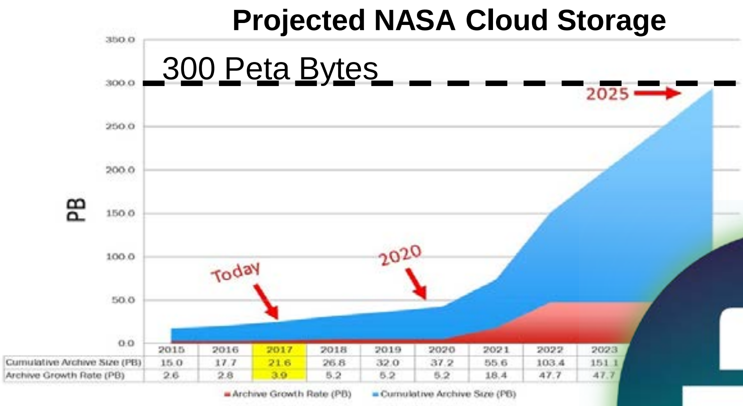
Pangeo community platform and its use at CNES

BiDS 2019, Guillaume Eynard-Bontemps, CNES/Pangeo

CNES
CENTRE NATIONAL D'ETUDES SPATIALES

Pangeo goals and motivation



	2015	2016	2017	2018	2019	2020	2021	2022	2023
Cumulative Archive Size (PB)	15.0	17.7	21.6	26.8	32.0	37.2	55.6	103.4	151.1
Archive Growth Rate (PB)	2.6	2.8	3.9	5.2	5.2	5.2	18.4	47.7	47.7

Problems

- Data volume crisis in (geo)sciences
- Software multiplication, non reproducibility
- Many copies of the same datasets
- Local vs HPC vs Cloud
- Technology gap: industry vs academia

Mission

To cultivate an ecosystem in which the next generation of open-source analysis tools for the geosciences can be developed, distributed, and sustained.

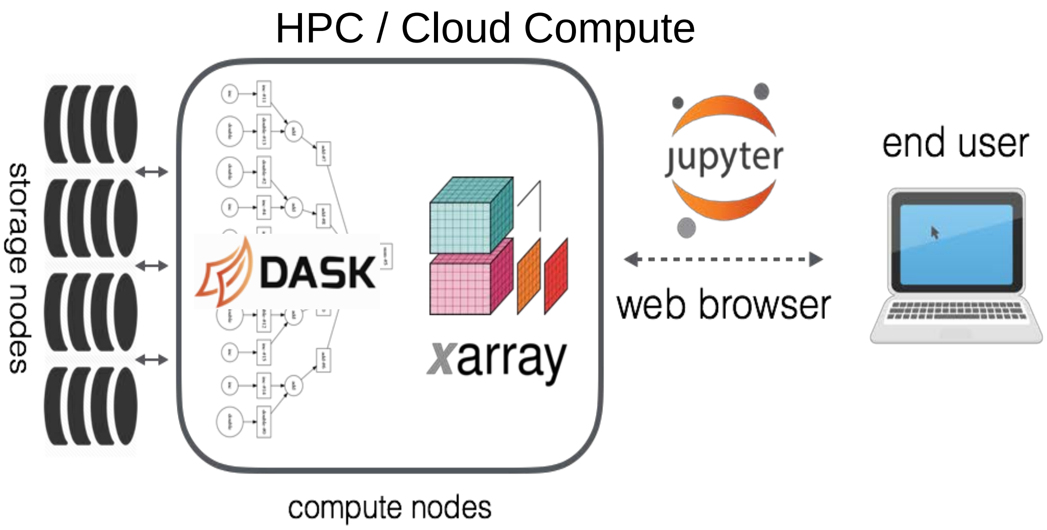
Goals/vision

- Foster collaboration around the open source Scientific Python ecosystem:
  - open and collaborative development
  - Welcoming and inclusive culture
- Support the development with domain-specific (geo)science and transverse packages
- Improve scalability of these tools to handle gigabytes to petabytescale datasets

EarthCube
TRANSFORMING GEOSCIENCES RESEARCH
NSF

Pangeo ecosystem

- Set of tools that will facilitate science at all scales
- Platform agnostic
- The core of the Pangeo ecosystem includes:
  - **Xarray** (data-model and toolkit for working with N-dimensional labeled arrays)
  - **Dask** (parallel computing)
  - **Jupyter** (interactive computing)
- Extensible: Series of 3rd party packages that build on top of core libraries
- Flexible: Individual components may be swapped in/out



