MC-Toolkit

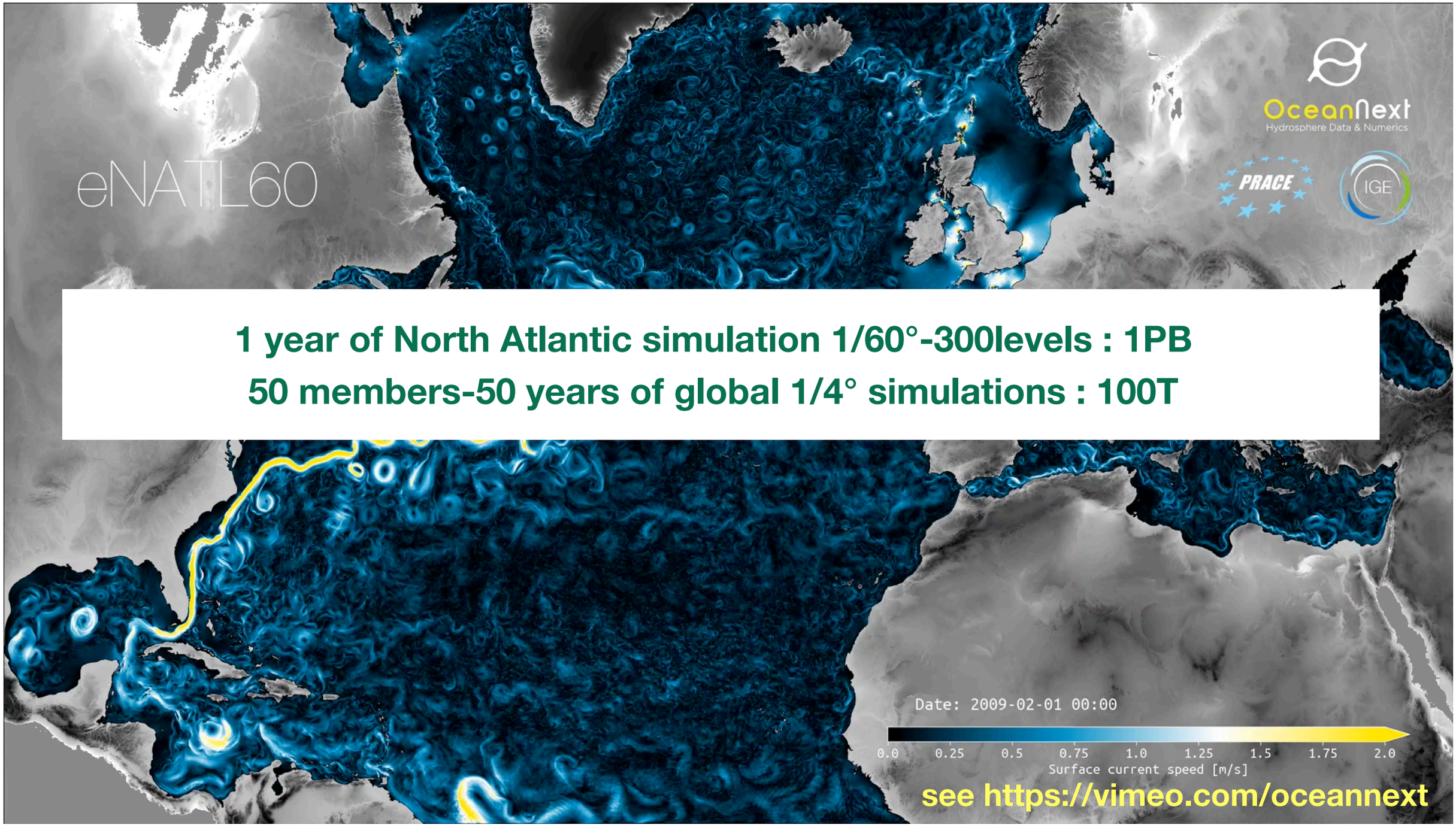
17 Nov. 2020



# Managing “big data“ in geosciences with PANGEO

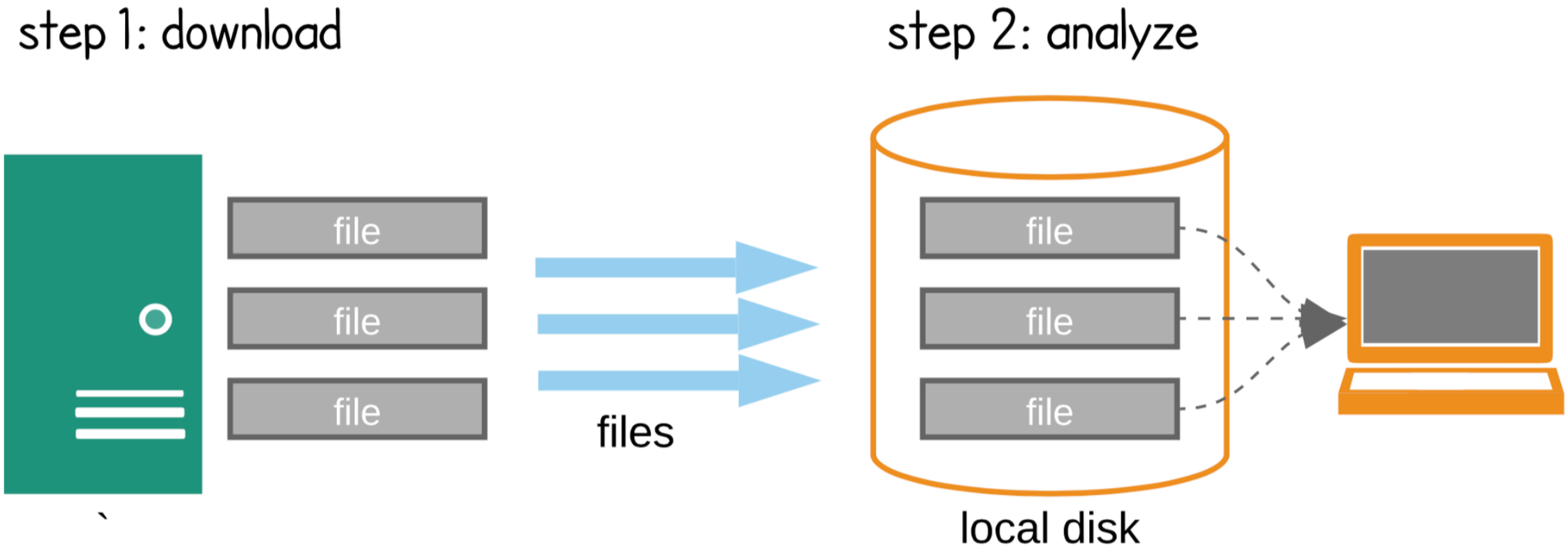
A. Albert, J. Le Sommer

# More and more data to analyse




**NEMO-eNATL60 simulations from IGE/Ocean