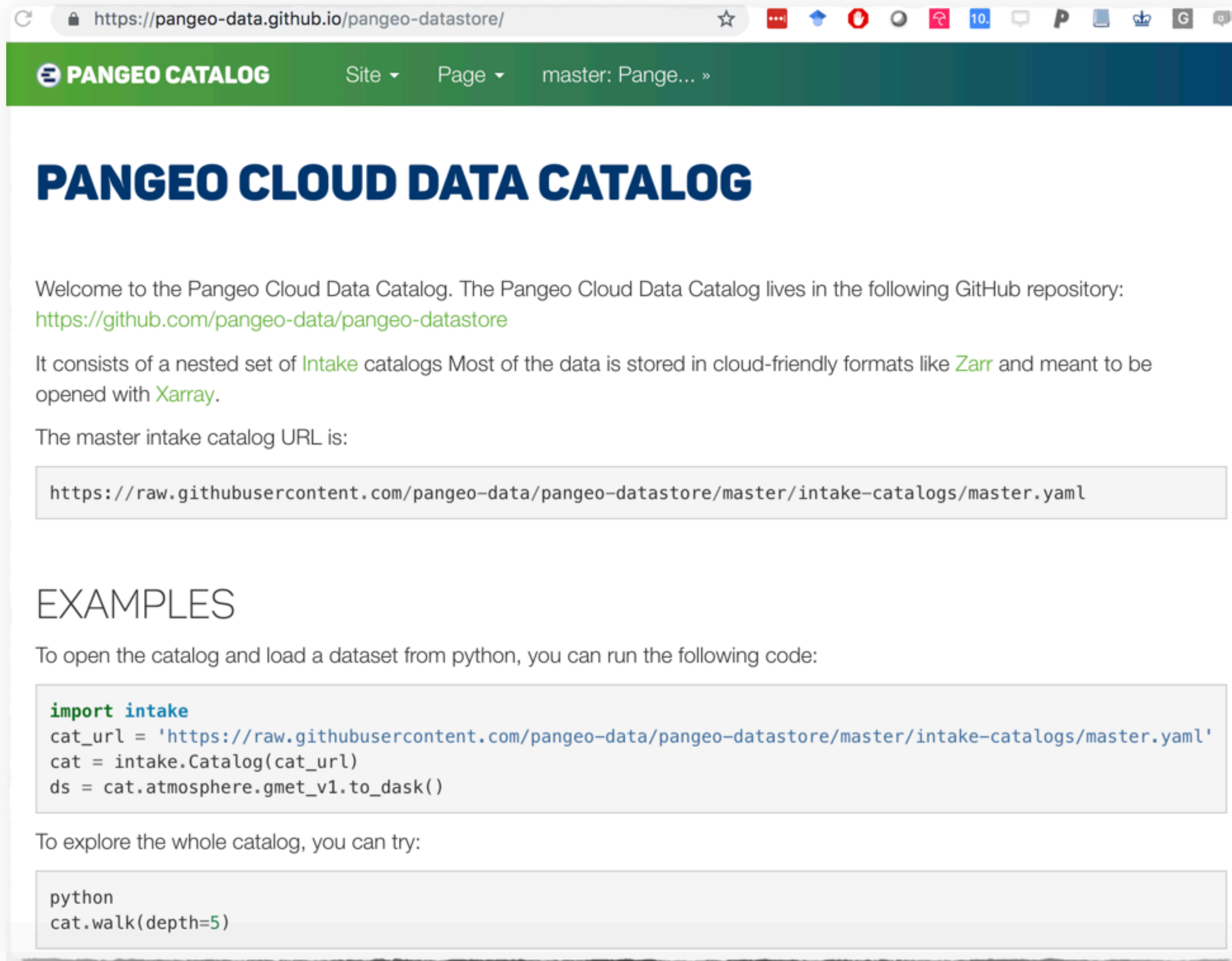
Name	Cloud: region	Staging URL	Production URL
dev	GCP: us-central1-b	https://staging.hub.pangeo.io	https://hub.pangeo.io
ocean	GCP: us-central1-b	https://staging.ocean.pangeo.io	https://ocean.pangeo.io
hydro	GCP: us-central1-b	https://staging.hydro.pangeo.io	https://hydro.pangeo.io
nasa	AWS: us-east-1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



