THE PANGEO BIG DATA ECOSYSTEM AND ITS USE AT CNES

Guillaume Eynard-Bontemps[1], *Ryan Abernathey*[2], *Joseph Hamman*[3], *Aurelien Ponte*[4], *Willi Rath*[5]

[1]Centre National d'Etudes Spatiales (CNES), Toulouse, France,
[2]Columbia University / Lamont Doherty Earth Observatory, New-York, USA,
[3]National Center for Atmospheric Research (NCAR), Boulder, USA,
[4]Ifremer, Univ. Brest, CNRS, IRD, Laboratoire dOcanographie Physique et Spatiale (LOPS), IUEM, Brest 29280,
[5]GEOMAR Helmholtz Centre for Ocean Research Kiel, Kiel, Germany.

ABSTRACT

Pangeo[1] is a community-driven effort for open-source big data initially focused on the Earth System Sciences. One of its primary goals is to enable scientists in analyzing peta-scale datasets both on classical high-performance computing (HPC) and on public cloud infrastructure. In only a few years, Pangeo has grown into a very productive community collaborating on the development of open-source analysis tools for ocean, atmosphere and climate science, and beyond. It provides a set of example deployments based on open-source Scientific Python packages like Jupyter[14], Dask[12], and xarray[13] that not only serve as working environments for actual scientific work, but also bring together developers with users and their actual use-cases.

In this paper, we first describe Pangeo: its motivation, community, the underlying technology stack and associated deployments, the different applications and the ongoing work. We then present its impact on the work of scientists from CNES: the context, an HPC deployment, and several use cases. We will conclude with a future outlook for Pangeo in this agency and beyond.

Index Terms— Pangeo, Dask, Jupyter, HPC, Cloud, Big Data, Analysis, Open Source

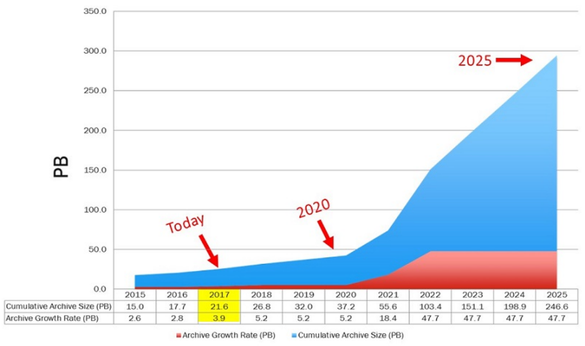
1. PANGEO

1.1. Motivations, mission and goals

The geoscience community is facing several building crises:

- Big Data: datasets are growing exponentially (see 1) and legacy software tools for scientific analysis cannot handle them. This is a major technological obstacle to scientific progress.

- Technology Gap: a growing gap between the technological sophistication of industry solutions (high) and scientific software (low).



	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025
Cumulative Archive Size (PB)	15.0	17.7	21.6	26.8	32.0	37.2	55.6	103.4	151.1	198.9	246.6
Archive Growth Rate (PB)	2.6	2.8	3.9	5.2	5.2	5.2	18.4	47.7	47.7	47.7	47.7

Fig. 1. Projected NASA EOS Cloud storage[2].

- Reproducibility: a fragmentation of software tools and environments renders most geoscience research effectively unreproducible and prone to failure.

Pangeo's core mission is to cultivate a collaborative environment in which the next generation of open-source analysis tools for ocean, atmosphere, climate and eventually other sciences can be developed, distributed, and sustained. These tools must be scalable in order to meet the current and future challenges of big data, and these solutions should leverage the existing expertise both inside and outside of the geoscience community.

To accomplish this mission, Pangeo collaborators have identified three specific goals.

- Foster collaboration around the open source Scientific Python ecosystem for the oceanographic, atmospheric, terrestrial, and climate science.

- Support the development with domain-specific geoscience packages.

- Improve scalability of these tools to handle petabyte-scale datasets on HPC and on cloud computing platforms.

