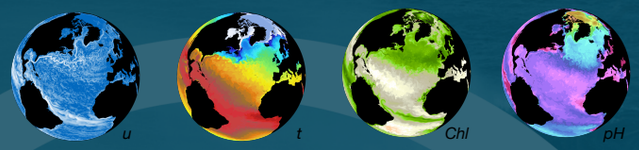
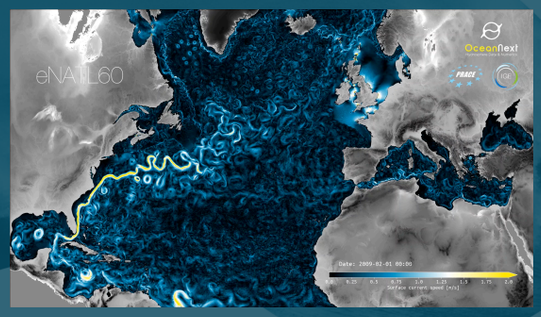
Hello everyone, my name is Aurélie, I work in France in a lab dealing with geosciences and environment and also in a small company called Ocean Next. I am going to speak to you about the tests I have performed on the french computing centers by deploying the PANGEO tools on them.

Our scientific objects :

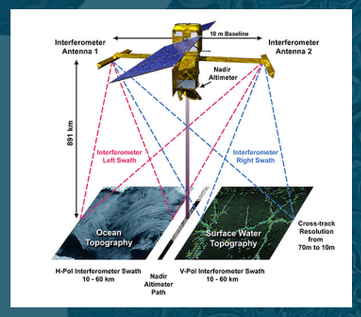


- oceanic processes
- climate projections
- operational systems

Ocean Modelling



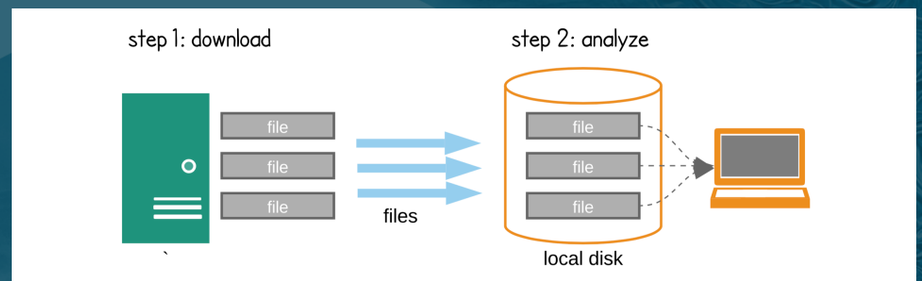
Satellite Observations : ex SWOT



Higher spatial, temporal resolution, more processes (tides) = more and more data to analyse

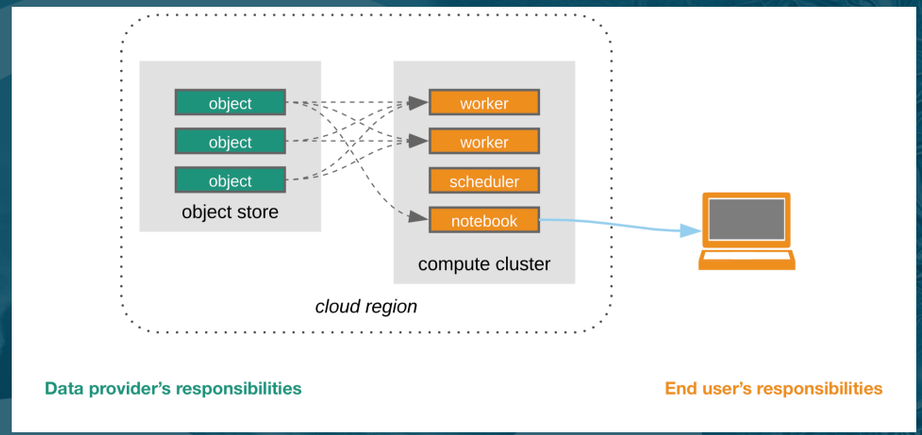
So first, in my team we are studying the ocean, its processes and its role in the climate engine by running simulations and comparing them to satellite observations. The modeling and the observing are getting more and more precise and are producing more and more data to analyse.