Next : dx ~1km, 300 levels + tides**

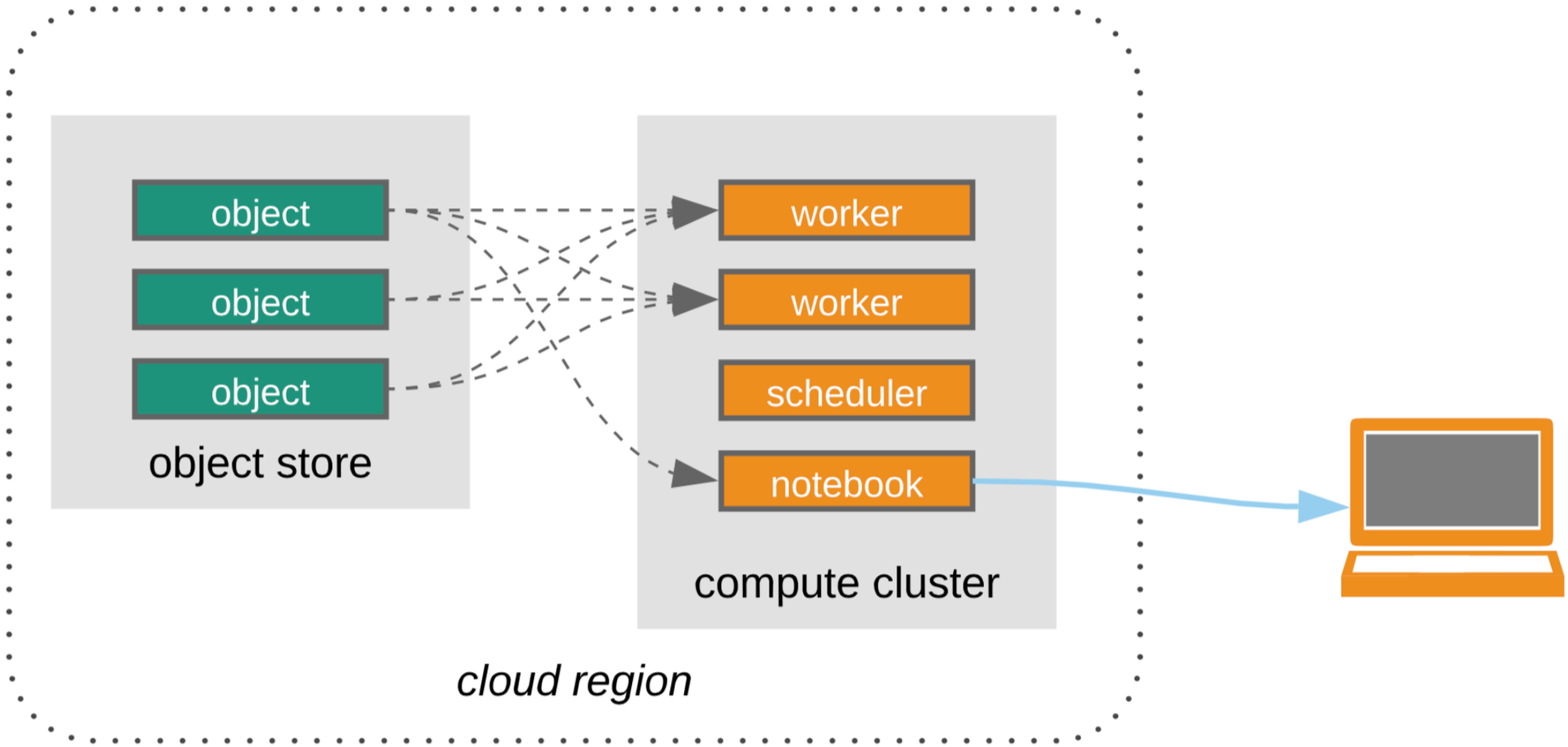
# Traditional workflow




**Data provider's responsibilities**

**End user's responsibilities**

# A new workflow ?




**Data provider's responsibilities**

**End user's responsibilities**

# What is PANGEO?