1.2. Community

Rather than being controlled by a single organization, Pangeo is a community-driven project, in the model of successful open-source software like Linux, Python, and Jupyter. In this spirit, all Pangeo discussions and products occur on GitHub[3], where anyone can easily get involved in the community. As of today, Pangeo spans a list of people from different governmental agencies, universities, or private companies, and from different countries (the USA, the UK, France, Austalia to name a few). By making the collaboration as broad as possible, Pangeo successfully leverages shared expertise to accomplish things no institution could alone.

1.3. Technology contributions to the Scientific Python stack

Pangeo's software ecosystem fits directly into the Scientific Python stack, involving packages such as Numpy, Pandas, or Sickit-learn. Three Python software projects are at the core of Pangeo:

- Dask is a library for parallel and distributed computing that coordinates with Pythons existing scientific software ecosystem. In many cases, it offers users the ability to take existing workflows and quickly scale them to much larger applications.
- Xarray is the interface for working with big datasets: it provides a Pandas-like API for labelled n-dimensional arrays and has backends for established and upcoming self-describing community data file formats and access protocols like netCDF, GeoTIFF, OPeNDAP, and Zarr. Xarray transparently integrates Dask arrays and hence enables users to easily scale their work to massively parallel computations.
- Jupyter: Jupyter notebooks and Jupyter Lab enable interactive computing and analysis from a web browser, and JupyterHub adds multi-user support. Jupyter notebooks are quickly becoming the standard open-source format for interactive computing not only in Python, but also in languages such as Julia and R.

There are several developments that either started in or are fueled by the Pangeo community:

- Modules to automatically deploy Dask distributed clusters in various infrastructures are being developed: dask-kubernetes for Kubernetes clusters and thus for the public cloud, dask-jobqueue[4] for HPC systems using scheduler such as Slurm, PBS Pro or LSF, and dask-yarn for YARN clusters (i.e. Hadoop).
- The integration of a Zarr backend in xarray paved the way to directly access cloud-based object storage in parallel computations.
- ...

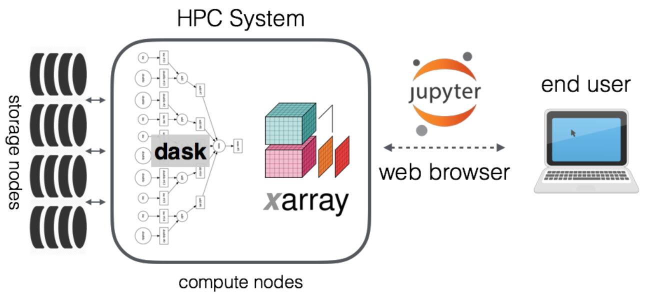


Fig. 2. Pangeo platform main components.

1.4. Deployments

The GitHub community offers online documentation, scripts and other tools to link them together in order to deploy a Pangeo platform (see 2) and put the software stack on HPC systems or in the public cloud. The main elements allowing to build and use the platforms along the core libraries are first a set of scripts and documentation that allows automatically creating the necessary cloud infrastructure. Currently available for Google Cloud Engine (GCE), and work is underway for Amazon Web Services (AWS) compatibility. Terraform software and CI/CD tooling are also under active development and offer the potential for automating these tasks.

1.5. Applications

Pangeo's first scientific domain are the earth-system sciences. There are already many applications documented in these fields. Some real science use cases can be found online[5]; these examples, along with their data, can be directly executed on cloud infrastructure and be controlled from a web browser.

Other scientific domains have developed an interest in the Pangeo approach, including:

- Satellite imagery analysis: a recent blog post[6] explains how to use Pangeo for processing and analysing in real time Landsat imagery. NASA has decided to fund Pangeo initiative for applying the ecosystem on remote sensing datasets that are being pushed on AWS[10].
- Astronomy: several scientists are already using the web platform to explore Gaia DR2 data on GCE.
- Neuroscience: there is some on-going work to use Pangeo for analysis on human brain or electrophysiological datasets.

As the computational resources necessary for most of these applications can only be met with distributed computing, the data must be in a cloud or distributed storage friendly format, like Zarr for multi-dimensional data, Cloud optimized

Geotiff for satellite imagery, or Parquet for tabular data. See [7] and [11] for more details on this crucial point.

