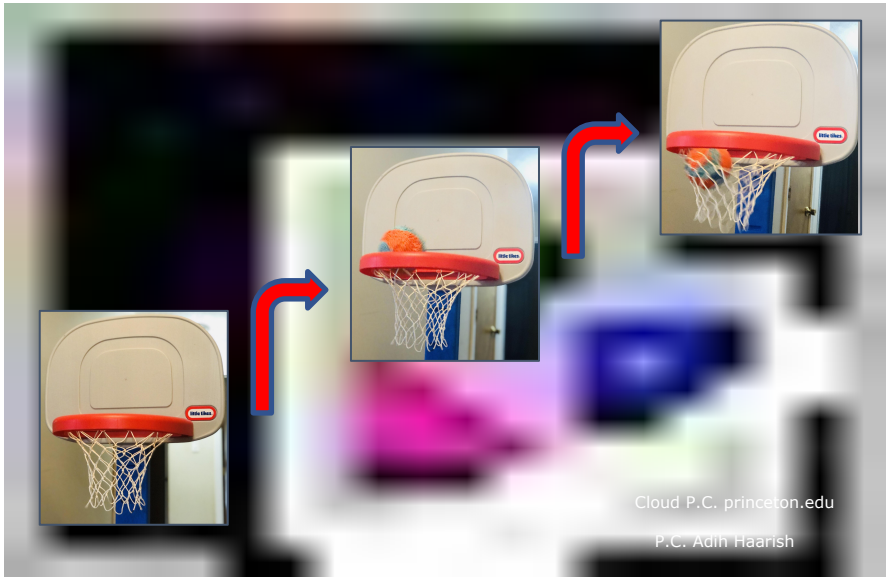
Climate in the cloud: Democratizing access to climate projections

OAR Cloud Tiger Webinar Series

V.Balaji, **Aparna Radhakrishnan**

Sharing our (ongoing) journey to the cloud.

- Setting up cloud storage
- Accessing cloud storage
  - Preparation for ESGF data publication in the cloud
  - Resources to foster analysis in the cloud
- Monitoring and cost exploration
- Acknowledgments



Setting up Cloud Storage

- Storage service used: Amazon Simple Storage Service (Amazon S3)
- Automatic cost savings with Intelligent-tiering enabled
  - Optimized for frequent access
  - Lower cost tier optimized for infrequent access
- Security policies: S3 bucket policies, object access and bucket access control lists (ACLs), IAM (Identify and Access Management) roles.
- Objects organized compliant to CMIP6 Data Reference Syntax (DRS)

e.g. /CMIP6/CMIP/NOAA-GFDL/GFDL-CM4/1pctCO2/r1i1p1f1/Amon/cct/gr1/v20180701/

Data volume: 398TB and counting

CMIP6 NetCDF model output from the ESGF public archives

Ack: CMIP6 data producers and coordinators worldwide

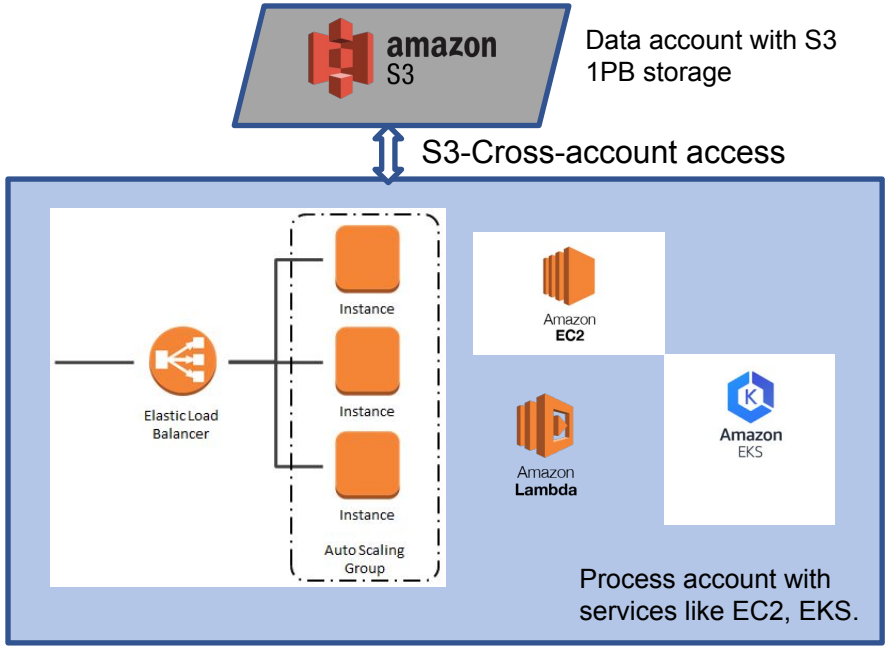
Accessing cloud storage (for development)

- We have 2 AWS awards/accounts from the ASDI initiative.
  - Data account
  - Process account

  We need to enable cross-account access for S3 operations.

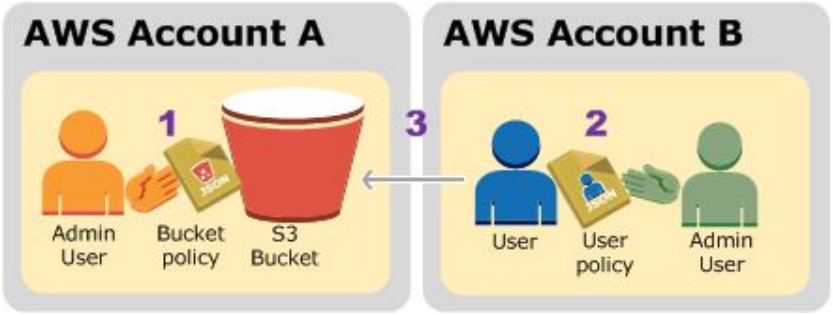
Few available options:

1. AWS automatic replication to the new account
2. Role switching: allows you to assume the role and permissions of another account
3. Cross-account permission with resource based policies and IAM policies



Cross Account Permissions

- We adopted the use of granting cross account permissions updating bucket policies and IAM role policy.
- This gives authorized users in the second account the credentials to perform select S3 operations in the data account with the same permissions as the original account



Ref
https://docs.aws.amazon.com/AmazonS3/latest/dev/example-walkt
roughs-managing-access-example2.html

Example: aws s3 ls s3://esgf-world/CMIP6/ --profile processaccount

```
PRE C4MIP/
PRE CDRMIP/
PRE CFMIP/
PRE CMIP/
PRE DAMIP/
PRE FAFMIP/
PRE GMMIP/
PRE HighResMIP/
PRE LUMIP/
PRE OMIP/
```

Example that uses **AWS CLI commands** to list, copy data to the data account from the process account profile. AWS CLI was used for data transfer operations in the first phase of the project

Accessing cloud storage (for development)

Use-case: The existing tools such as the **ESGF software stack Data publication project** in AWS requires interaction with the Amazon S3 bucket to perform R/W operation

Our team explored options to mount S3 as a local drive

s3fs

A FUSE filesystem application backed by AWS.