pangeo.io

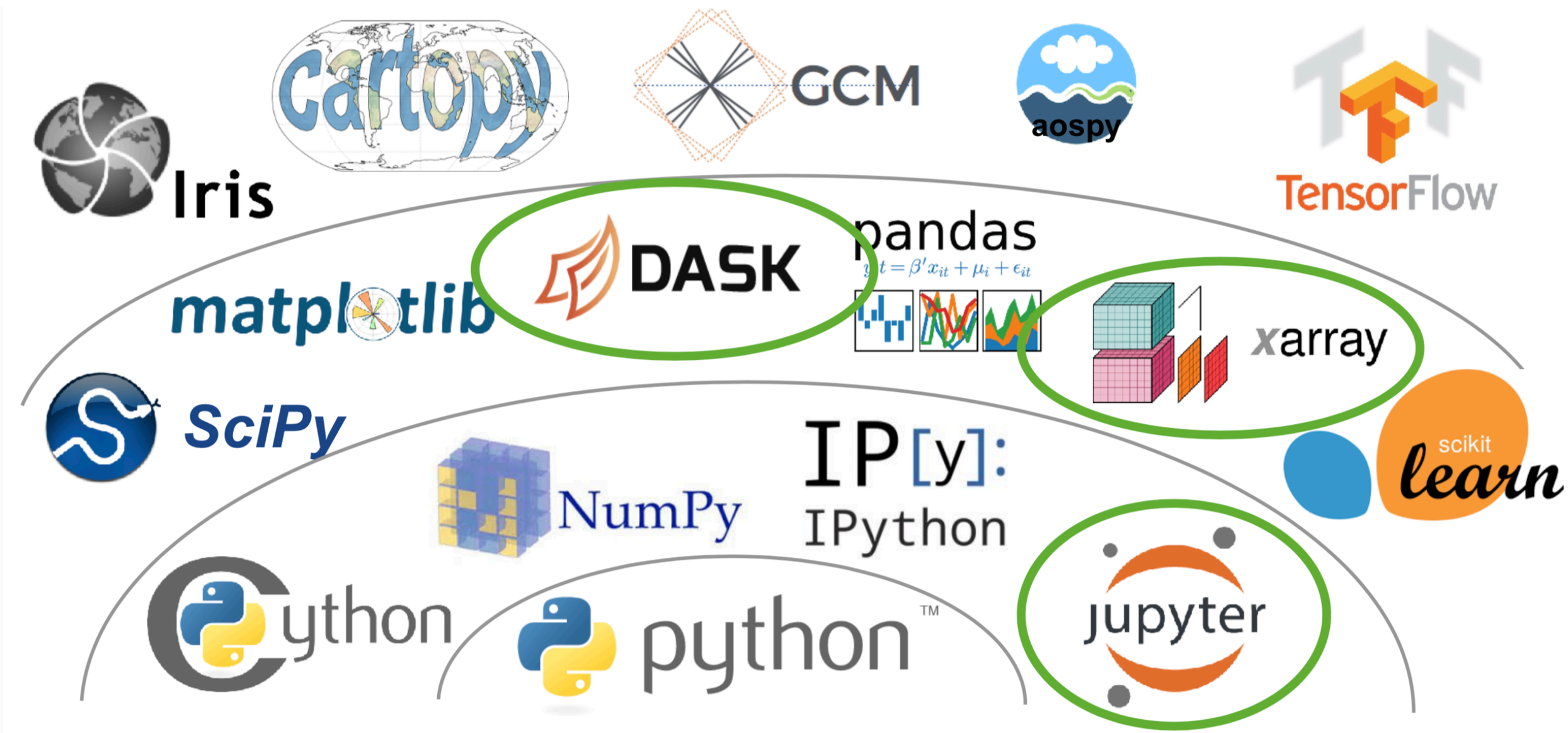
## A community platform for Big Data geoscience

- ✦ “big data geosciences” software ecosystem
- ✦ international community of scientists, developers
- ✦ a cloud deployment

