



Pangeo

A community platform
for Big Data geoscience



Sommaire

Pangeo

- Qu'est-ce?
- Architecture

Pile logicielle

- Jupyter
- Xarray
- Dask

Atelier

Pangeo – Qu'est-ce?



PANGEO

A community platform
for Big Data geoscience

Un écosystème logiciels “big data geosciences”

Une communauté internationale de développeurs

Une infrastructure partagée sur le cloud



Mission : entretenir un écosystème dans lequel la prochaine génération d'outils, d'analyse « open source » pour les géosciences appliquée à de gros volumes de données peut être développée, distribuée et maintenue.

Vision:

- Développement ouvert et collaboratif.
- Outils de mise à l'échelle des calculs pour des ensembles de données de petite à très grande taille.
- Structures pour intégrer l'analyse scientifique aux données.
- Encourager une culture du développement accueillante et participative.



PANGEO

Présentation: [ESIP Tech Dive](#) (01/2018) par Ryan Abernathy & Matthew Rocklin

Site web: <http://pangeo.io/>

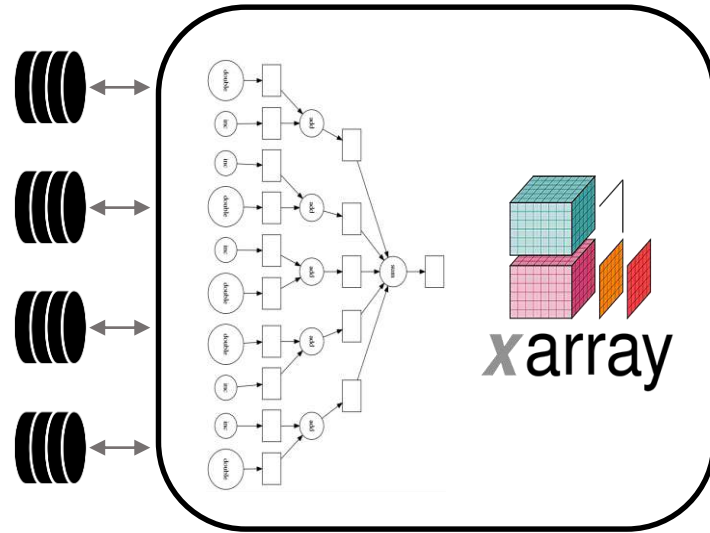
Github: <https://github.com/pangeo-data>

Blog: <https://medium.com/pangeo>

Gitter: <https://gitter.im/pangeo-data>

Pangeo - Architecture

Données, prêtes à l'analyse, stockées et cataloguées sur un système de stockage distribué accessible à l'échelle globale (p. ex. S3, GCS)