pros: easy setup

cons: Latency when performing IO centric operations and metadata operations such as listing/searching. TBD: cost-efficiency.

goofys:

A high-performance, POSIX-ish Amazon S3 file system written in Go

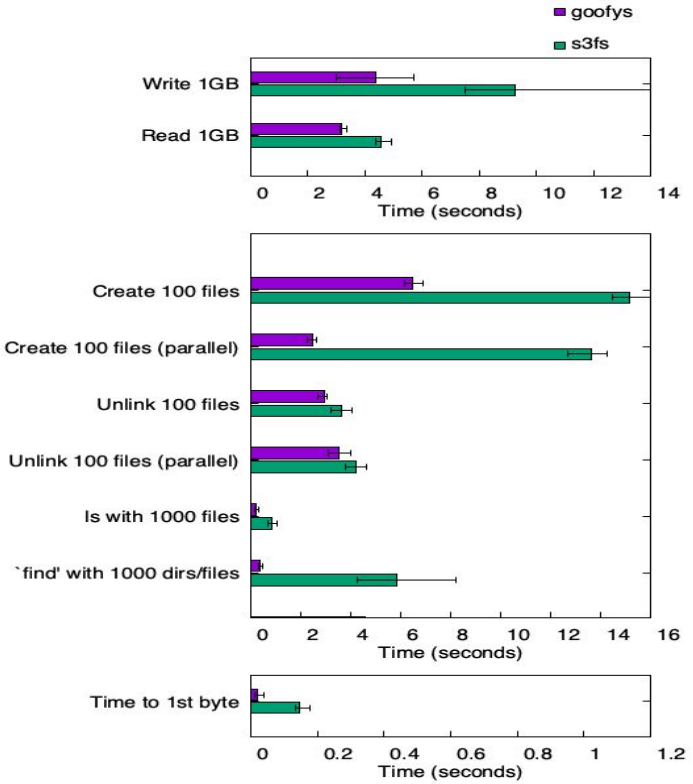
pros: easy setup, in-memory cache

cons: TBD

~~**Amazon Lustre FSx:** Amazon High-performance file system.~~

pros: High Performance

cons: Redundant copy of S3, NOT cost efficient for our use-case



Benchmarking results from https://github.com/kahing/goofys

Building blocks for analysis: The GIST

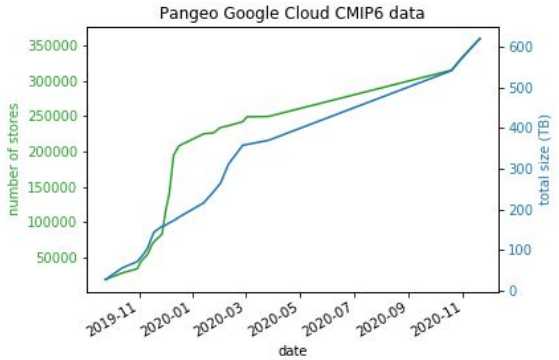
- Gentle Learning curve
- Interoperability
- Scalability Shareability
- Traceability

Community-driven development collaborations

Community-driven development

A community of people working collaboratively to develop software and infrastructure to enable Big Data geoscience research.

Mission



Naomi Henderson et al, AGU 2020

To cultivate an ecosystem in which the next generation of open-source analysis tools for ocean, atmosphere and climate science 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 outside of the geoscience community.

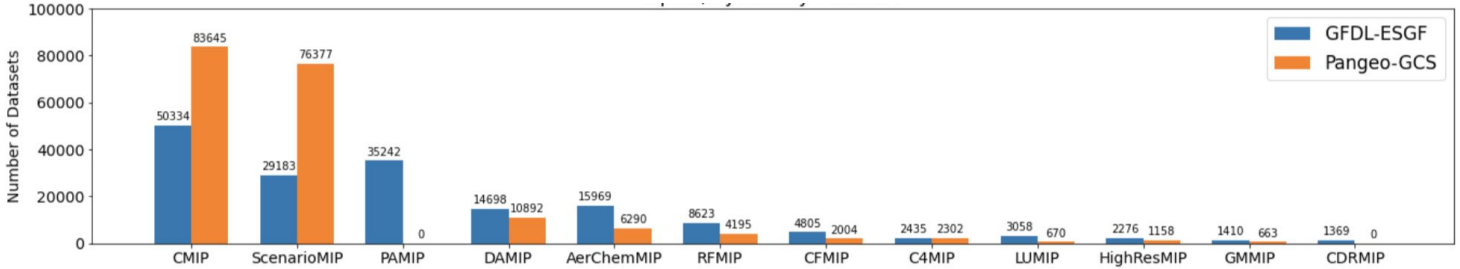


Fig courtesy: Naomi Henderson et al, October 2020.

Ack: Ryan Abernathey and Pangeo team. https://pangeo.io/about.html#about-pangeo

Xarray

- A high-level API for loading, transforming, and performing calculations on multi-dimensional arrays.
- Built leveraging NumPy and pandas API
- Code it like you say it (Easy to get started!)
  - E.g. ds.sel(time='2000-01') ds['thetao'].sel(z_l=2.5).mean(dim='time')
- Incentive, motivation to adhere to CF conventions.
  - Leverages the use of CF metadata Conventions
- Simple gateway to exploring different data formats and input sources.
  - E.g. NetCDF, OPeNDAP, Google cloud data store, Zarr,……
- xarray's data structures can be backed by dask

Gentle learning curve

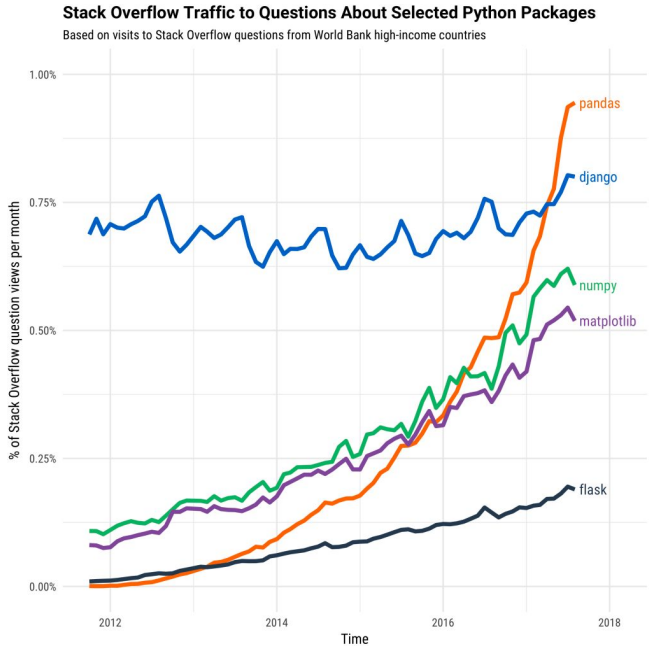
xarray: originally developed by S. Hoyer.

https://github.com/pydata/xarray

DASK

- Fits into a cohesive ecosystem
- Multi-DASKS: Scales up (distributed clusters) and down (single-machine scheduler)
- Less coding intervention to make (many) script Dask-able.
- Provides effective triaging and monitoring system (Dask dashboard),..