# PANGEO : the community

- bringing together researchers, engineers and developers
    - members in big institutions (ECMWF, UKMO, CNES, …)
    - inclusive and open-source approach (github, blog, tutorial, etc…)
    - annual US and Europe workshops (~30 p. in Paris, May 2019 )

# PANGEO : software ecosystem




**Credit: Stephan Hoyer, Jake Vanderplas (SciPy 2015)**

# PANGEO : software ecosystem


- interactive “web-based” framework for computation
- a reproducible environment for :
  - code
  - text
  - equations (LaTeX)
  - visualisations

**...back to oxygen**

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

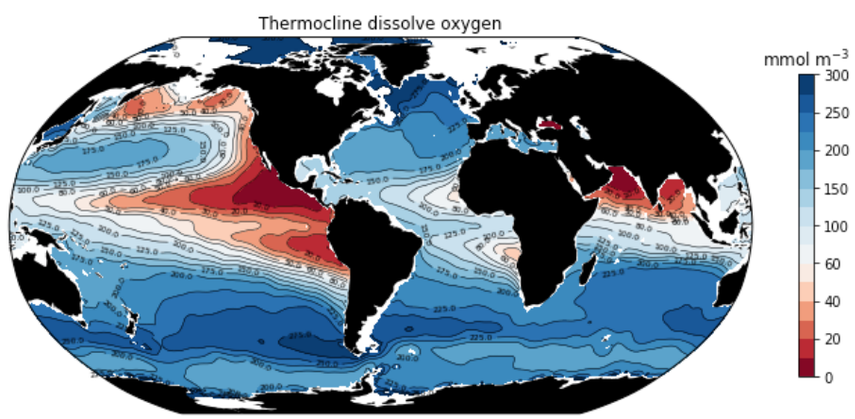
In [7]:
```
levels = [0, 10, 20,  30, 40, 50, 60, 80, 100, 125, 150, 175, 200, 225,
          250, 275, 300]

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

Add a cyclic point to accomodate the periodic domain.

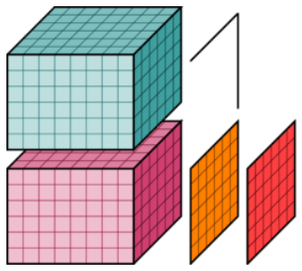
In [8]:
```
from cartopy.util import add_cyclic_point
field, lon = add_cyclic_point(ds.O2, coord=ds.lon)
lat = ds.lat
```

**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');
```

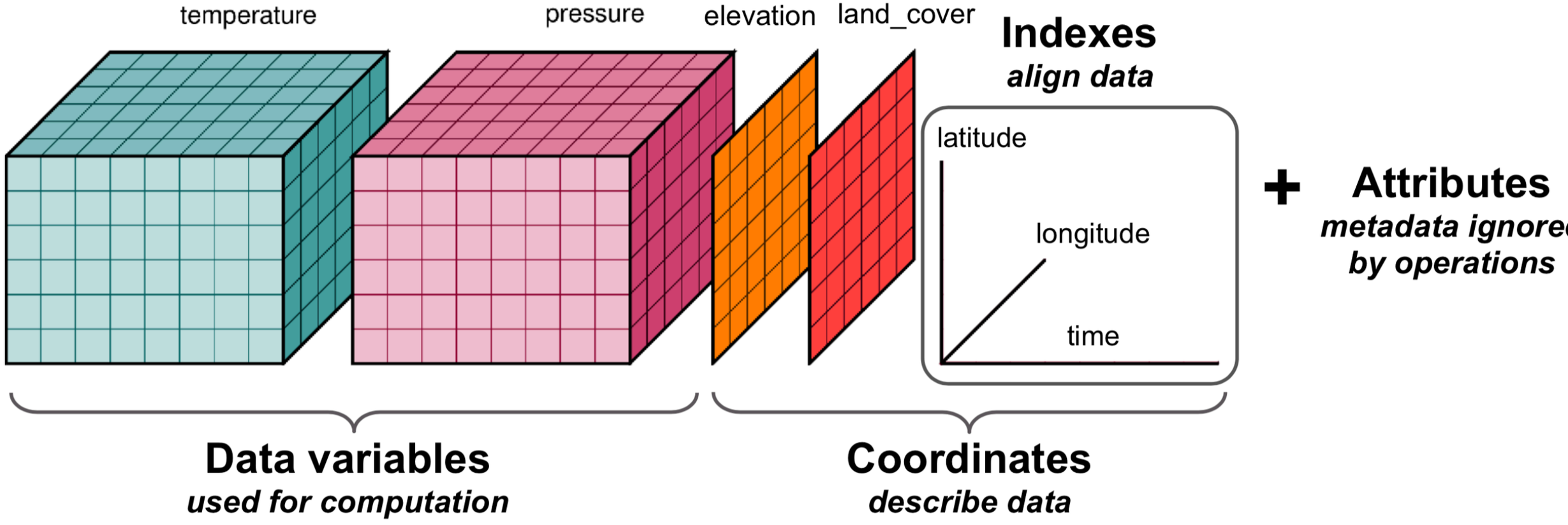


# PANGEO : software ecosystem



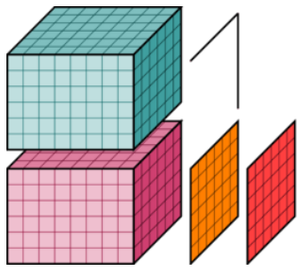
xarray

- data structure N-D matrix
- API derived from pandas
- the base for a lot of other tools