Problem :



So that our traditional way of dealing with data is becoming obsolete. In fact, our simulations and observations are traditionally stored on facilities on which we perform some post processing or extractions and we download the data on our personal computer in order to do the scientific analysis. Now, the post processing part is getting slower and is even failing because of memory issues and the downloading part is no longer possible given the size of the datasets

Solution :



We want to be more efficient and perform the scientific computations and analyses directly where the data is stored, that means in our case some High Performance Computing centers where we do the actual modeling, or on cloud servers where the observations are stored. This approach prevents unnecessary downloading and also allows the use of the computing ressources available on the HPC or on the clouds

How to implement it :

PANGEO

A community platform for Big Data geoscience

- software ecosystems
- tutorials and guidelines
- live community (IRL, github, medium, discourse ...)

This approach is strongly supported by the international PANGEO group that offers expertise, practical tools and methods to achieve the transition.

In practice, for our studies :

OCCIGEN : parallel Bull supercomputer
3,5Pflop/s, Infiniband network
+3000 CPU nodes - 24/28 cores - 64/128 Go
4 CPU/GPU nodes - 28 cores - 256Go

JEAN-ZAY : parallel HPE supercomputer
28Pflop/s, Intel Omni-Path network
+1500 CPU nodes - 40 cores - 192 Go
4 CPU/GPU nodes - 48 cores - 3To

HAL : intermediate size cluster, 460 nodes,
Infiniband network, jupyterhub service

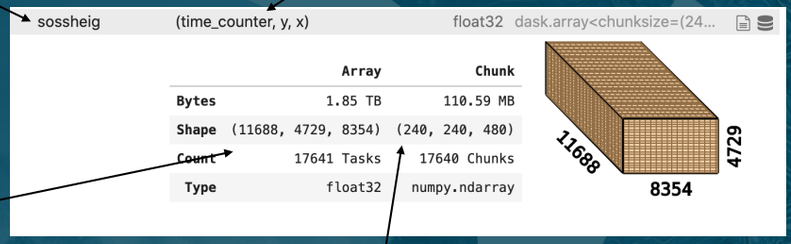
We only have to adapt the methods to the machines we have access to. For french oceanographers, various HPC centers are available. I will talk about this 3 deployments, but I also tested smaller scale centers and lab-level or team-level servers. The more natural HPC center to start on is the CINES center where most of our simulations are produced and stored. The more recent and very performant supercomputer Jean Zay is also used in our community to produce high resolution modeling of the ocean. Last a more intermediate size cluster has been tested because it is a platform commonly used by the french spatial agency partners for calibrating and distributing satellite observations.

A PANGEO deployment on this machines consists in installing, via our favorite package manager, these 3 libraries jupyter, xarray and dask that will operates together to allow parallel computations on datasets bigger than machine memory.

The data :

Sea Surface Height

16 months of data for the whole North Atlantic



hourly, 1/60° resolution

data chunked along the 3 dimensions

The test I replicate on the machines is involving a 3D (time and 2D) field of sea surface height, an output of North Atlantic modeling at very high resolution. the whole field is 1,85TB big. We would typically want to compute spatial or temporal mean of this dataset to compare with satellite observations in order to validate surface circulation and variability.

NETCDF vs ZARR format :

- object storage of chunked, compressed N-D arrays
- access to the data in parallel
- metadata in a single location
- required for cloud computing

what about the performance on HPC ?

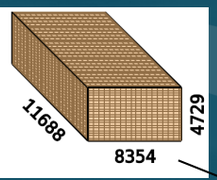
Deployment	Format	Opening	Computing mean
Occigen	netcdf	9.78s	2h25
	zarr	893ms	5min30
HAL	netcdf	4.77s	>4h
	zarr	149ms	4min47

Zarr format is also better for HPC

The original format of our model outputs is Netcdf4/HDF5. We know that for a cloud storing the zarr format is preferred for it is an object storage. We therefore converted the dataset I just presented into the zarr format. We compared the results of a temporal mean over it in both format, in the exact same conditions. Also the same chunking is applied when opening the netcdf files than the one applied during the construction of the zarr dataset. The opening of the dataset is faster and the performance of the computation is far better with zarr. But you have to keep in mind that the chunking operation is done once and for all at the construction of the zarr while it is part of the computation and will be done each time we use the netcdf files.