1.6. Ongoing work

The community and the technological stack are currently advanced along the following lines:

- To facilitate learning and using the Pangeo software stack at scale, there is a Binder deployment[8] providing access to a cloud-based platform with a single click.
- To actively foster collaboration and to make sure that Pangeo remains a diverse and inclusive project, an emerging focus is on community governance.
- To better cope with domain-specific software and hardware requirements at this stage of the development, the formerly single example deployment of Pangeo has been split into many specific environments. This helps with reaching out to new fields and yields increasingly automated work flows for cloud environment updates using modern tooling for Continuous Deployment with Hubploy and CircleCI.
- To reach out to possible new collaborators but also to enable more and more interested scientists, Pangeo members offer workshops in various settings.

2. CNES DEPLOYMENT AND USE CASES

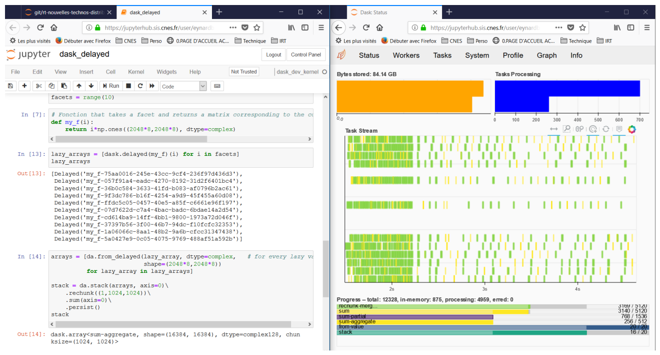
2.1. Context and projects

The Centre National d'Etudes Spatiales (CNES) is the government agency responsible for shaping and implementing France's space policy in Europe. As such it covers a wide range of subjects: Ariane launcher, Sciences, Earth observation, telecommunication and defence.

On the ground segment processing side, we are involved in several Big Data projects, most of them being hosted in CNES Data Center:

- Gaia data processing center for some science unit,
- Sentinel product Exploitation Platform, for sharing and online processing of Copernicus products,
- Surface Water and Ocean Topography (SWOT) mission, with French processing center hosted at CNES,
- Euclid astronomy mission, for the overall coordination of Science Data Centers.

CNES HPC is also hosting a lot of other processing involving remote sensing data, flight dynamics, altimetry or other domains.



```
facets = range(10)

In [7]: # Fonction that takes a facet and returns a matrix corresponding to the c
def my_f(i):
    return i*np.ones((2048*8,2048*8), dtype=complex)

In [13]: lazy_arrays = [dask.delayed(my_f)(i) for i in facets]
lazy_arrays

Out[13]: [Delayed('my_f-75aa0016-245e-43cc-9cf4-236f97d436d3'),
 Delayed('my_f-057f91a4-eadc-4270-8192-31d2f6401bc4'),
 Delayed('my_f-36b0c584-3633-41fd-b083-af0796b2ac61'),
 Delayed('my_f-9f3dc786-b16f-4254-a9d9-45f455a60d08'),
 Delayed('my_f-ffdc5c05-0457-40e5-a85f-c6661e96f197'),
 Delayed('my_f-07d7622d-c7a4-4bac-badc-6bdae14a2d54'),
 Delayed('my_f-cd614ba9-14ff-4bb1-9800-1973a72d046f'),
 Delayed('my_f-37397b56-3f00-46b7-94dc-f10fcfc32353'),
 Delayed('my_f-1a06066c-8aa1-48b2-9a6b-cfcc31347438'),
 Delayed('my_f-5a0427e9-0c05-4075-9769-488af51a592b')]

In [14]: arrays = [da.from_delayed(lazy_array, dtype=complex,   # for every lazy va
                           shape=(2048*8,2048*8))
          for lazy_array in lazy_arrays]

stack = da.stack(arrays, axis=0)\
    .rechunk((1,1024,1024))\
    .sum(axis=0)\
    .persist()
stack

Out[14]: dask.array<sum-aggregate, shape=(16384, 16384), dtype=complex128, chun
ksize=(1024, 1024)>
```

```
Progress – total: 12328, in-memory: 875, processing: 4959, erred: 0
rechunk-merg...                       3169 / 5120
sum                                   3140 / 5120
sum-partial                            768 / 1536
sum-aggregate                          256 / 512
from-value                              20 / 20
stack                                   16 / 20
```

Fig. 3. Computation in Jupyter and Dask dashboard.