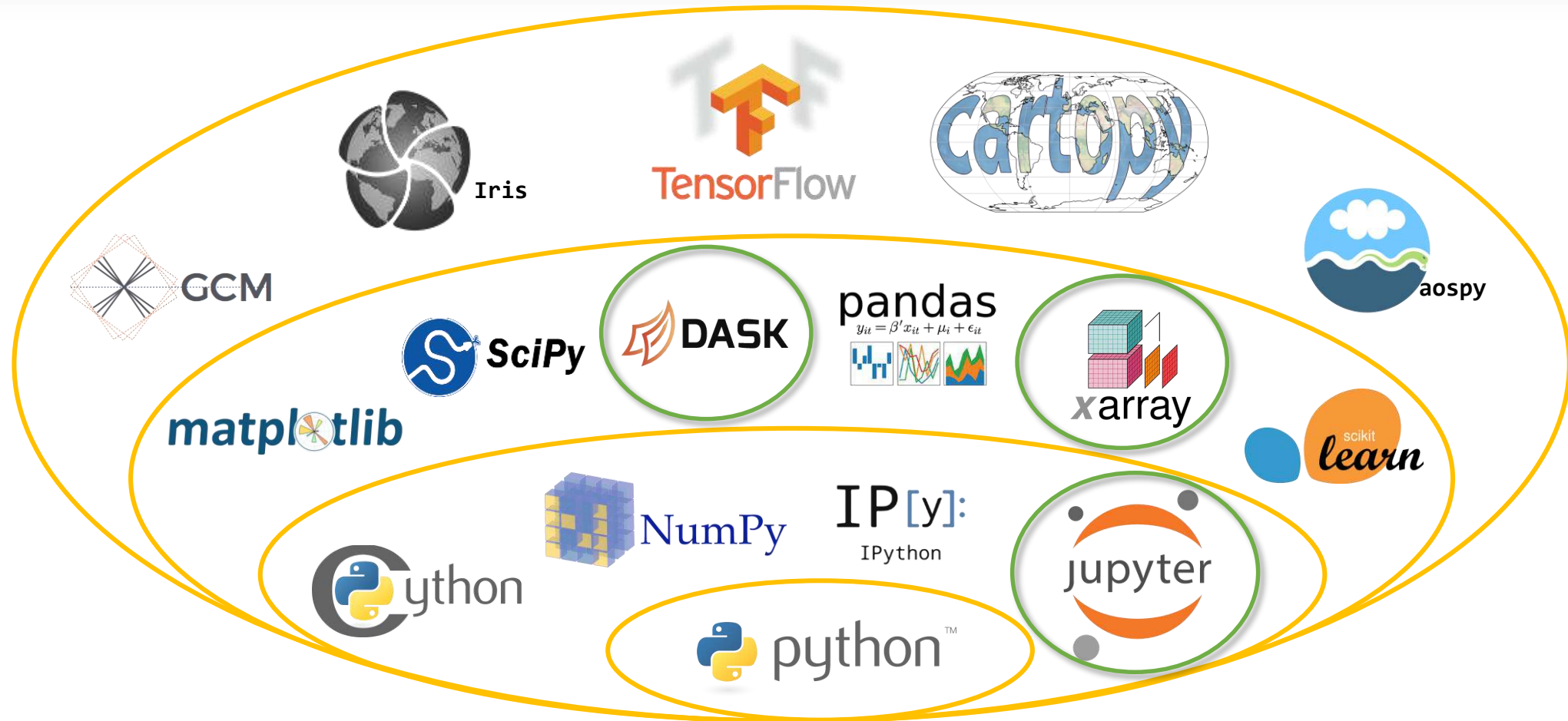


Système de calcul parallèle construit sur Kubernetes ou HPC.
Dask dit aux nœuds ce qu'ils doivent faire.

Jupyter pour un accès interactif sur des systèmes distants.

xarray fournit des structures de données et une interface intuitive pour interagir avec les ensembles de données.

Pangeo – Pile logicielle





Un environnement de calcul interactif

Un format de document reproductible :

- Code
- Texte
- 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))

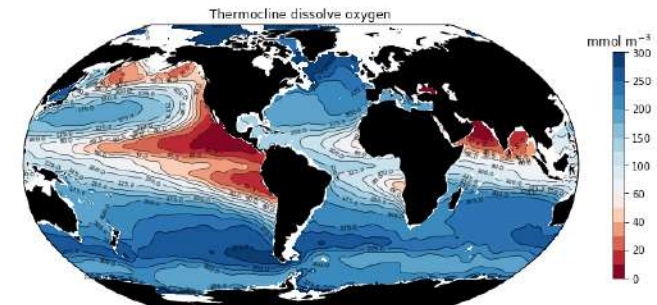
# filled contours
cf = ax.contourf(lon, lat, field, levels=levels, norm=norm, cmap='RdBu',
                transform=ccrs.PlateCarree());

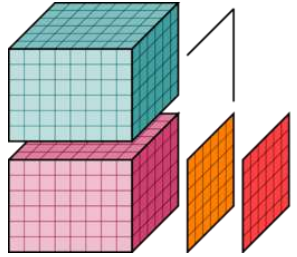
# contour lines
cs = ax.contour(lon, lat, field, colors='k', levels=levels, linewidths=0.5,
               transform=ccrs.PlateCarree());