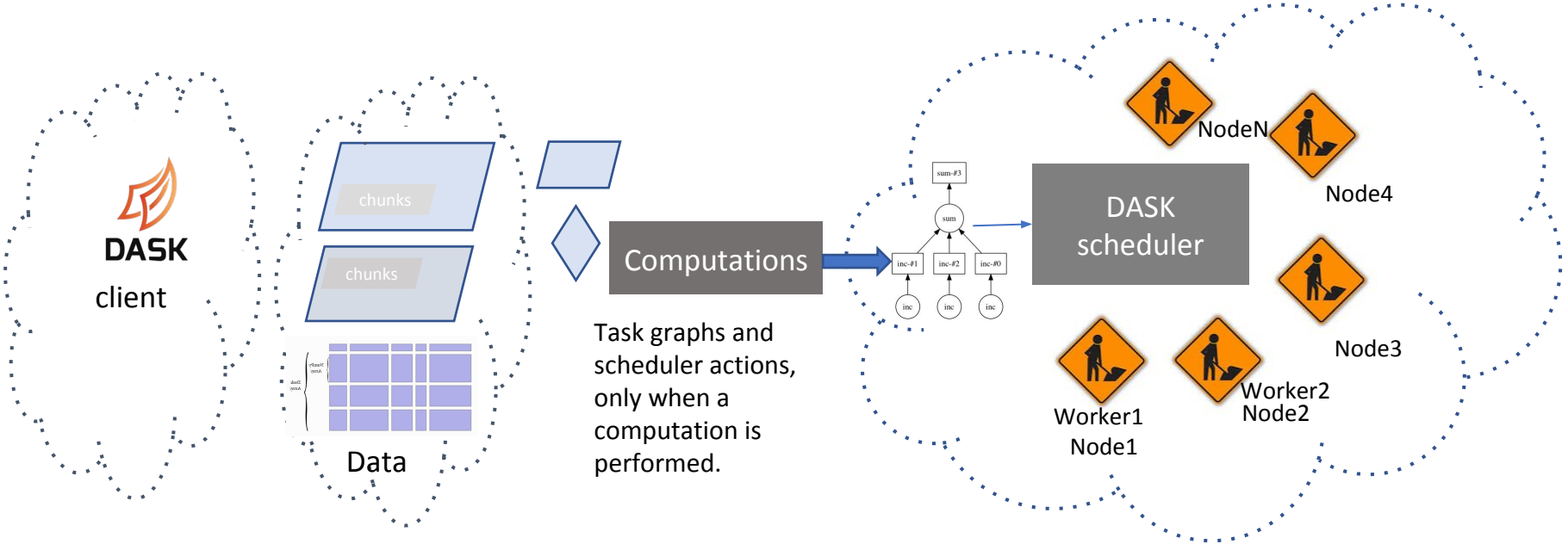
Image credit to Stack Overflow blogposts and R. Abernathey



Dask, originally developed by Matthew Rocklin
https://dask.org/

Scalable
Inter-operable
Gentle learning curve

Chain of actions in (lazy) Dask: A bird's-eye view



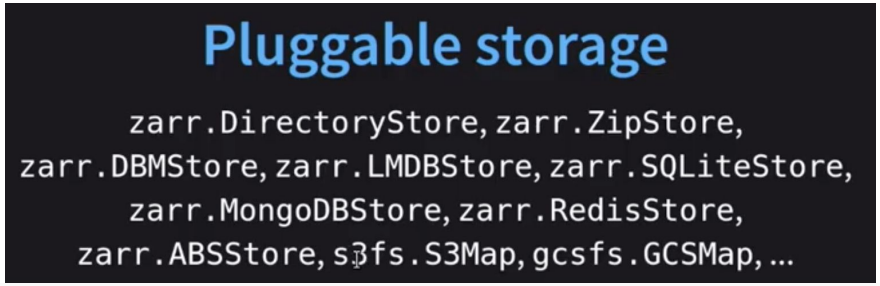
Task graphs and scheduler actions, only when a computation is performed.

Ability to use single node (all cores) or multiple nodes in a cluster

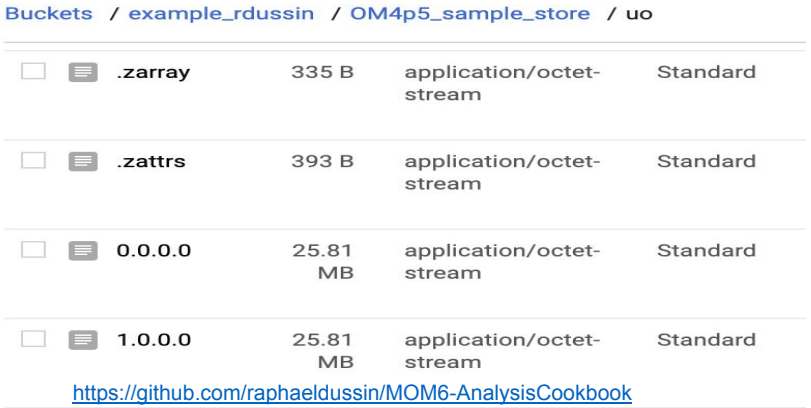
Ability to seamlessly operate in the cloud.

Zarr

- A storage format optimized for high throughput distributed reads on multi-dimensional arrays.
- Zarr works well on both traditional filesystem storage and on Cloud Object Storage.



Zarr Scalable Storage of Tensor Data for Use in Parallel and Distributed Computing, SciPy 2019, A. Miles