2.2. Pangeo on our HPC System

CNES main processing platform is a modestly sized High Performance Computer named HAL. Its computional resources are: about 8000 Intel cores, 6PB storage using IBM Spectrum Scale technology, some NVidia Volta GPU. It uses PBS Pro jobqueue system to schedule the load on compute nodes and handle user and project resource sharing.

Pangeo platform has recently been deployed on the cluster, which basically means the configuration of two main components: a JupyterHub and Dask through dask-jobqueue (see 3 for a graphical overview).

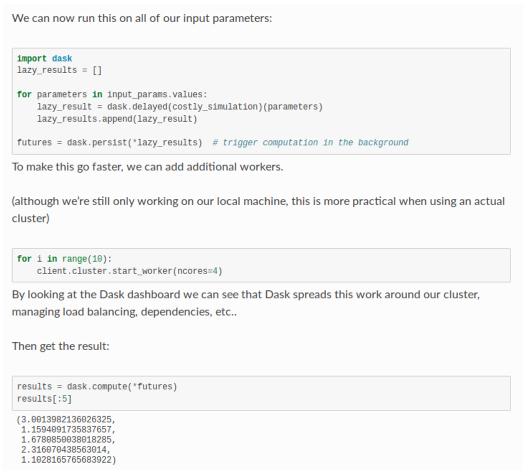
JupyterHub has been deployed on a Virtual Machine, which has direct access to HAL cluster through PBS commands. This allows configuring JupyterHub Batchspawner which launch user notebooks using PBS *qsub* command, alongside Wrapspwaner to be able to select adequate system resources for launched notebook.

CNES directly contributes to dask-jobqueue in order to improve its usability on HAL. This Python module deployment (alike Xarray or other domain scientific library) is quite simple as it can be done through conda or pip packaging system. A template configuration is proposed to all HAL users. There is also a few commands documented in order to be able to use the Python environment and thus create Dask clusters from inside Jupyter, some demonstration notebooks have been shared internally.

2.3. From embarrassingly parallel to more complex workflows with Dask

One important use of our cluster is to do batch jobs: apply the same computation or process to several inputs, which can for example be a list of files or a list of parameters. This is usually done with job arrays PBS mechanism. Results are then written into our central storage facility, and a final job will gather them and consolidate them if needed. There are three main drawbacks to this approach:

- When scaling this mechanism to numerous (say undreds of thousands) and short (under one minute) jobs,


We can now run this on all of our input parameters:

```
import dask
lazy_results = []

for parameters in input_params.values:
    lazy_result = dask.delayed(costly_simulation)(parameters)
    lazy_results.append(lazy_result)

futures = dask.persist(*lazy_results)  # trigger computation in the background
```

To make this go faster, we can add additional workers.

(although we're still only working on our local machine, this is more practical when using an actual cluster)

```
for i in range(10):
    client.cluster.start_worker(ncores=4)
```

By looking at the Dask dashboard we can see that Dask spreads this work around our cluster, managing load balancing, dependencies, etc..

Then get the result:

```
results = dask.compute(*futures)
results[:5]
```

```
(3.0013982136026325,
 1.1594091735837657,
 1.6780850038018285,
 2.316070438563014,
 1.1028165765683922)
```

Fig. 4. Embarrassingly parallel workload using Dask's Delayed API.

this can lead to PBS scheduler contention and slow responsiveness.

- This often means a lot of bash script and machinery to chain several analysis together, leading to workflows that are hardly readable and difficult to maintain.
- As results can be really small and are exchanged through a centralized storage system, this also means a lot of I/O load onto the File System and side effects for other users.