The screenshot shows a web browser displaying the PANGEO CATALOG website. The browser's address bar shows the URL `https://pangeo-data.github.io/pangeo-datastore/`. The website has a green header with the PANGEO logo and navigation links for 'Site', 'Page', and 'master: Pange...'. The main content area features the title 'PANGEO CLOUD DATA CATALOG' and a welcome message. It provides the GitHub repository URL `https://github.com/pangeo-data/pangeo-datastore` and explains that the catalog uses `Intake` and stores data in `Zarr` format, which can be accessed with `Xarray`. A code block shows the master intake catalog URL: `https://raw.githubusercontent.com/pangeo-data/pangeo-datastore/master/intake-catalogs/master.yaml`. Below this, there is an 'EXAMPLES' section with instructions on how to open the catalog and load a dataset from Python, followed by a code block showing the necessary Python code.

`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:

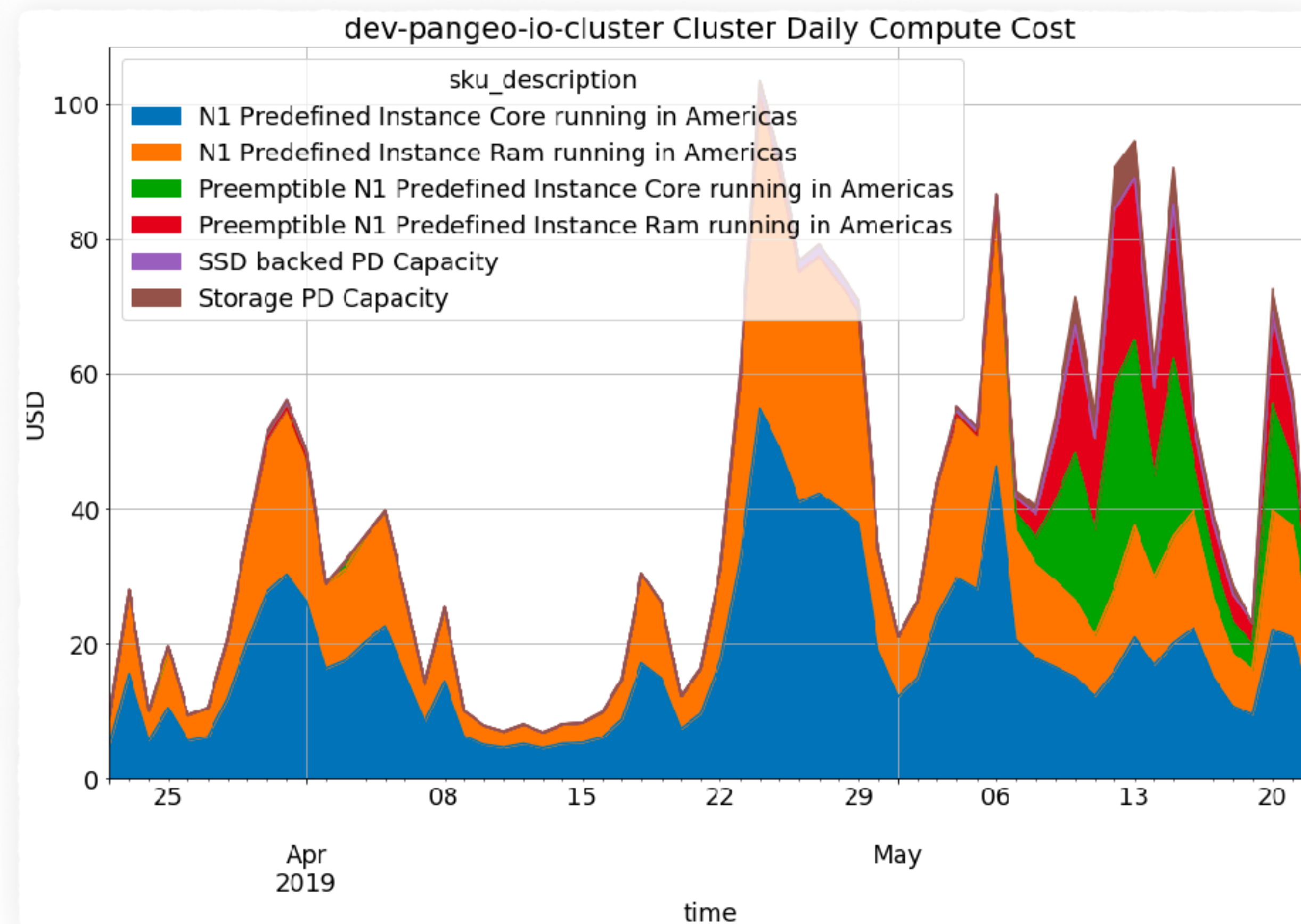
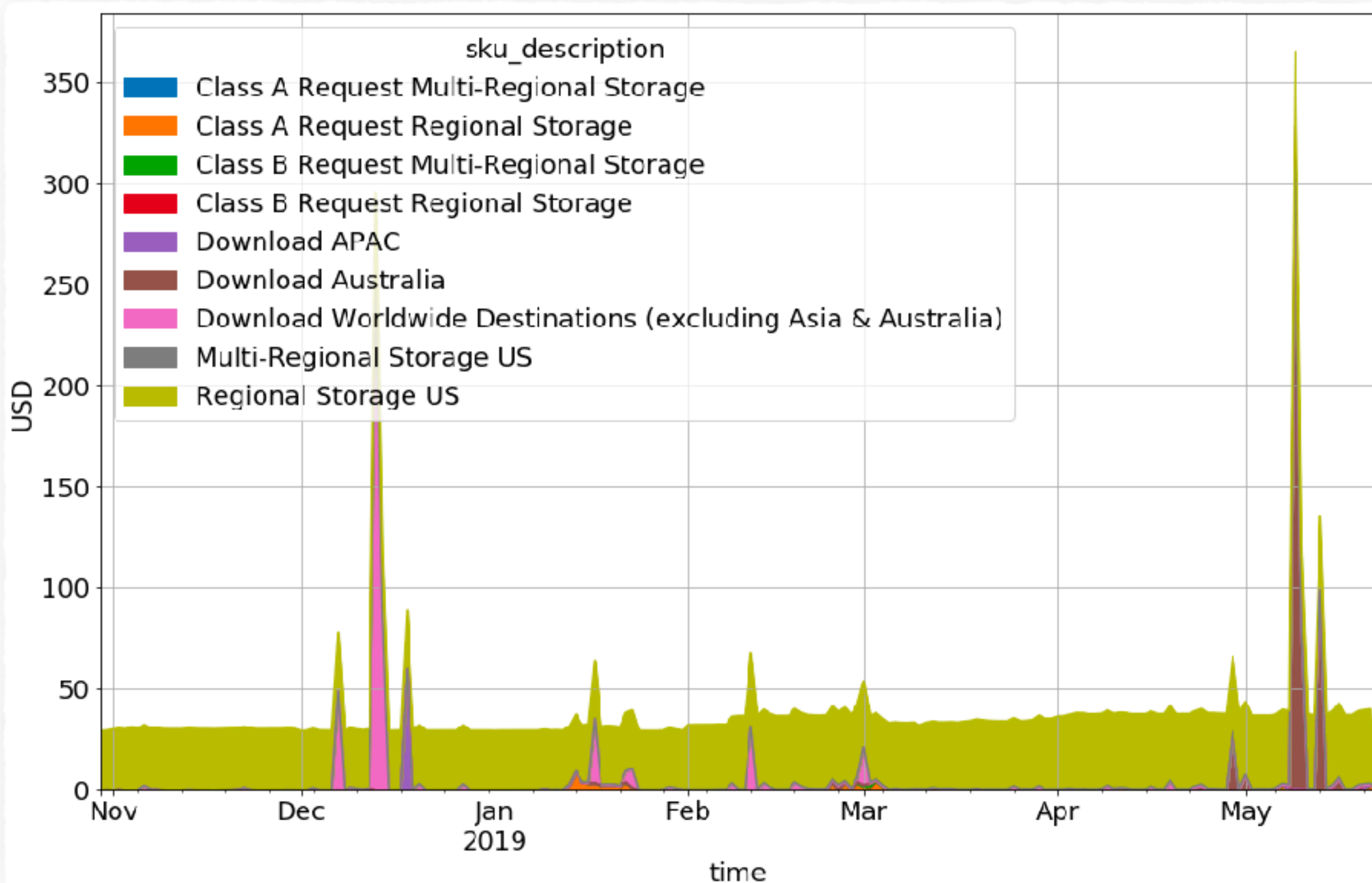
```
import intake
cat_url = 'https://raw.githubusercontent.com/pangeo-data/pangeo-datastore/master/intake-catalogs/master.yaml'
cat = intake.Catalog(cat_url)
ds = cat.atmosphere.gmet_v1.to_dask()
```