https://github.com/raphaeldussin/MOM6-AnalysisCookbook

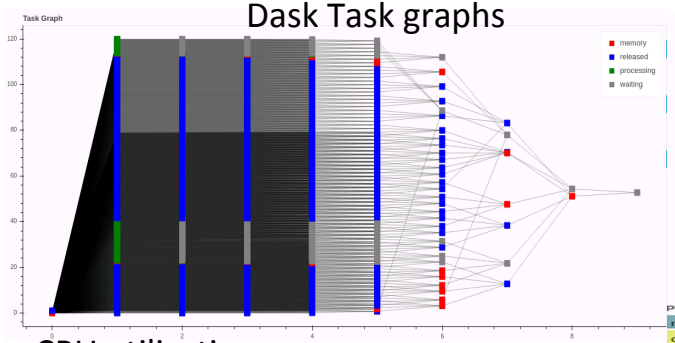
Examples: Dask and xarray

Use dask.distributed task scheduler and launch DASK using SLURMcluster

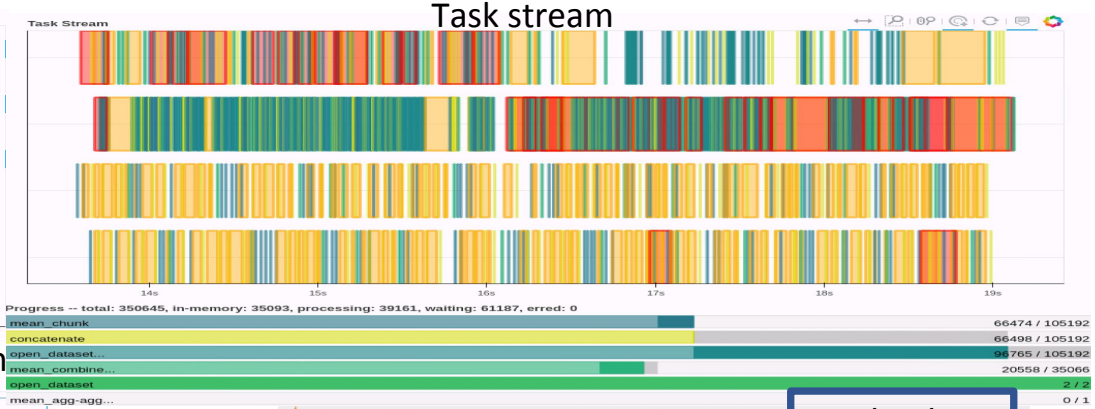
[3]:
```
from dask.distributed import Client

#Instantiate Dask client
if (clusterType == "local"):
    from dask.distributed import LocalCluster
    cluster = LocalCluster(dashboard_address=dashPort,local_directory=localdir)
else:
    from dask_jobqueue import SLURMCluster
    scheduler_options = {}
    scheduler_options["dashboard_address"] = dashPort
    cluster = SLURMCluster(queue='batch',memory=mem,project='gfdl_f',cores=numCores,walltime='2:60:00',
                           scheduler_options=scheduler_options,log_directory=logdir,
                           local_directory=(os.getenv('TMPDIR')))

cluster.scale(numWorkers)
client = Client(cluster)
client
```