Examples of 3rd party packages in the Pangeo Ecosystem:

- Data discovery
- Regridding and GIS
- Vector calculus
- Signal processing
- Thermodynamics

Pangeo diversity

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 CHEYENNE	Cloud Google Cloud Platform	Local

Pangeo vs state of the art

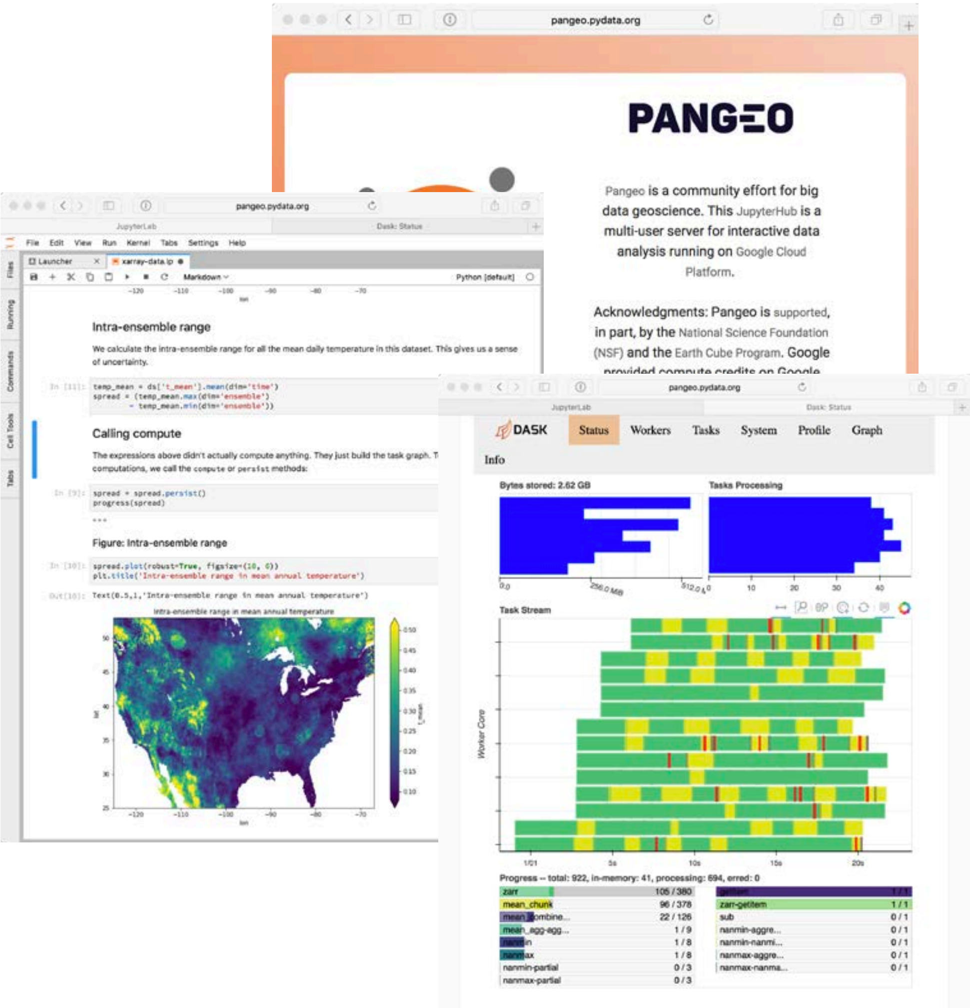
Spark	vs Dask
Mature	Less Mature
Robust	Pretty strong
JVM/Python	Python only
Query optimized	Science optimized
Collections & Dataframes	Collections, DF, Arrays, Futures…
Python overhead	Python only
For big tabular data	For science data
Hadoop/Cloud/HPC	Hadoop/Cloud/HPC

I don't know much about Array DBs!!!