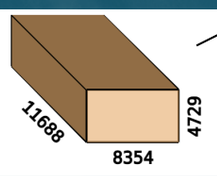
The chunking :

Chunk
110.59 MB
(240, 240, 480)
17640 Chunks
numpy.ndarray



Chunksize	Opening	Computing temporal mean	Computing spatial mean
240x240x480	90ms	4min52 (41761 Tasks)	4min17(41602 Tasks)
1x4729x8354	158ms	14min39s (27275 Tasks)	4min53 (35065 Tasks)

Chunk
158.02 MB
(1, 4729, 8354)
11688 Chunks
numpy.ndarray

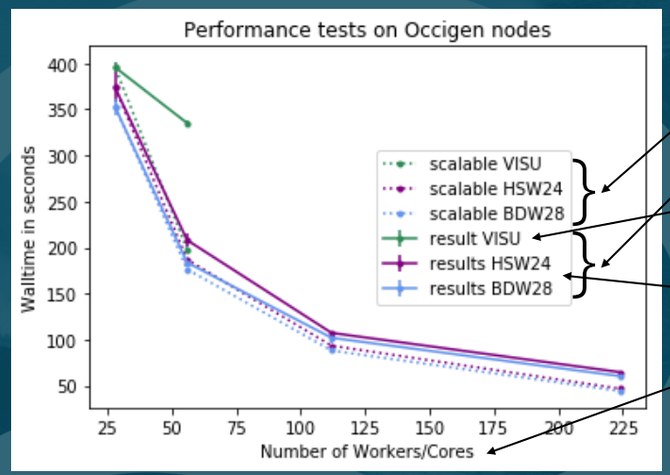


Advice : Chunk along all the dimensions when building the zarr, it will be more flexible for all operations

By the way, the chunking process is a very sensitive operation. We tried chunking along all dimensions and along the time dimension only, the chunk size being roughly of the same size. We tried a spatial and a temporal mean on this two zarr datasets. The temporal mean over the dataset that is chunked along the time dimension only is significantly slower than when we chunked along all dimensions. The spatial mean takes the same amount of time for both datasets, because the time dimension is chunked in both cases.

Before going into the details of the performance tests results, I wanted to clarify a bit of the vocabulary used with the dask library. To have access to the ressources of the machine we are working on, we have to define a cluster and a client. A cluster has a number of workers, cores and some memory available. It will distribute the tasks via the scheduler. We either are deploying the jupyter notebook directly on a computing node and then a Local Cluster is defined or we request the nodes via dask-jobqueue that will launch a job. Here is an example with the SLURM queuing system. We then have to wait for the jobs to be running to proceed with the computation. We want to know if the computation will be going faster when we increase one or more of the three parameters : number of workers, number of cores and amount of memory.

Results on occigen - CINES :



results if the nodes were scalable

actual results

visualisation node : 1 at a time, LocalCluster

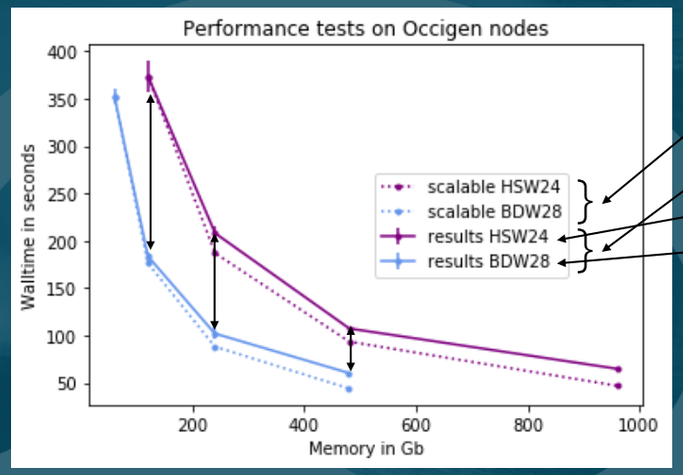
computation nodes : dask-jobqueue

same number of cores and workers, option threads_per_worker=1

better results with computation nodes, close to scalability for cores/workers

So here are the results for the occigen deployment. 2 types of computing nodes are available through dask-jobqueue and one visualisation node only is available via localcluster. We selected the option threads_per_worker equal to 1, and plotted the results against the number of cores or workers which are the same. We drew in dashed lines what the results would be if the machine was scalable in terms of number of workers/cores. The results are better for the computing nodes and close to scalability.