Dask dashboard

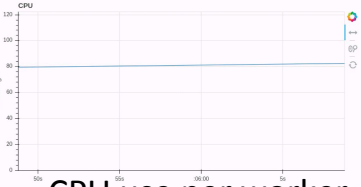
Dask Task graphs



Task stream

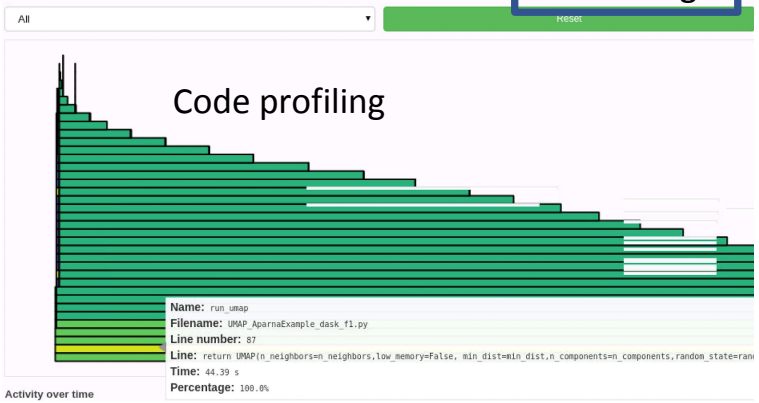


CPU utilization



Memory utilization

Status Workers Tasks System Profile Graph Info → Worker logs

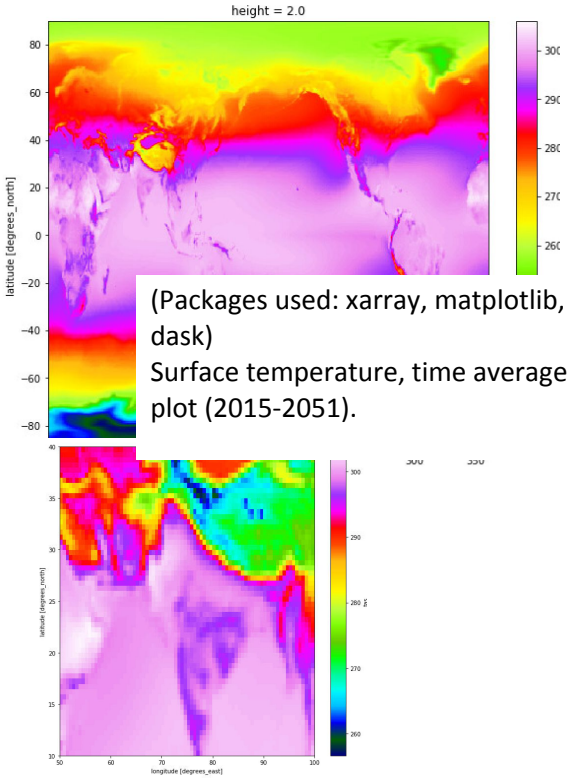


CPU use per worker

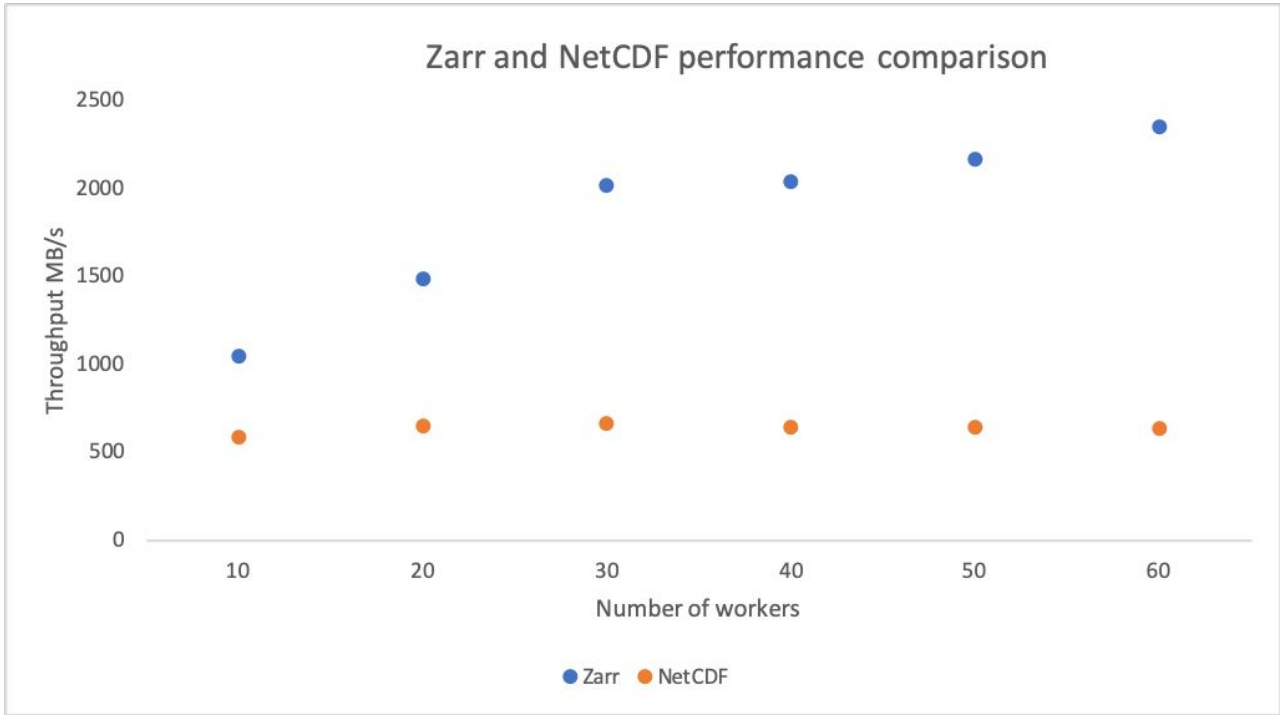
1319.4 %

name	address	nthre	cpu	mem	mem	mem	num_	read_bytes	write_bytes
Total (8)		8	1319.4 %	25 G	504 (	39.0	248	689 KiB	224 KiB
0	inproc://140.208.147.176/35178/4	1	161.4 %	3 GiB	63 G	4.9 %	31	6 KiB	28 KiB
1	inproc://140.208.147.176/35178/5	1	161.3 %	3 GiB	63 G	4.9 %	31	6 KiB	28 KiB
2	inproc://140.208.147.176/35178/6	1	161.1 %	3 GiB	63 G	4.9 %	31	6 KiB	28 KiB
3	inproc://140.208.147.176/35178/3	1	161.2 %	3 GiB	63 G	4.9 %	31	6 KiB	28 KiB
4	inproc://140.208.147.176/35178/13	1	170.1 %	3 GiB	63 G	4.9 %	31	219 KiB	28 KiB
5	inproc://140.208.147.176/35178/14	1	169.3 %	3 GiB	63 G	4.9 %	31	218 KiB	28 KiB
6	inproc://140.208.147.176/35178/11	1	164.0 %	3 GiB	63 G	4.9 %	31	6 KiB	28 KiB
7	inproc://140.208.147.176/35178/12	1	171.0 %	3 GiB	63 G	4.9 %	31	220 KiB	28 KiB

Preliminary Benchmarking Results



(Packages used: xarray, matplotlib, dask)
Surface temperature, time average plot (2015-2051).

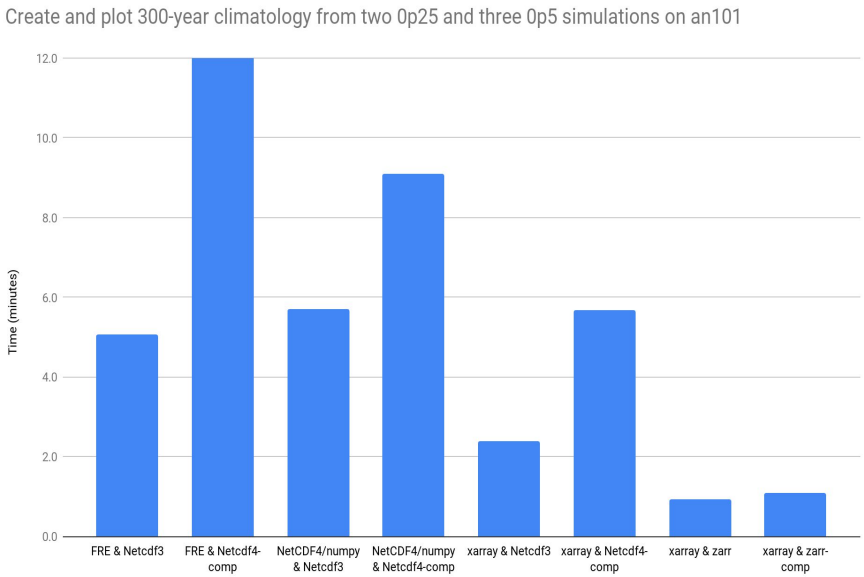


Ack: Zhao, Ming; et al.
https://doi.org/10.22033/ESGF/CMIP6.2262
2015-2051

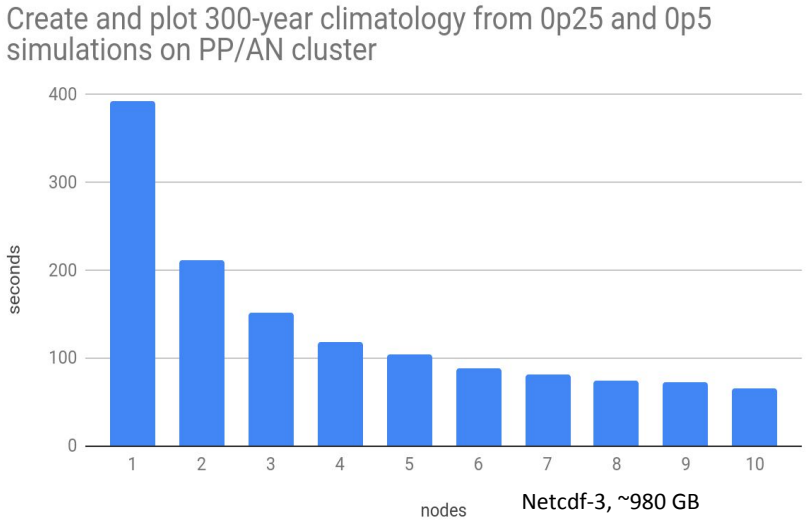
Ref. https://github.com/aradhakrishnanGFDL/enes2020

Preliminary Benchmarking Results (contd..)

2D tos SST notebook timing varying method & data format



Resource scaling test



zarr was faster than NetCDF

compressed zarr was *much* faster than compressed NetCDF

Perry et al, 2019

Examples: Dask and scikit-learn

Before DASK

```
X, y = make_blobs(n_samples = 150000, n_features = 2, centers = 2, cluster_std = 1.9)
model = DBSCAN(eps = 0.5, min_samples = 20)
%time model.fit(X)
```

```
CPU times: user 6 s, sys: 638 ms, total: 6.64 s
Wall time: 6.61 s
```