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

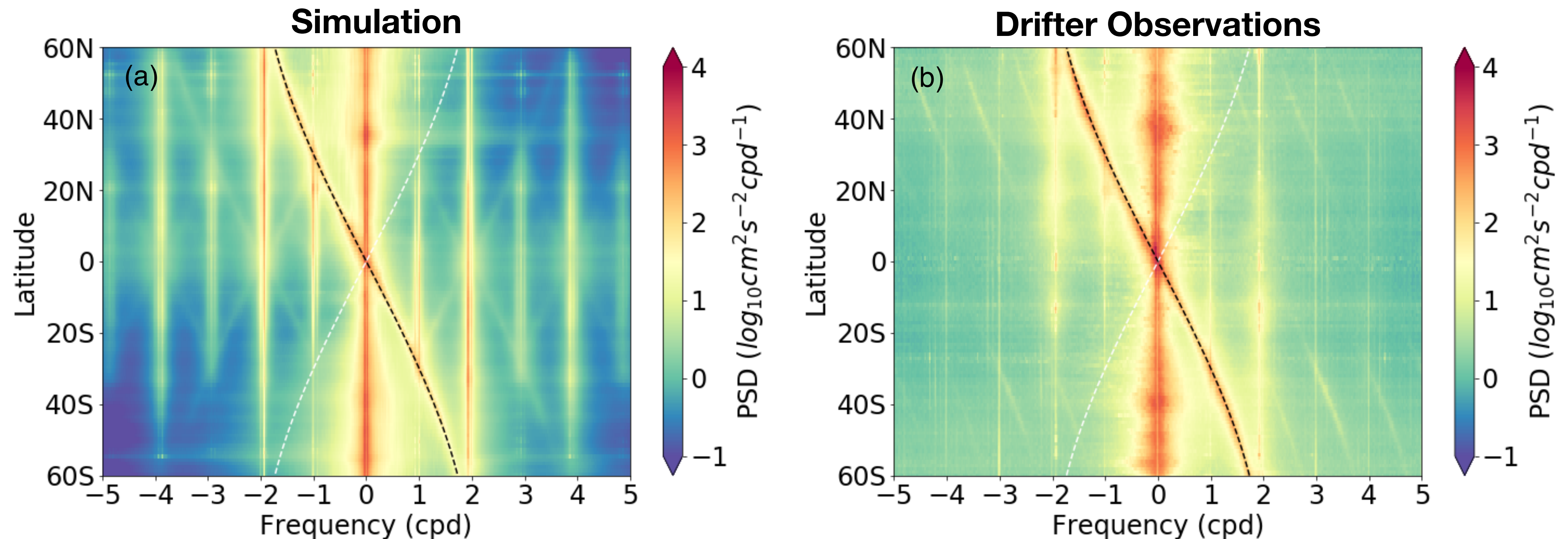


DEMO

<https://tinyurl.com/pangeo-ocean>

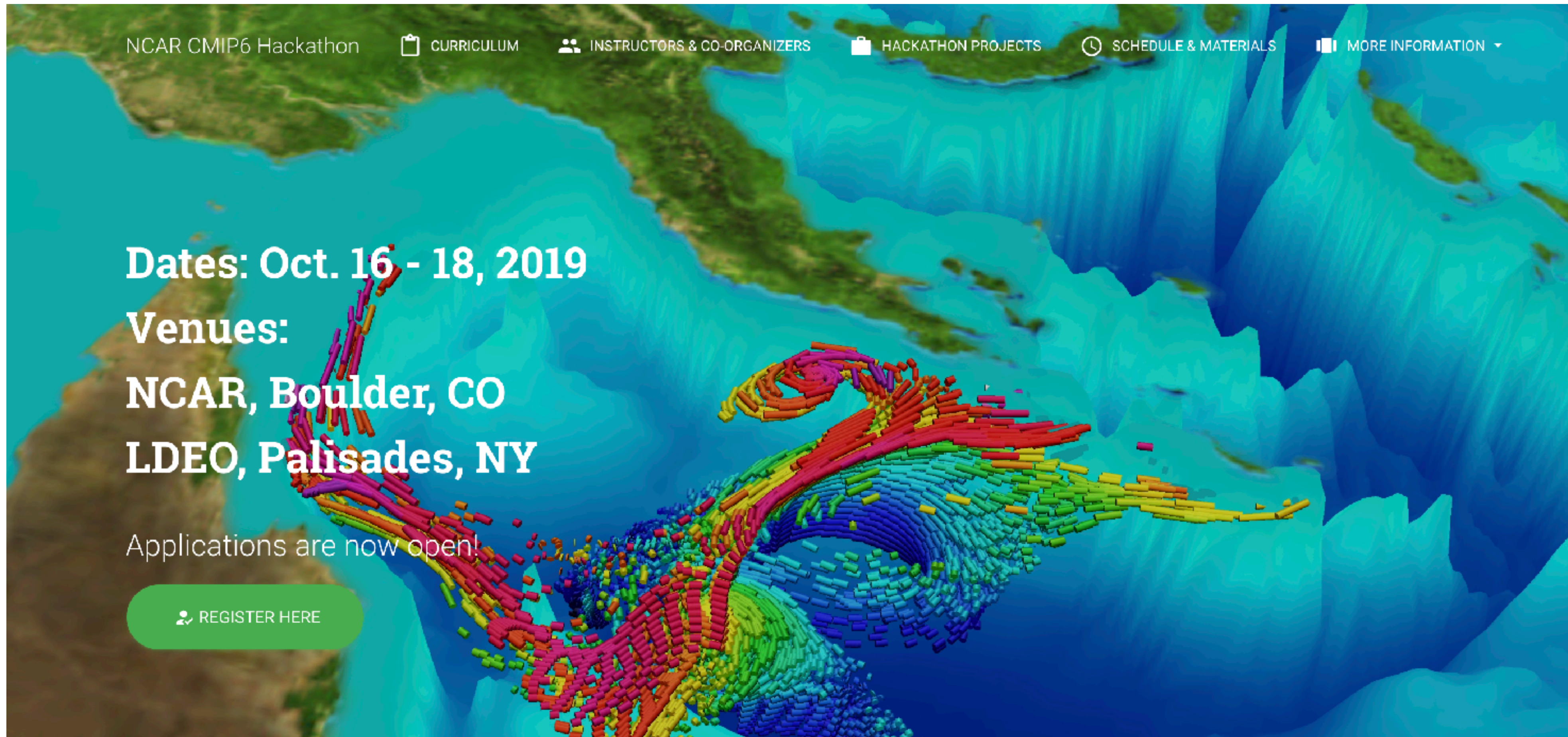
EXAMPLE CALCULATION

Used Pangeo to process 25TB of data on HPC Cluster



Yu et al. (2019) GRL

CMIP6 HACKATHON



NCAR CMIP6 Hackathon

CURRICULUM

INSTRUCTORS & CO-ORGANIZERS

HACKATHON PROJECTS

SCHEDULE & MATERIALS

MORE INFORMATION

Dates: Oct. 16 - 18, 2019

Venues:
NCAR, Boulder, CO
LDEO, Palisades, NY

Applications are now open!