*"netCDF meets pandas.DataFrame"*

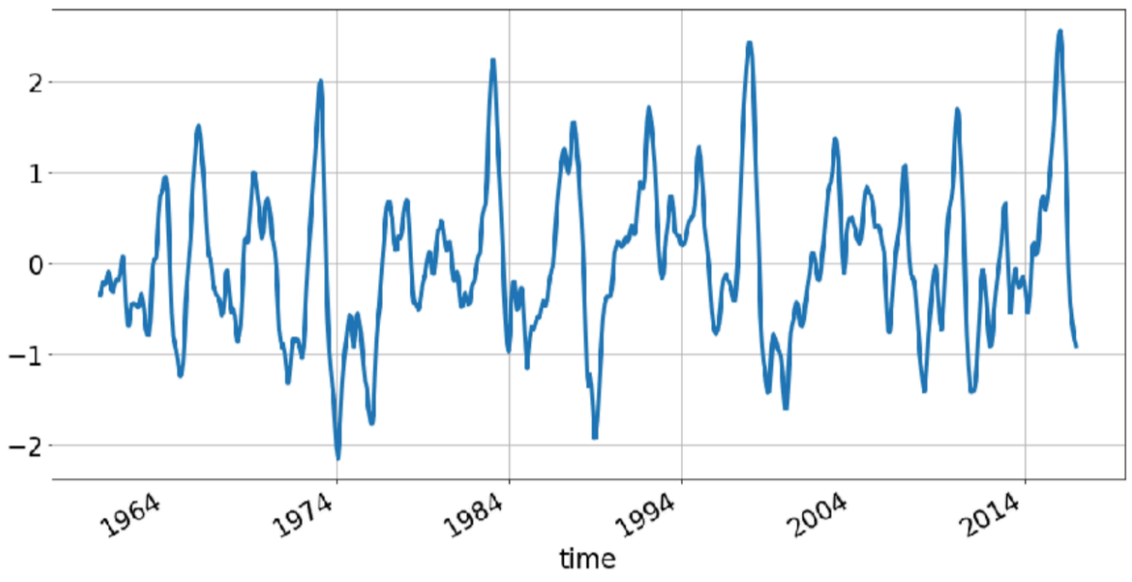
# PANGEO : software ecosystem

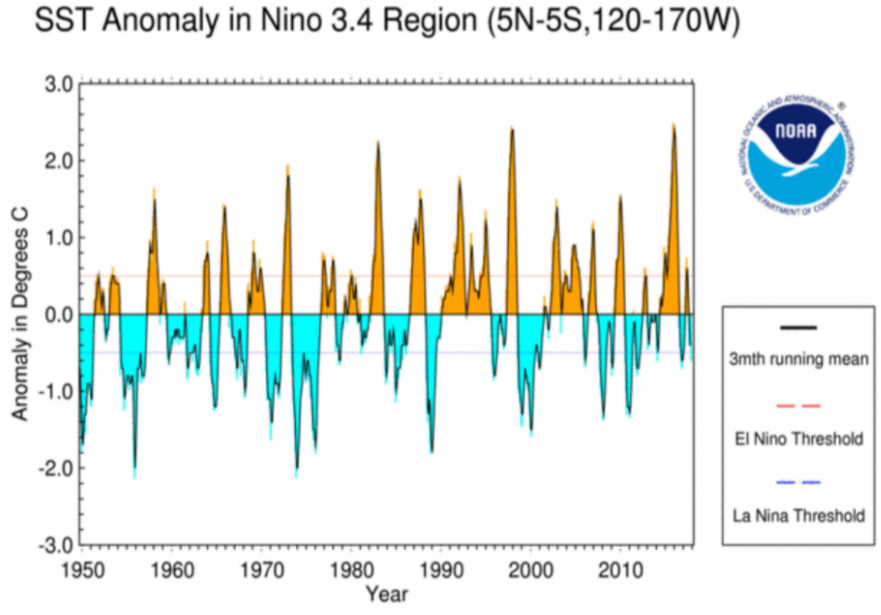


xarray

- data structure N-D matrix
- API derived from pandas
- the base for a lot of other tools

```
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()
```