PANGEO vs …y databases *(Rasdaman, SciDB…)*

- Laptop to cluster
- Serverless
- NetCDF/TIFF no ingestion
- Scales with Dask
- Python only
- Can build array db with Pangeo (Open data Cube)

Pangeo public cloud deployment



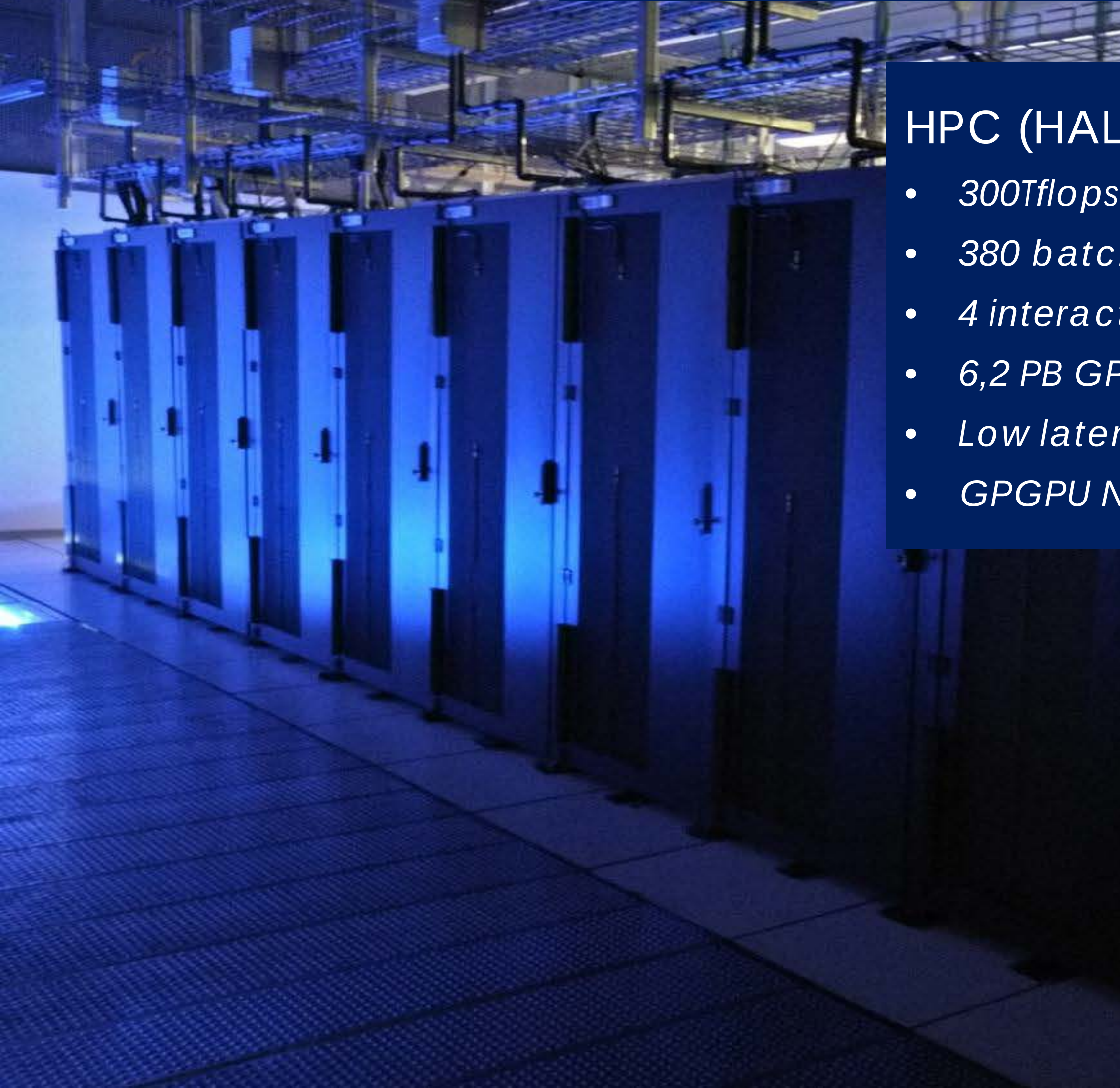
hub.pangeo.io

pangeo.binder.io

JupyterHub/BinderHub running on the Google Cloud

- Kubernetes for both Jupyter and Dask-distributed
  - Dask-kubernetes
- Exploring/evaluating:
  - Cloud storage
  - User environment customization
  - Data discovery
- Kubernetes Helm-chart (github.com/pangeo-data/helm-chart)
- CI/CD with Hubploy and CircleCI
- Deployments exist on AWS and Azure.