# 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');
```



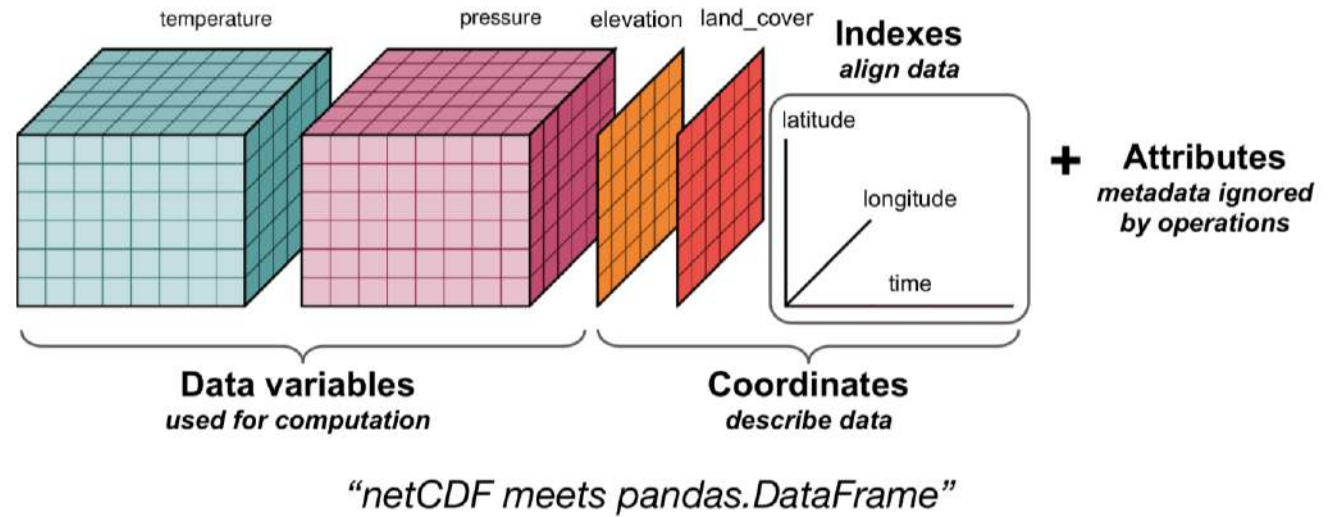


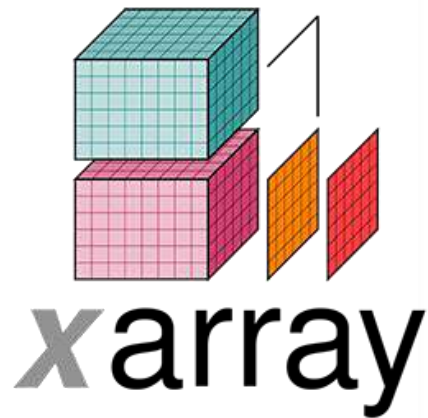
xarray

Structures de données pour tableaux N-D

Une API inspirée de pandas

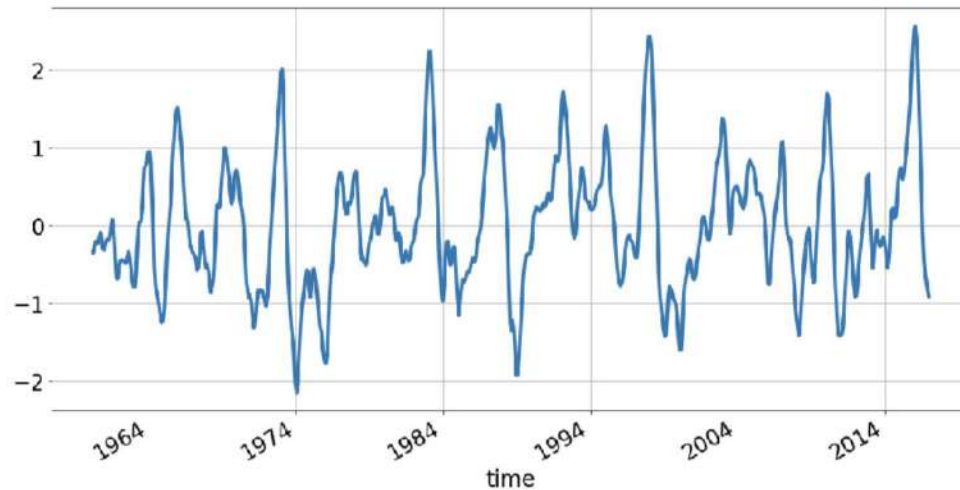
Interopérable avec les outils standards pydata



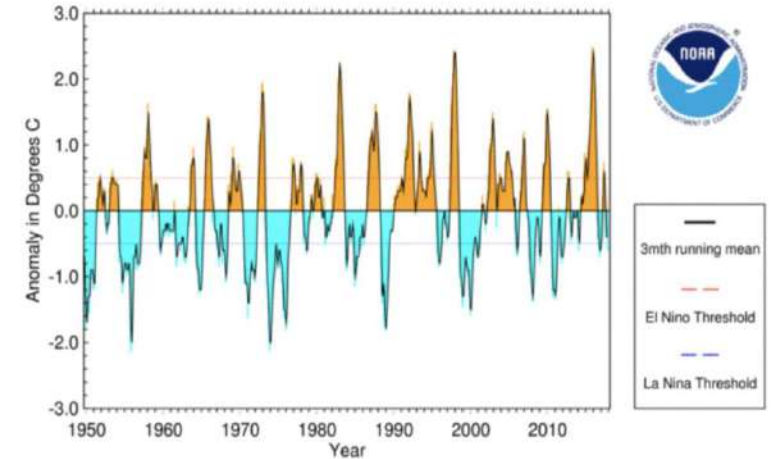


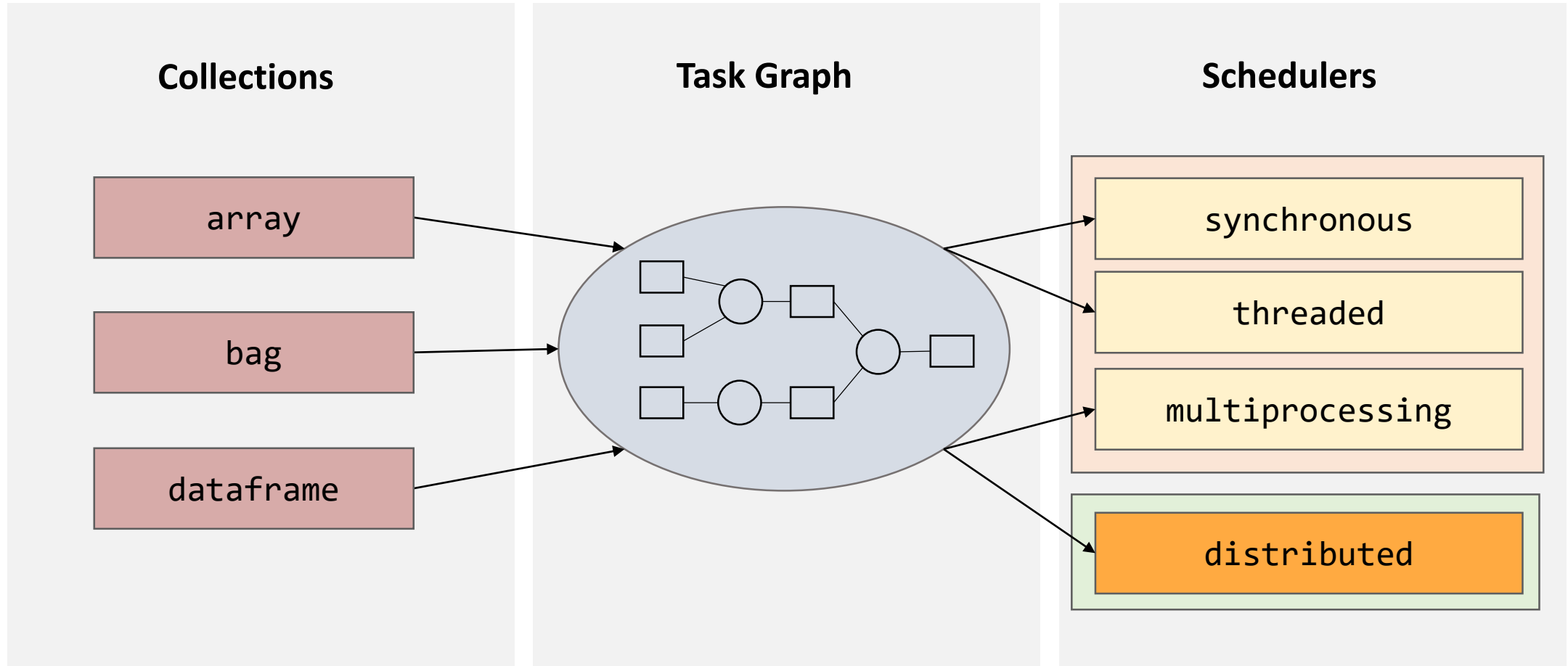
```
sst_clim = sst.groupby('time.month').mean(dim='time')
sst_anom = sst.groupby('time.month') - sst_clim
nino34_index = (sst_anom.sel(lat=slice(-5, 5), lon=slice(190, 240))
                .mean(dim=('lon', 'lat'))
                .rolling(time=3).mean(dim='time'))