AFTER DASK

```
cluster.adapt(minimum=1,maximum=4)
```

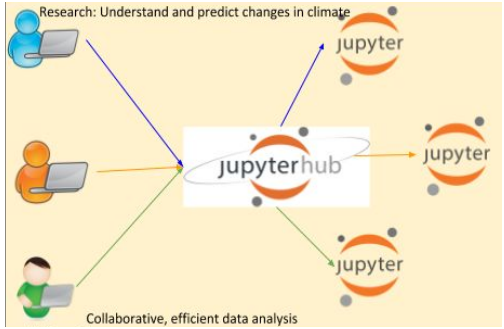
##Test with DASK

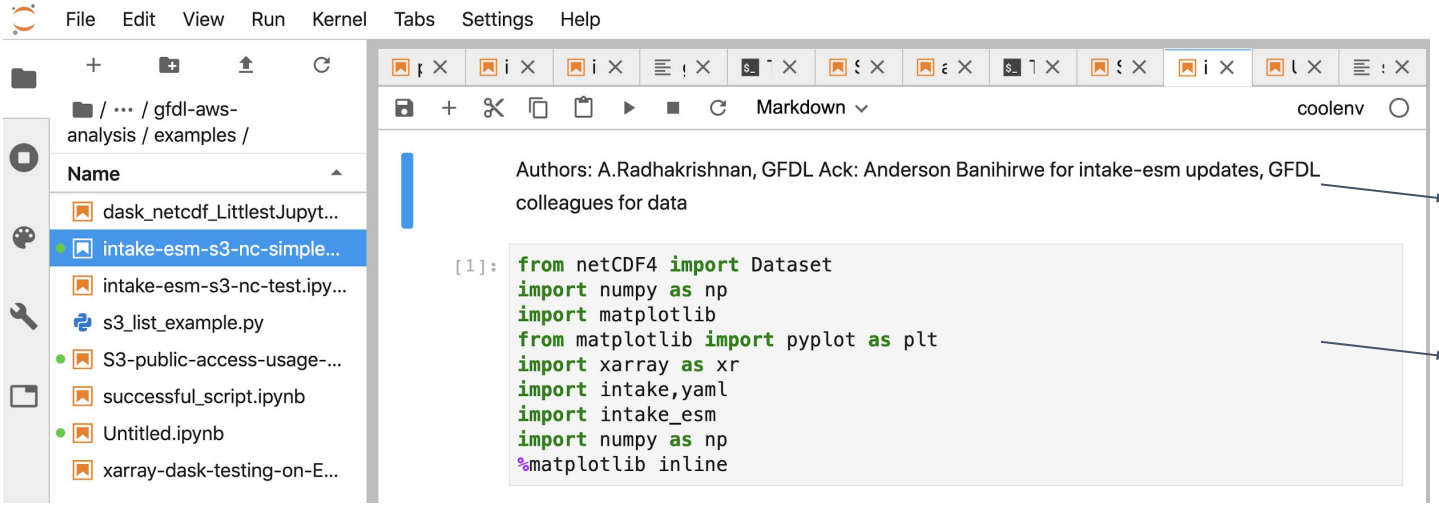
```
from joblib  import parallel_backend,parallel
X, y = make_blobs(n_samples = 150000, n_features = 2, centers = 2, cluster_std = 1.9)
model = DBSCAN(eps = 0.5, min_samples = 20,n_jobs=-1)
with parallel_backend('dask'):
    %time model.fit(X)
```

```
CPU times: user 8.73 s, sys: 563 ms, total: 9.29 s
Wall time: 3.92 s
```

JupyterHub

- The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code and documentations.
- JupyterHub brings the power of Jupyter notebooks to groups of users.
- JupyterHub gives users access to computational environments and resources without burdening the users with installation and maintenance tasks.
- Ideal for "Bringing desktop to data"





Screenshot of JupyterHub deployed in our AWS

Ref. https://jupyter.org/hub

Amazon Elastic Compute Cloud - EC2

Amazon EC2 provides scalable compute capacity on the cloud

One EC2 instance can give you the power to launch a simple JupyterHub

With the combined power of many EC2 instances you can speed workflow

Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)
EC2 instance naming convention: <family><generation>.<size>

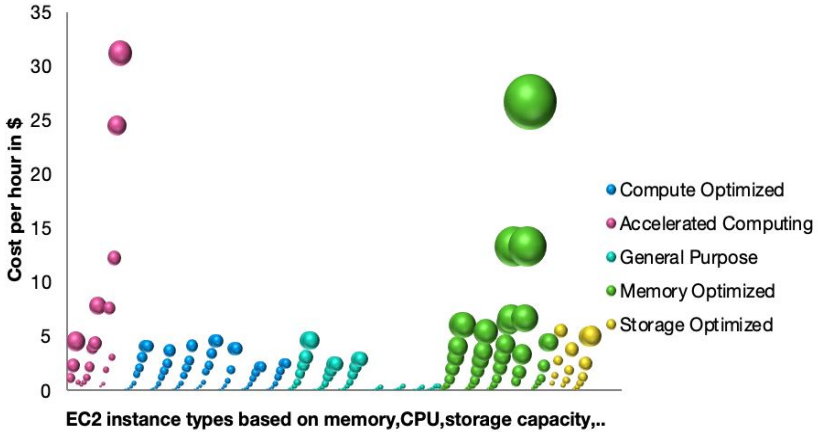
	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	Network Performance
☐	t2	t2.nano	1	0.5	EBS only	Low to Moderate
☑	t2	t2.micro Free tier eligible	1	1	EBS only	Low to Moderate
☐	t2	t2.small	1	2	EBS only	Low to Moderate
☐	t2	t2.medium	2	4	EBS only	Low to Moderate
☐	t2	t2.large	2	8	EBS only	Low to Moderate
☐	t2	t2.xlarge	4	16	EBS only	Moderate
☐	t2	t2.2xlarge	8	32	EBS only	Moderate

Cost and usage of EC2

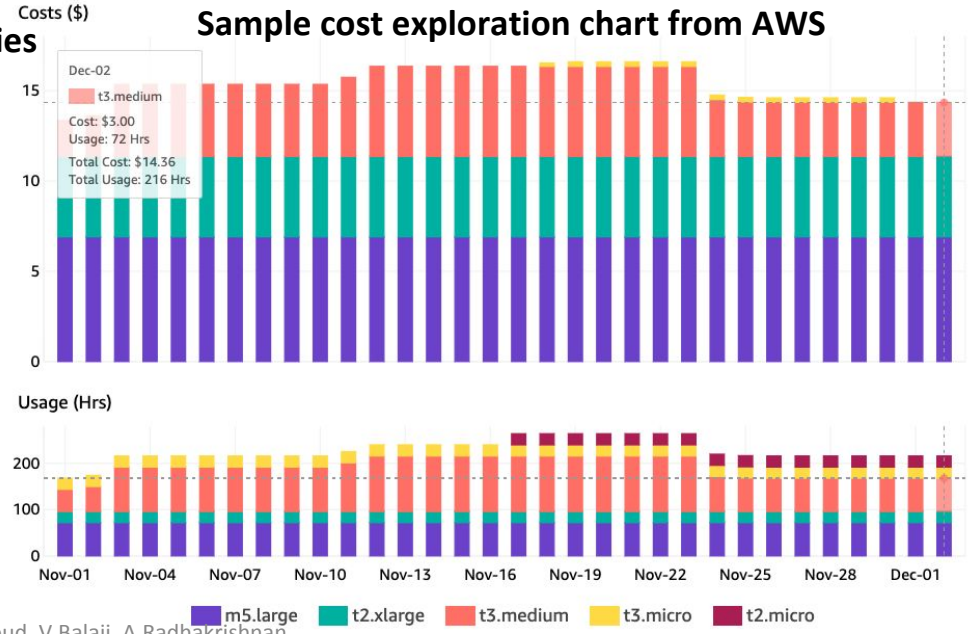
Choice of EC2 instances may depend on the nature of your application requirements, region specification and cost factors.

Scope for optimization: Auto-scaling, scheduled time for nodes.