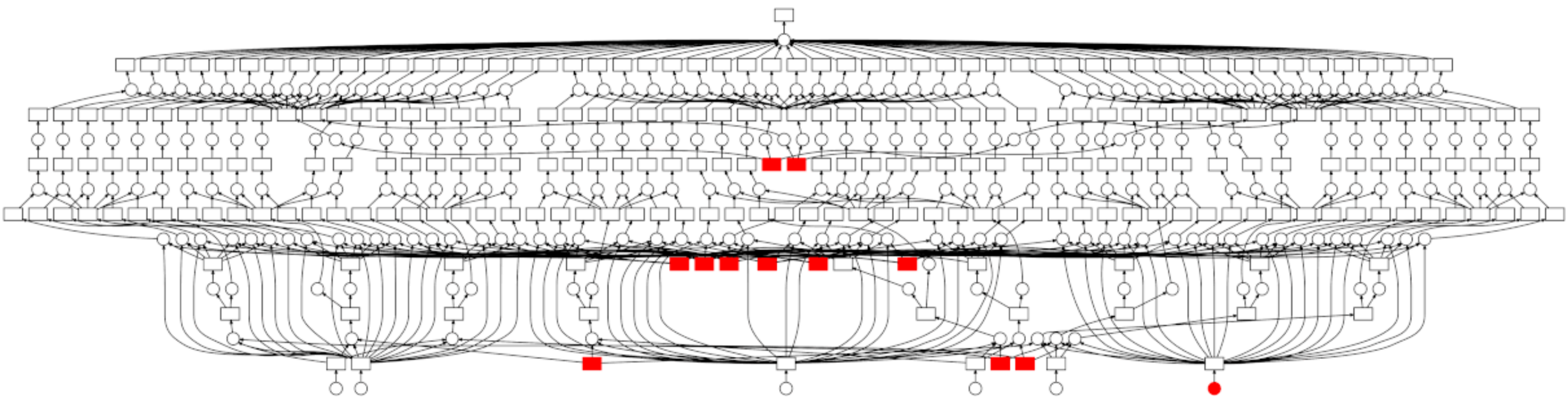


# PANGEO : software ecosystem

DASK

- parallel computing library for python
- allows to scale Numpy, Pandas, Scikit-Learn, Xarray...
- can be deployed on HPC systems or on the cloud
- with a task graph and dashboard to manage the execution



# PANGEO : software ecosystem

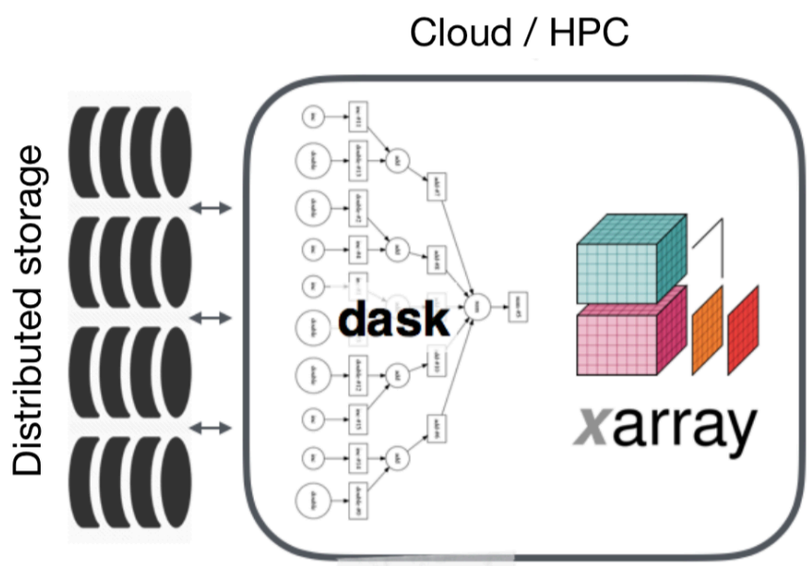
A library ecosystem based on these softwares :

- xgcm/xrft : Fourier transform for xarray data
- CNES/pangeo-pyinterp : interpolation of geo-referenced data
- willyrath/xorca : xarray with NEMO ocean model outputs
- serazing/xscale : statistics and signal computing
- the complete list : http://xarray.pydata.org/en/stable/related-projects.html

Open source and robust tools that allow us to scale up our analysis

# PANGEO : a deployment

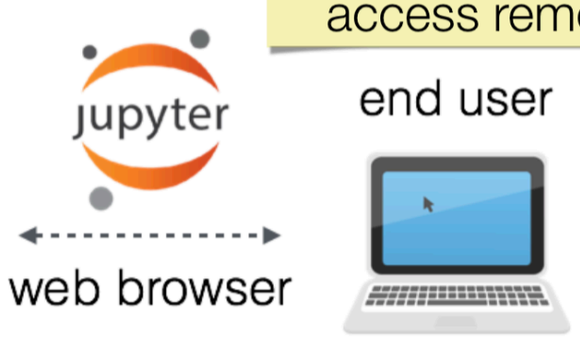
"Analysis Ready Data"
stored on globally-available
distributed storage.




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

Dask tells the nodes what to do.