Results on occigen - CINES :



results if the nodes were scalable

actual results

computation node : 128Go/node

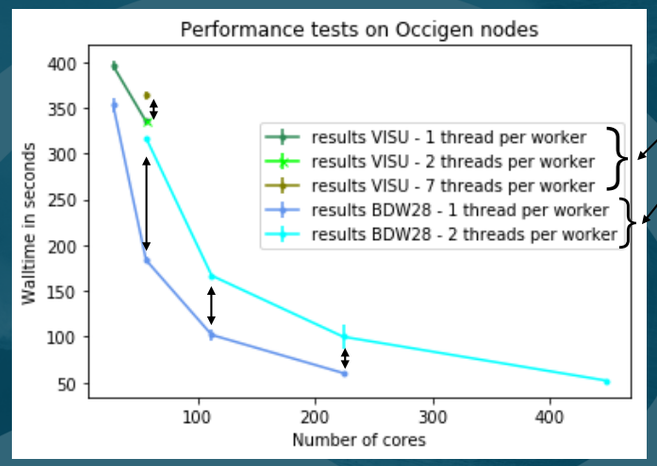
computation node : 64Go/node

same number of cores and workers, option threads_per_worker=1

better results with computation nodes with lower memory, close to scalability for memory

In terms of memory, the computation nodes differ : the ones called HSW24 are holding twice the memory of BDW28 but they show significantly worst results because with the same amount of memory, twice the cores is used in BDW28. Both are still close to scalability memory wise.

Results on occigen - CINES :



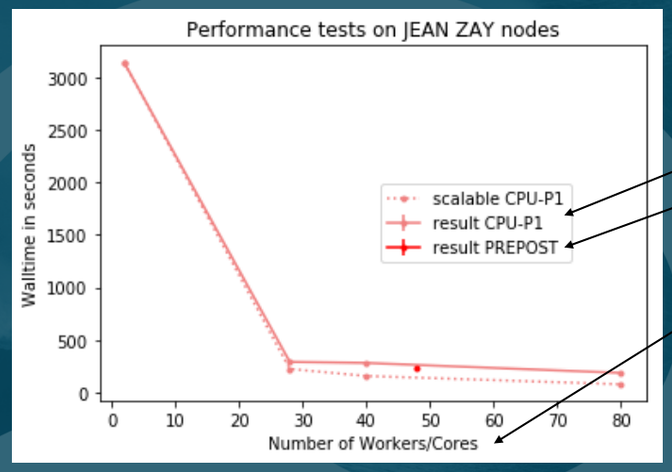
visualisation node : 56 cores max

computation nodes : 56 cores/node

better results with computation nodes and mono-thread (1 core per worker)

Now I wanted to see the impact of multi-threading, meaning in this case several cores per workers in the dask cluster. It looks like mono-threading produce better results for this deployment.

Results on jean-zay - IDRIS :



no dask-jobqueue, salloc + LocalCluster

CPU nodes : 161Gb memory

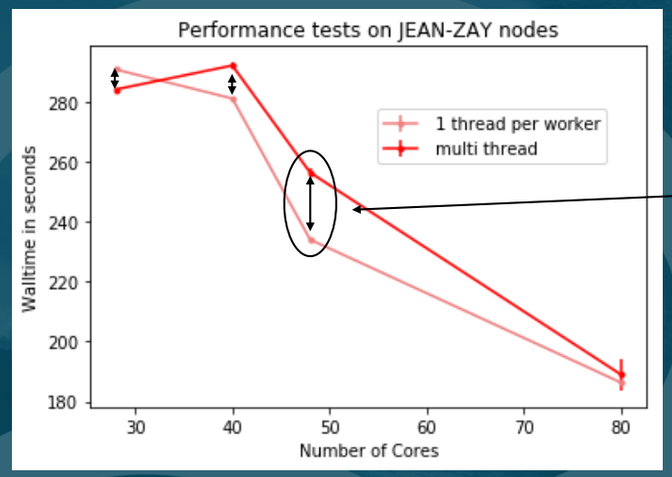
pre-post processing node : 1 at a time, 2.2Tb memory

same number of cores and workers, option threads_per_worker=1

cpu nodes almost scalable, no better results with big memory of prepost node

We move on to newest french supercomputer Jean-Zay. The deployment there is a bit different than on occigen : dask-jobqueue is not operational yet, so we have to request nodes via salloc and deploy the dask cluster directly on the nodes. Therefore, we cannot increase the memory at will and are stuck with the amount of memory available on one node (even if we ask for multiple nodes). There are also two types of nodes: computational ones and one called prepost with a very big amount of memory 2,2TB ! But the results on the prepost node is deceivingly not better than what we can achieve with computation nodes that have 12 times less memory.