Cost comparison of different types of EC2 instance families



Sample cost exploration chart from AWS

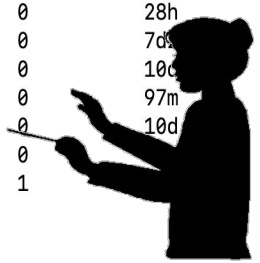


Disclaimer: Cost information on this slide may vary based on region, time, etc. The values presented here are only approximate costs. For accurate info, please refer to your AWS account billing dashboard.

Kubernetes and JupyterHub

- Kubernetes is a container orchestration system designed to facilitate containers and virtual machines for an application
- Kubernetes cluster in conjunction with JupyterHub can be configured to use multiple EC2 instances, which give a much larger compute power than our first attempt

```
(base) aos-pqlvdl:gfdl-aws-analysis ar46$ kubectl --namespace jhub get pods
NAME                              READY   STATUS    RESTARTS   AGE
continuous-image-puller-58x82     1/1     Running   0          10d
continuous-image-puller-6vwg4     1/1     Running   0          10d
continuous-image-puller-8h6gm     1/1     Running   0          28h
continuous-image-puller-j4qhl     1/1     Running   0          10d
continuous-image-puller-m2r2x     1/1     Running   0          27h
continuous-image-puller-m9s65     1/1     Running   0          28h
continuous-image-puller-mvjft     1/1     Running   0          7d23h
continuous-image-puller-r97mw     1/1     Running   0          10d
continuous-image-puller-rwlb2     1/1     Running   0          28h
continuous-image-puller-xh9zl     1/1     Running   0          28h
continuous-image-puller-xvsq5     1/1     Running   0          7d
hub-dfbc895cf-xrp5z               1/1     Running   0          10
jupyter-jupyter-5fadmin           1/1     Running   0          97m
proxy-75d76c4c64-bhs94            1/1     Running   0          10d
user-scheduler-5d97fdd64f-dxd6n   1/1     Running   0
user-scheduler-5d97fdd64f-sm2rf   1/1     Running   1
```



Amazon Elastic Kubernetes Service - EKS

Amazon has a built in way to run Kubernetes services called **EKS**

- EKS can utilize multiple EC2 instances to create a cluster and provide compute power to multiple Kubernetes pods
- minikube (sets up Kubernetes cluster locally) could be employed for testing (we used it to test the ESGF software stack)
- EKS installation is based on this tutorial from AWS.
- Auto-scaling options can be enabled as needed.

Installing JupyterHub in EKS

Setup Kubernetes with required security permissions, e.g AWS EKS

↓

Setup helm package manager

↓

Install JupyterHub using JupyterHub Helm Charts

↓

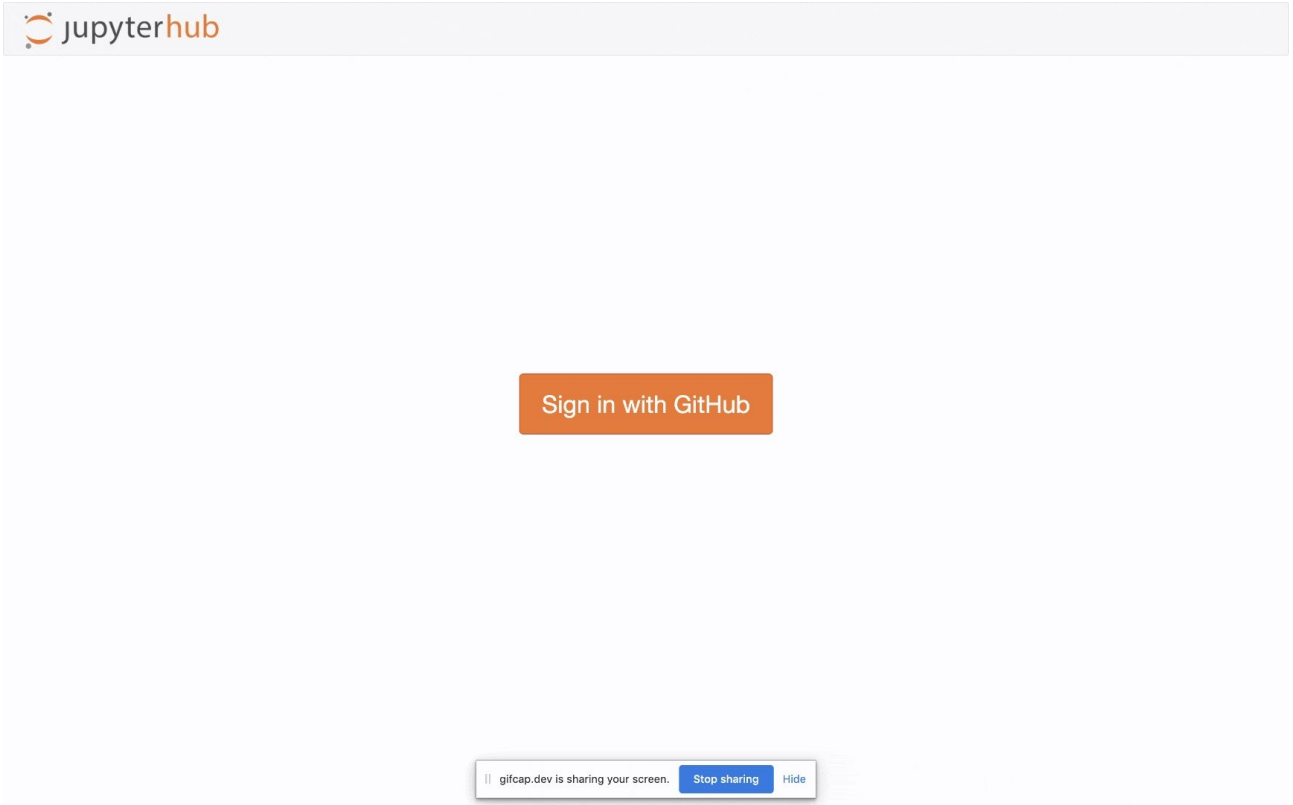
Update configurations to customize authentications, dask worker environments, etc

↓

Run a helm upgrade to update

```
(base) aos-pqlvdl:gfdl-aws-analysis ar46$ helm list
NAME            REVISION        UPDATED                         STATUS          CHART                   APP VERSION     NAMESPACE
appt-release    1               Tue Aug 18 14:23:55 2020        DEPLOYED        dask-4.3.1              2.23.0          default
jhub            1               Sun Aug  9 00:43:22 2020        DEPLOYED        jupyterhub-0.9.0        1.1.0           jhub
(base) aos-pqlvdl:gfdl-aws-analysis ar46$
```