CNES Datacenter overview



HPC (HAL)

- *300Tflops*
- *380 batch servers / 8400cores*
- *4 interactive servers pre/post processing w/ GPU*
- *6,2 PB GPFS / 200TB burst buffer/ 50GBs bandwidth*
- *Low latency network*
- *GPGPU Nvidia Volta V100*

HPC DRSF (Ktulu)

- *20 Tflops*
- *2 interactive servers pre/post processing w/ GPU*
- *24 servers / 576 cores*
- *120TB GPFS*
- *Low latency network*

HPC usecases in CNES

Two main kinds of processing

Numerical simulation (HPC)

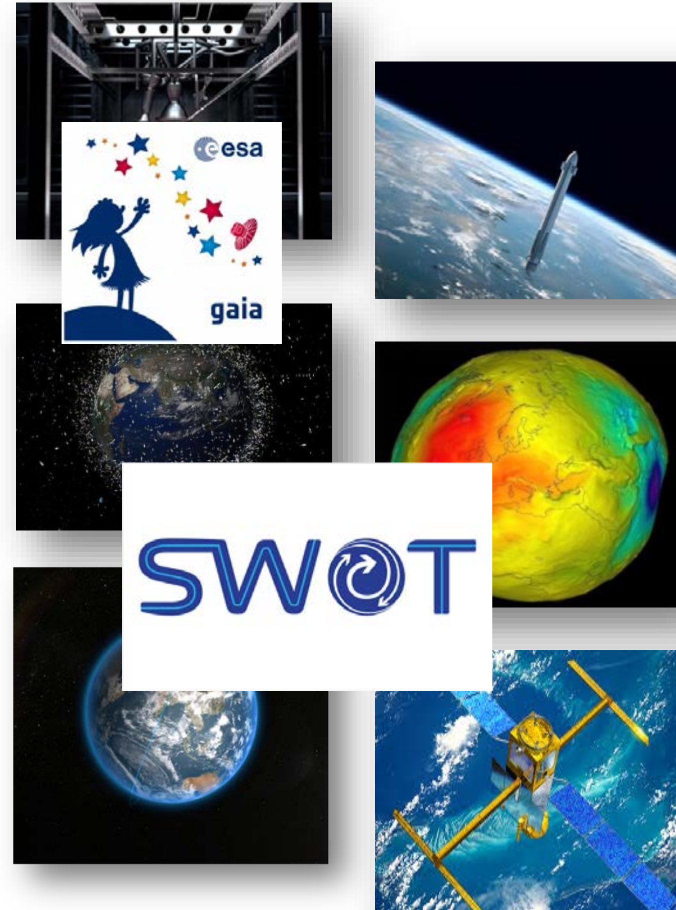
- Upstream phase, R&D
- Highly optimized technics
- Fine grain parallelism

Trends : multiscale, multiphysics

Data Processing (HTC)

- Downstream phase, operation
- Sensors data → scientific data
- Coarse grain parallelism