REGISTER HERE

<https://cmip6hack.github.io>

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

cesm-pop-highres-ocean

Code Python 3

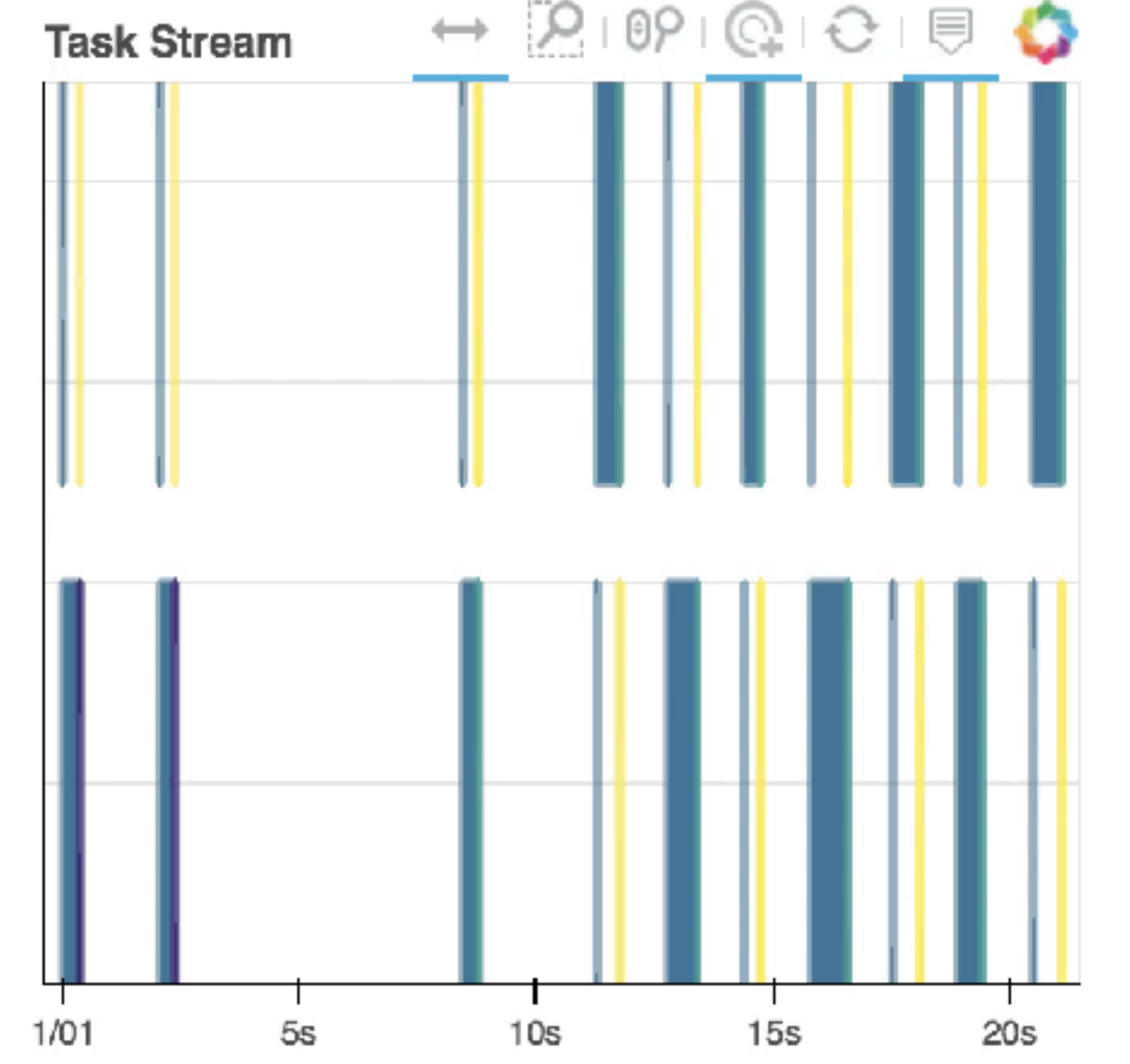
```
[5]: <xarray.DataArray 'SST' (time: 14965, nlat: 2400, nlon: 3600)>
dask.array<shape=(14965, 2400, 3600), dtype=float32, chunksize=(1, 2400, 3600)>
Coordinates:
  * time      (time) float64 1.679e+04 1.679e+04 ... 3.175e+04 3.176e+04
  * nlon      (nlon) int64 0 1 2 3 4 5 6 7 ... 3593 3594 3595 3596 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



Dask Progress

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