Jupyter for interactive access remote systems




Xarray provides data structures and intuitive interface for interacting with datasets

- a cloud deployment (Jupyterhub ) pangeo.pydata.org
- other deployments on cluster/HPC : https://pangeo.io/deployments.html
- NCAR HPC and NASA, CNES cluster HAL.
- no jupyterhub on Occigen (personnal deployment), jupyterhub on Jean-Zay

# Two more tools for our analysis

**DASK-JOBQUEUE**

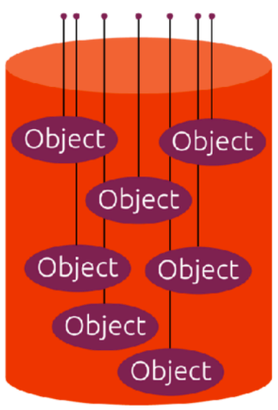
- dask on job scheduler (PBS, Slurm, etc…)
- a french spin-off project in PANGEO
- a pythonic interface to work with dask workers/clusters

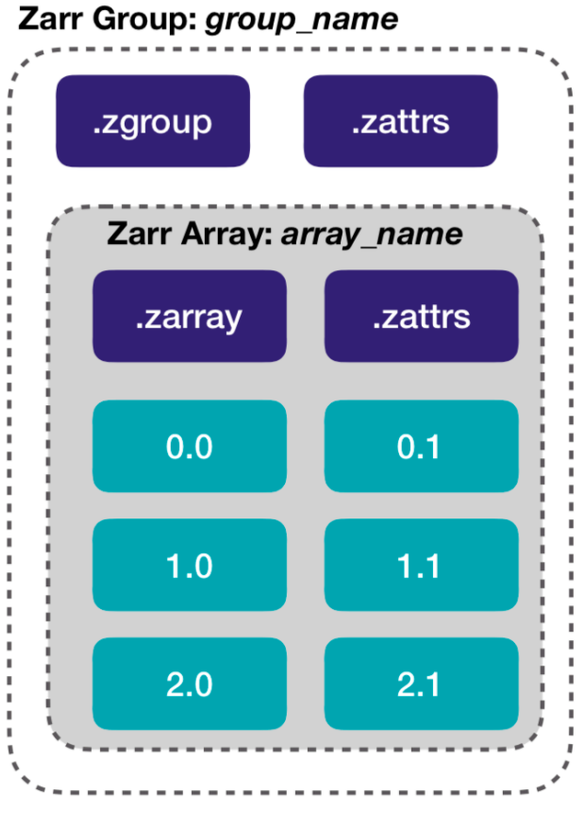
```python
from dask_jobqueue import PBSCluster
from distributed import Client
cluster = PBSCluster(project=..,
  queue=.., cores=1, processes=1,
  memory="100GB", walltime=...)
# Ask for 10 nodes
cluster.scale(10)
# OR scale adaptively based on load
cluster.adapt(minimum=1, maximum=100,
            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

```python
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)
```

# Two more tools for our analysis

## ZARR OBJECT STORAGE




- Open source library for storage of chunked, compressed ND-arrays
- Created by Alistair Miles (Imperial) for genomics research (*@alimanfoo*); now community supported standard
- Arrays are split into user-defined chunks; each chunk is optional compressed (zlib, zstd, etc.)
- Can store arrays in memory, directories, zip files, or any python mutable mapping interface (dictionary)
- External libraries (s3fs, gcsf) provide a way to store directly into cloud object storage
- Implementations in Python, C++, Java (N5), Julia

**Required for cloud computing and better performance on HPC**

# Exemple

```
files=sorted(glob.glob('/work/ALT/odatis/eNATL60/BLBT02/gridT-2D/eNATL60*gridT-2D*.nc'))
drop_vars = ['nav_lat', 'nav_lon', 'somxl010', 'sosaline', 'sosstsst']
extra_coord_vars = []
chunks = dict(time_counter=240, y=240, x=480)
open_kwargs = dict(drop_variables=(drop_vars + extra_coord_vars),
                   chunks=chunks,decode_cf=True,decode_times=False,
                   concat_dim="time_counter",combine='nested')
```

```
%time ds = xr.open_mfdataset(files, parallel=True, **open_kwargs)
```

```
CPU times: user 3.75 s, sys: 197 ms, total: 3.95 s
Wall time: 4.77 s
```

```
%time ds=xr.open_zarr('/work/ALT/odatis/eNATL60/zarr/eNATL60-BLBT02-SSH-1h')
```

```
CPU times: user 100 ms, sys: 29.2 ms, total: 130 ms
Wall time: 149 ms
```

```
from dask.distributed import Client
from dask_gateway import Gateway

gateway = Gateway()
cluster = gateway.new_cluster()
cluster.scale(nb_workers)
c = Client(cluster)
c
```

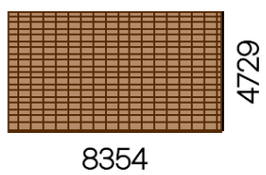
```
%time mean=ds.sossheig.mean(dim='time_counter')
```

```
CPU times: user 116 ms, sys: 7.68 ms, total: 123 ms
Wall time: 120 ms
```

```
mean
```

xarray.DataArray 'sossheig' (y: 4729, x: 8354)

| | Array | Chunk |
|---|---|---|
| Bytes | 158.02 MB | 460.80 kB |
| Shape | (4729, 8354) | (240, 480) |
| Count | 41761 Tasks | 360 Chunks |
| Type | float32 | numpy.ndarray |




▸ Coordinates: (0)

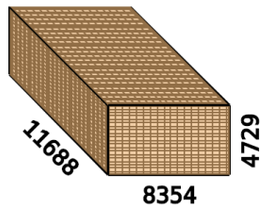
▸ Attributes: (0)