Pangeo, mainly through Dask, gives an adequate solution to all these problems (see 4 for an example). All the workflow, cluster creation, data management parts are handled from Python code. Reservation to PBS are only done one time using dask-mpi or by bigger blocks using dask-jobqueue for long running worker process. No need to write or exchange data through disk, all data management is done through memory or network with TCP over high-speed networks (i.e. Infiniband). This allows users to analyse the result of a simulation in the same piece of code where we launched it, and eventually gather only the reduced valuable part for later use. The result of all this is elegant and simple Python code, which can scale easily to thousands of cores and does not stress our HPC cluster scheduling components.

2.4. Interactively simulating remote sensing data through dask array

We also used Pangeo to implement the generation of simulated data from a remote sensing satellite. The main idea of the computation is that we need to generate many large two dimensional arrays, each 2 or 4 GB in size, that represent a part of the simulated output, and them sum all of those together. A first implementation was previously done using Python's NumPy and multiprocessing modules, but it was difficult to do the sum in memory, to scale beyond one compute node, and it involved temporary file writing.

Thanks to Jupyter and Dask, we were able to rapidly and interactivly prototype a new algorithm version. Dask solves all of the previous algorithm problems, taking advantage of NumPy array chunking and efficient lazy task execution in graphs for chained operations. One problem remains with our current simple solution: it does not optimize the memory usage, as we need to create all arrays before rechunking and summing them together, but thanks to Pangeo we will fix this soon enough.

3. CONCLUSION

Pangeo is a powerful ecosystem which enables science at scale on Cloud or on premise infrastructure. We encourage every lab, governement agency, or even industry players to take a look at what it provides. The community is open and eager to entrain new users and collaborators. At CNES, we decided to focus on this tooling for our shared computing infrastructure, and it is already showing its power and its benefit. Ongoing work is focusing on developing more and more use cases with Pangeo to identify where it is most useful, and possible limits to the software stack. A collaboration with the french national institute for marine sciences (Ifremer where the Xarray/Dask environment is already used for science[9]) on SWOT data valorization is on going and should foster these developments. We are also trying to participate actively in the community and share our vision and needs, helping to steer the common effort in a direction beneficial to CNES researchers.

REFERENCES

REFERENCES

[1] Ryan Abernathey, Kevin Paul, Joseph Hamman, Matthew Rocklin, Chiara Lepore, Michaem Tippett, et al. (2017): Pangeo NSF Earthcube Proposal.

[2] Mark McInerney, ESDIS Project Deputy Project Manager/Technical: EOSDIS Cloud EvolutionNASA EOSDIS web site.

[3] Ryan Abernathey et al: Pangeo github project issue tracker.

[4] Joseph Hamman, Matthew Rocklin, Jim Edwards, Guillaume Eynard-Bontemps, Loc Estve (2018): Scalable interactive analysis workflows using dask on HPC Systems.

[5] Pangeo community: Pangeo use cases.

[6] Scott Henderson, Daniel Rothenberg, Matthew Rocklin, Ryan Abernathey, Joseph Hamman, Rich Signell, and Rob Fatland: Cloud Native Geoprocessing of Earth Observation Satellite Data with Pangeo.

[7] Ryan Abernathey: Step-by-Step Guide to Building a Big Data Portal.

[8] Joseph Hamman, Ryan Abernathey: Pangeo meets Binder.

[9] S. Fresnay, A. L. Ponte, S. Le Gentil, and J. Le Sommer: Reconstruction of the 3-D Dynamics From Surface Variables in a High-Resolution Simulation of North Atlantic. DOI: 10.1002/2017JC013400.

[10] Joe Hamman: Pangeo applications for NASA Earth Observing Data.

[11] Matthew Rocklin: HDF in the Cloud — Challenges and solutions for scientific data.

[12] Dask Development Team: Dask: Library for dynamic task scheduling, 2016.

[13] Stephan Hoyer and Joseph J. Hamman: xarray: N-d labeled arrays and datasets in python. *Journal of Open Research Software*, 5, apr 2017. doi: 10.5334/jors.148.

[14] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing: Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016. doi: 10.3233/978-1-61499-649-1-87.