nino34_index.plot()
```

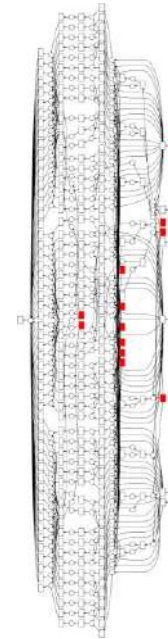
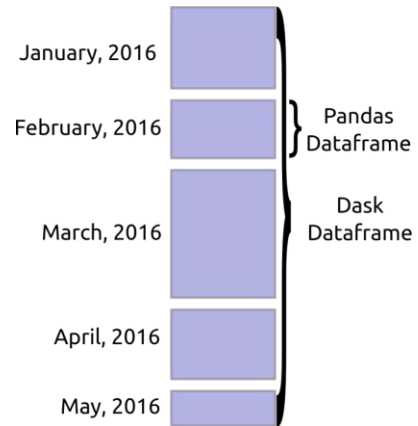


SST Anomaly in Nino 3.4 Region (5N-5S,120-170W)





Dask – Connexion des utilisateurs de Python au HPC



Utilisateur

Ecrit un code de haut niveau
(NumPy/Pandas/Scikit-Learn)

Se transforme en
graphe

S'exécute sur le HPC

Dask – Ecosystème

The screenshot displays the JupyterLab interface with a Dask cluster running. The main code cell shows the following Python code:

```
[1]: from dask.distributed import Client
client = Client()
client

[2]: import dask.array as da
x = da.random.randoms((10000, 10000), chunks=(1000, 1000))
y = x + x.T - x.mean(axis=0)
y = y.persist()

[ ]:
```

The output of the first cell shows the Client and Cluster information:

Client	Cluster
Scheduler: tcp://127.0.0.1:10881	Workers: 4
Dashboard: +	Cores: 4
http://127.0.0.1:8787/status	Memory: 16.68 GB

The interface includes several monitoring panels:

- Task Graph:** A graph showing the execution flow of tasks across 4 workers. The x-axis represents time (0 to 4 seconds), and the y-axis represents task ID (0 to 100). Blue bars indicate released tasks, and red bars indicate memory usage.
- Task Stream:** A stream of task execution bars, showing the progress of individual tasks over time.
- Dask Load:** A bar chart showing the bytes stored in memory, currently at 1.79 GB.
- Dask Progress:** A table showing the progress of various tasks:

Task Name	Progress
random_sample	100 / 100
mean_chunk	100 / 100
add	100 / 100
sub	100 / 100
transpose	90 / 90
mean_combine...	30 / 30
transpose-add	10 / 10
mean_agg-agg...	10 / 10

Atelier sur HAL

Ces différents notebooks permettent de découvrir les concepts présentés plus haut. Les différentes cellules présentées doivent juste être exécutées. Ouvert à tous publics : débutant ou non.

Présentation des concepts de Dask

- [pangeo_01_dask.ipynb](#)

Analyse des grilles DUACS

- [pangeo_02_duacs_l4.ipynb](#)

xarray

- [pangeo_03_xarray.ipynb](#)

Atelier sur HAL

Ces différents notebooks permettent de découvrir et d'approfondir les concepts manipulés par dask: graphes, tableaux, scheduler. Les différentes cellules présentées doivent juste être exécutées, et il y a quelques exercices. Demande une connaissance plus approfondie de Python. Mais tout le monde peut les faire.

Les graphes de calcul

- 01_dask_delayed.ipynb

Les ordonnanceurs

- 02_distributed.ipynb

Les tableaux

- 03_array.ipynb

Analyse des collections

- 04_bag.ipynb

Autres aspects

Interpolation des données

Co-localisation

D'autres besoins ?