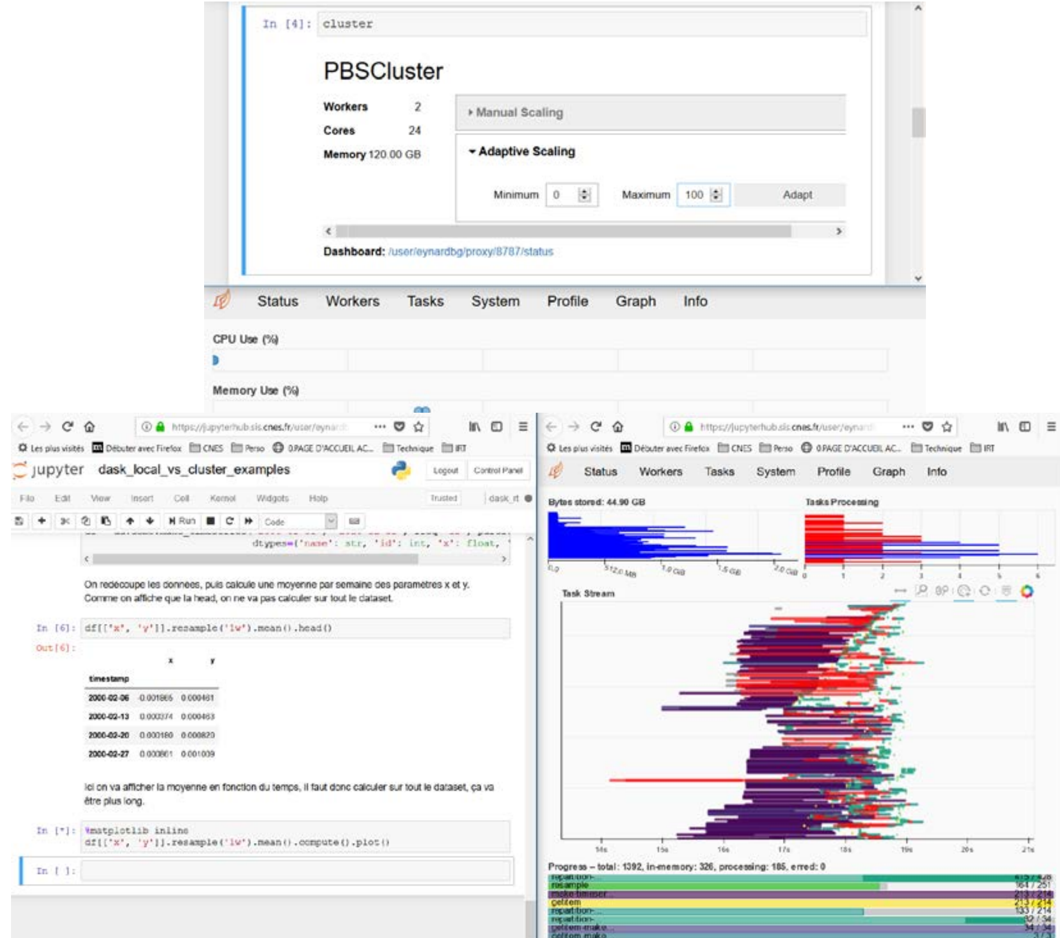
Trends : data volume explosion



Pangeo at CNES

cnes

- JupyterHub and notebooks for interactive computing
    - Hub on a VM with qsub access
    - Batchspawner, Wrapspawner
- dask.distributed: parallel workers across many HPC nodes
- Xarray for computational toolkit and I/O
- New tool for deploying dask clusters on HPC: **dask-jobqueue**
    - Start a cluster from a notebook
    - Interactive (or not) distributed computing
    - Auto scaling capabilities



Demo : Dask and dask-jobqueue basic example

```python
from dask_jobqueue import PBSCluster
cluster = PBSCluster(cores=4, memory="15GB", local_directory='$TMPDIR',
                     queue='qdev', project='DaskTest',
                     walltime='01:00:00', interface='ib0')
cluster.adapt(minimum=2, maximum=16)

from dask.distributed import Client
client = Client(cluster)

def my_costly_simulation(args):
    ## Do something
    return sum(args)

input_params = [...]

futures = client.map(my_costly_simulation, input_params)
results = client.gather(futures)
```

cluster manuellement, ou de manière automatique (Adaptive cluster). On peut aussi configurer ça via des lignes des codes comme montré ci-dessous.

```
cluster
```

Spécification manuelle de la taille du cluster. Le paramètre donné est le nombre de workers Dask.

```
cluster.scale(8)
```

Indiquer qu'on veut un cluster adaptatif (dont la taille variera en fonction de la charge).

Lancer la cellule suivante (ou précédente) doit modifier les informations affichées dans le widget.

```
cluster.adapt(minimum=2, maximum=16)
```

Déclaration du client

Afin que toutes les API de dask (dataframe, delayed, bag ...) utilisent le cluster Dask qu'on a démarré, il est