Results on jean-zay - IDRIS :

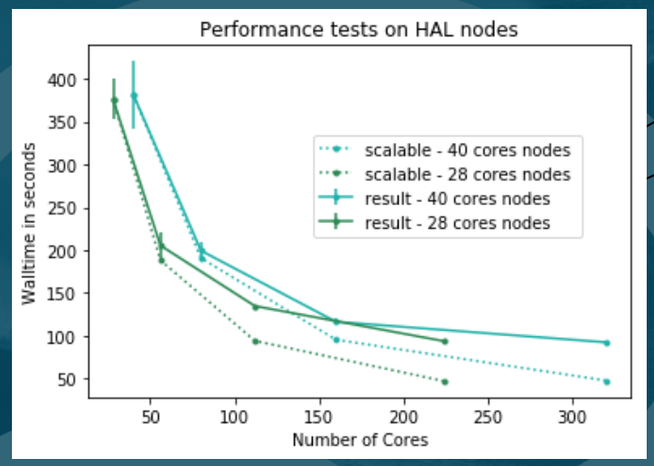


pre-post processing node

no big difference between monothread and 2 cores per workers, except for prepost node monothread better

The impact of multi-threading seems to be significant on the prepost node but not on the computation nodes. The results on jean-zay are not complete compared to the occigen ones but we are hoping that we will be taking more advantage of the ressources when dask-jobqueue is operational on it.

Results on hal - CNES :



results if the nodes were scalable

actual results

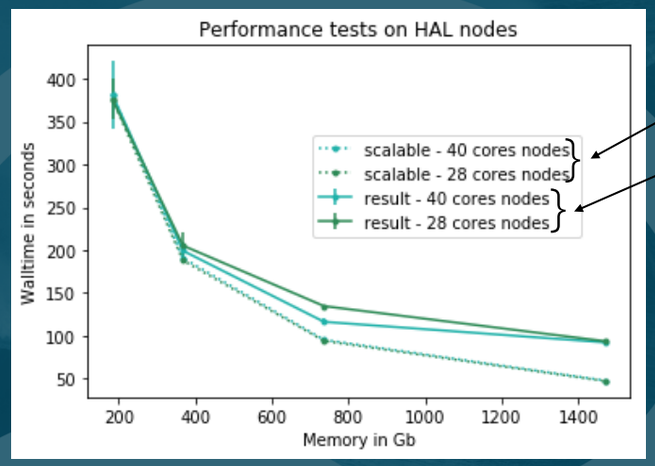
2 types of nodes accessible by dask-jobqueue:

- 28 cores, 4 cores per worker, 184 Gb
- 40 cores, 5 cores per worker, 184 Gb

better results with fewer cores per nodes (same memory) , better scalability for few cores for both type of nodes

Last, the CNES cluster, which is of an intermediate size have also been tested. Two types of computation nodes : 1 with 28 cores and the other with 40, same memory for both. The results are better for the nodes with fewer cores and for both the scalability is reached only for the low number of cores.

Results on hal - CNES :



results if the nodes were scalable

actual results

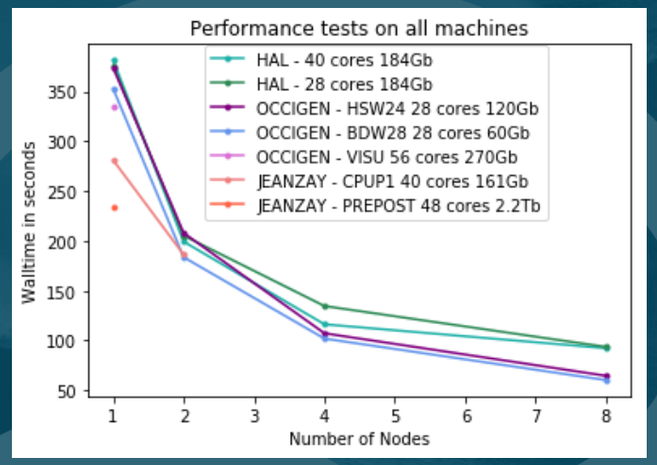
2 types of nodes accessible by dask-jobqueue:

- 28 cores, 7 cores per worker, 184 Gb
- 40 cores, 5 cores per worker, 184 Gb

same results , better scalability for lower memory for both type of nodes

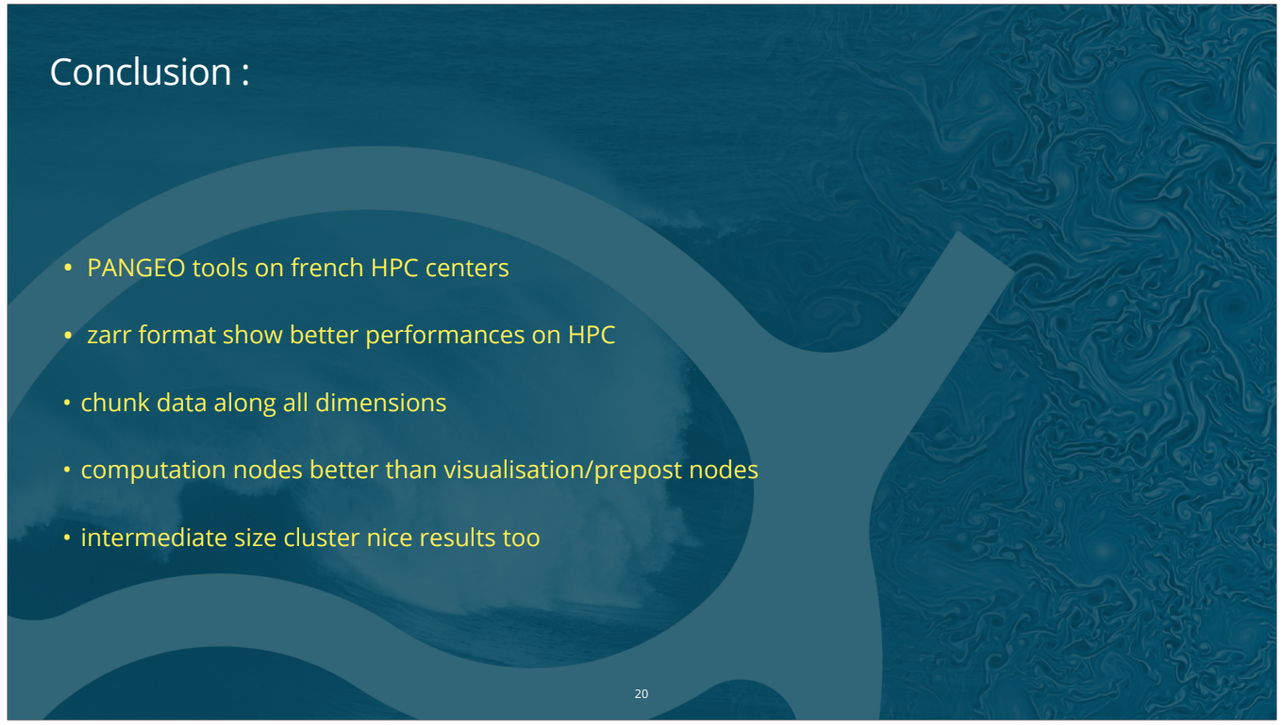
In terms of memory, the two types of nodes show similar results and same type of scalability/

Comparison between machines :



- BDW28 node from Occigen best results
- HAL cluster comparable to supercomputers
- Jean-Zay promising results (implementation of dask-jobqueue needed)
- nodes dedicated to data processing and visualisation promising results but only 1 node available

Last, the comparison between machine can only be done along the number of nodes and show that occigen bdw28 show the best results overall, really close to the other type of computing node of occigen. The smaller cluster HAL have similar results than the supercomputers. We yet have to investigate the scalability of Jean Zay further. The nodes dedicated to visualisation and pre or post processing should be more numerous to be interesting for our computation.



To summarize, we deployed the PANGEO tools directly on french HPC centers where our simulations outputs are living and we tested their performances by computing a simple operation over a large dataset. We discovered that zarr format is performant on HPC too, that chunking is key and we recommend chunking along all dimensions. For the moment the computation nodes are better and we only ask for a few of them compared to what we need for the actual modeling of the ocean. Thanks for your attention.