```
%time mean.load()
```

```
CPU times: user 5.38 s, sys: 561 ms, total: 5.95 s
Wall time: 4min 17s
```

| | | | |
|---|---|---|---|
| sossheig | (time_counter, y, x) | float32 | dask.array<chunksize=(24... |

| | Array | Chunk |
|---|---|---|
| Bytes | 1.85 TB | 110.59 MB |
| Shape | (11688, 4729, 8354) | (240, 240, 480) |
| Count | 17641 Tasks | 17640 Chunks |
| Type | float32 | numpy.ndarray |

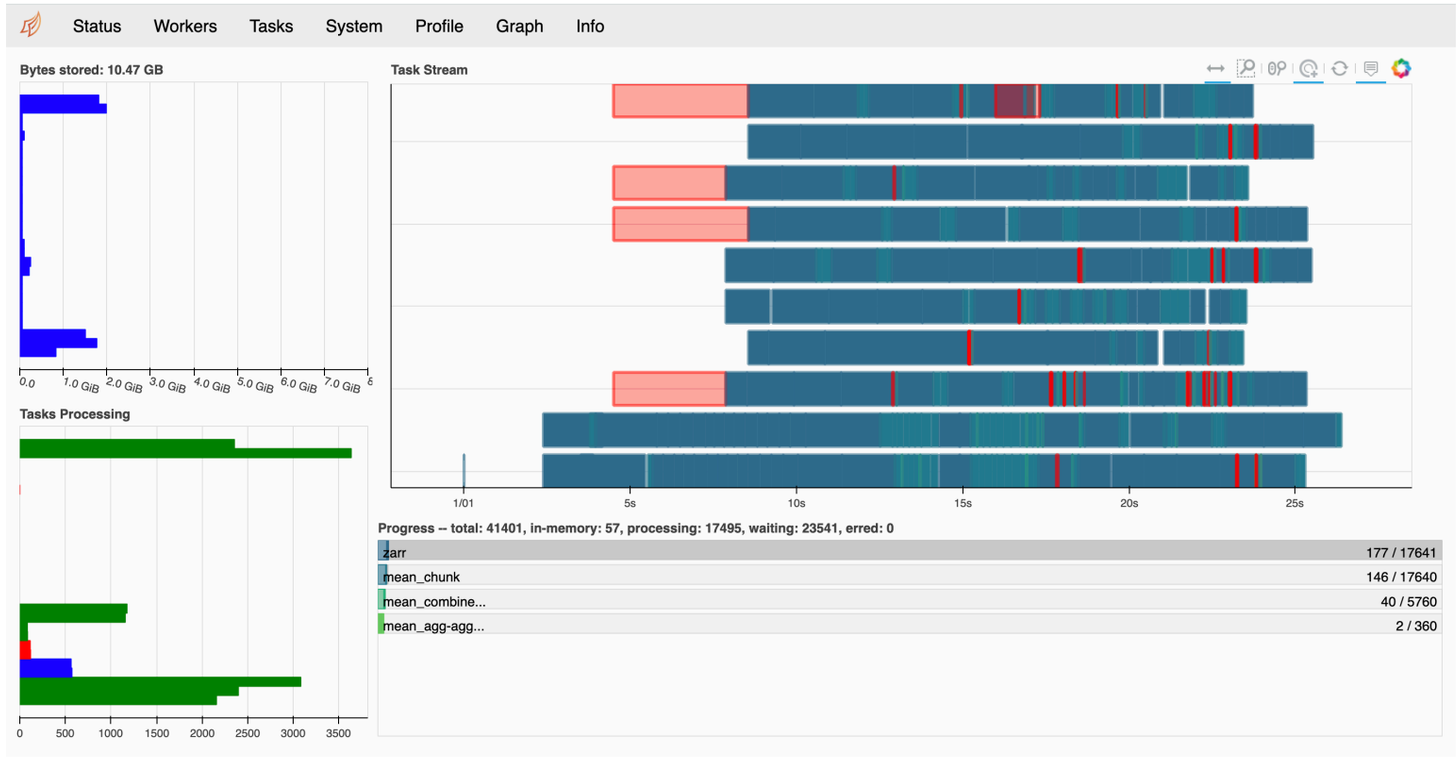



## Client

**Scheduler:** gateway://traefik-gcp-uscentral1b-prod-dask-gateway.prod:80/prod.3a7a52cc26364e0198c7e52896bc0d60
**Dashboard:** /services/dask-gateway/clusters/prod.3a7a52cc26364e0198c7e52896bc0d60/status

## Cluster

**Workers:** 5
**Cores:** 10
**Memory:** 42.95 GB



# Demonstration on binder

**on github : https://tinyurl.com/MCToolkit**