Dask Memory Use

jupyter

Control Panel

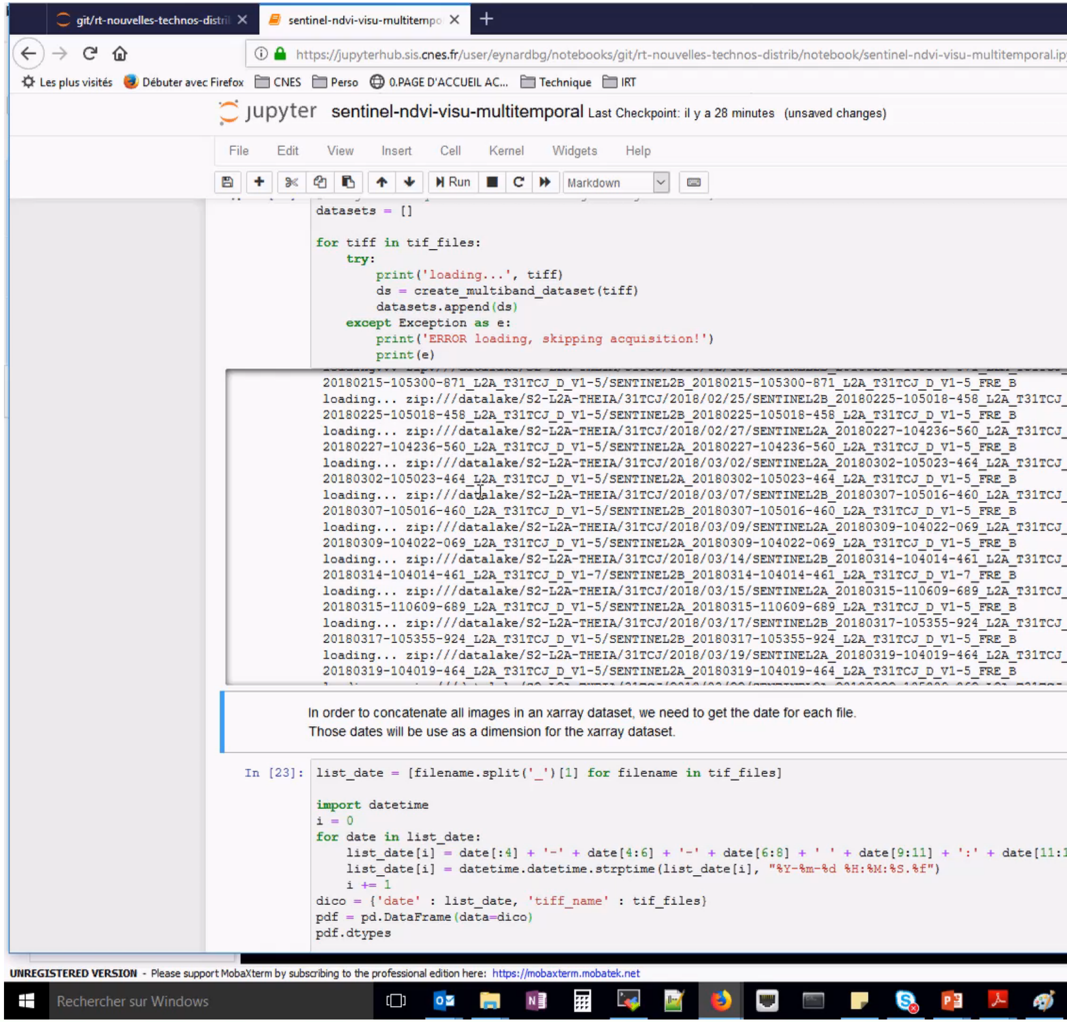
404 : Not Found

You are requesting a page that does not exist!

Takeways:

- No enormous job arrays with short duration jobs. Bigger reservations. Scheduler relief.
- No use of shared storage for syncing results → less bad IOs.
- Simplified high level orchestration, all with Python and Dask

Demo : Multi temporal NDVI with Xarray



Takeways:

- Interactive processing over temporal series with Jupyter/Dask/Xarray
- Interactive vizualisation
- Recompute upon user Action
- Statistical analysis
- JPEG2000 is not a good interactive processing format
- Share data in cloud/distributed processing ready format

Conclusion, Useful links

Conclusions

- Pangeo ecosystem greatlty facilitates distributed computing and data analysis at scale
- It changes ways of doing it too
- Non monolithic platform built on top of existing Scientific Python stack and new related packages
- Community is always here to help
- Dask more versatile and easy to use than Spark.

Next steps

- Broaden users and use cases at CNES
- Encourage people to get in touch with Pangeo community
- Work in cooperation with others (Ongoing with Ifremer and CLS on SWOT aval data processing)

Pangeo wesite and discussions:

https://pangeo.io

https://github.com/pangeo-data/pangeo/issues

https://medium.com/pangeo

Pangeo Example + Binder:

https://github.com/pangeo-data/pangeo-example-notebooks

http://binder.pangeo.io/v2/gh/pangeo-data/pangeo-example-notebooks/master

Dask jobqueue:

https://github.com/dask/dask-jobqueue

Dask simple examples:

https://github.com/dask/dask-examples

My email

Guillaume.Eynard-Bontemps@cnes.fr