JupyterHub in EKS



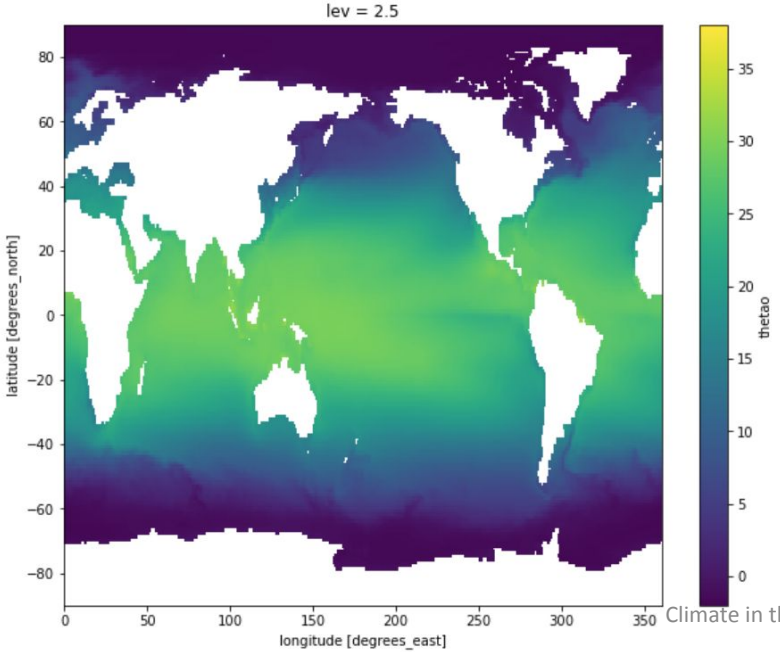
Performance comparison in the cloud/AWS

1 EC2 instance running dask local cluster in Littlest JupyterHub

```
CPU times: user 7min 5s, sys: 2min 41s, total: 9min 46s
Wall time: 24min 49s
```

```
/opt/conda/lib/python3.7/site-packages/dask/array/numpy_compat.py:41: Runtim
ue_divide
  x = np.divide(x1, x2, out)
```

```
: <matplotlib.collections.QuadMesh at 0x7eff18f5cdd0>
```

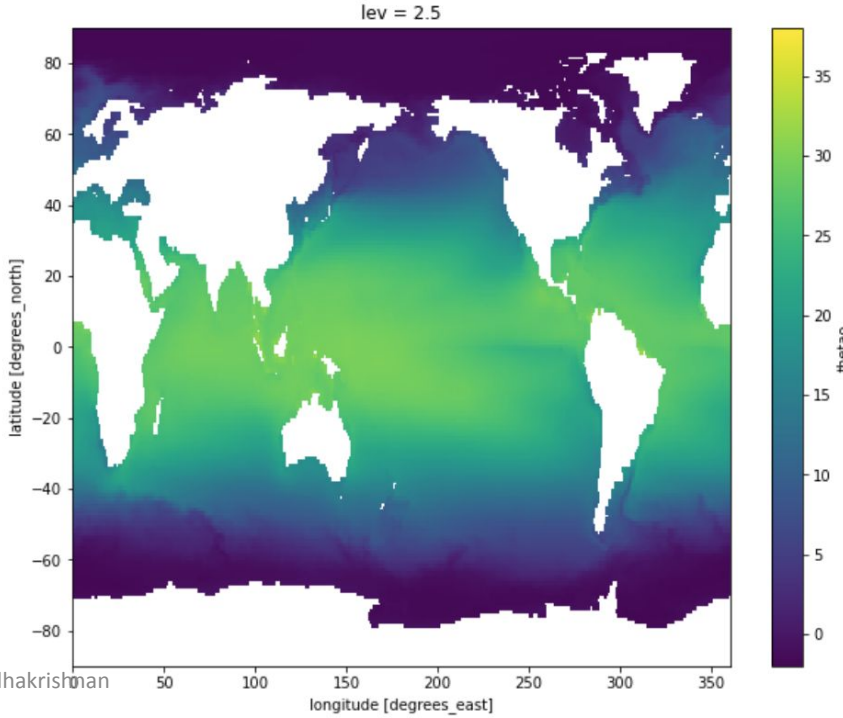


3 worker nodes/EC2 running Dask distributed on JupyterHub in EKS

Data citation: Adcroft, Alistair et al, (2018). NOAA-GFDL GFDL-CM4 model output prepared for CMIP6 OMIP. Version 20180701[1].Earth System Grid Federation. https://doi.org/10.22033/ESGF/CMIP6.1403

```
CPU times: user 1.77 s, sys: 53.2 ms, total: 1.82 s
Wall time: 5min 46s
```

```
Out[7]: <matplotlib.collections.QuadMesh at 0x7fd5076a5390>
```



Useful resources to access S3 using Python

1. Quick-start-guide to use our example Jupyter notebooks.
2. Example script that shows how to list contents of an S3 bucket using python3, boto3.
3. Example to show how to open a dataset in S3 using s3fs in Python.

```
import s3fs
import xarray as xr
fs_s3 = s3fs.S3FileSystem(anon=True)
s3path =
's3://<bucketName>/CMIP6/OMIP/NOAA-GFDL/GFDL-CM4/omip1/r1i1p1f1/Omon/thetao/gr/v20180701/thetao_Omon_GFDL-CM4_omip1_r1i1p1f1_gr_170801-172712.nc'
remote_file_obj = fs_s3.open(s3path, mode='rb')

xarray_dataset_object = xr.open_dataset(remote_file_obj,
chunks={'time': 1})
```

xarray.Dataset

▸ Dimensions: (bnds: 2, **lat**: 180, **lev**: 35, **lon**: 360, **time**: 3600)

▾ Coordinates:

lon	(lon)	float64	0.5 1.5 2.5 ... 357.5 358.5 359.5
lat	(lat)	float64	-89.5 -88.5 -87.5 ... 88.5 89.5
lev	(lev)	float64	2.5 10.0 20.0 ... 6e+03 6.5e+03
time	(time)	object	1708-01-16 12:00:00 ... 2007-12-...

▾ Data variables:

lat_bnds	(time, lat, bnds)	float64	dask.array<chunksize=(240, 180, 2), meta=np.nd...
lon_bnds	(time, lon, bnds)	float64	dask.array<chunksize=(240, 360, 2), meta=np.nd...
thetao	(time, lev, lat, lon)	float32	dask.array<chunksize=(1, 35, 180, 360), meta=np...
time_bnds	(time, bnds)	object	dask.array<chunksize=(1, 2), meta=np.ndarray>
lev_bnds	(time, lev, bnds)	float64	dask.array<chunksize=(240, 35, 2), meta=np.nda...

▾ Attributes:

title : NOAA GFDL GFDL-CM4 model output prepared for CMIP6 OMIP experiment forced by CORE Inter-Annual Forcing (CIAF) version 2.0 and initialized with observed physical and biogeochemical ocean data

history : File was processed by frametar (GFDL analog of CMOR). TripleID: [experiment_id_TFN7SZc1yU,realiz_id_HX30DNwWjL,run_id_aOuCxFhBdP]

Data exploration

Encourage the use of Data/Metadata **cataloging capability**.

E.g. intake-esm: a data cataloging utility from Pangeo that uses

ESM collection specifications file and is built on top of intake, pandas, and xarray

GFDL's CatalogBuilder- A python API to build custom data inventory catalogs (e.g. intake-esm) from data hosted in the cloud or your local file system.

Ack: A.Radhakrishnan, Rebecca Monge

Useful resources to access S3 using Python

Our S3 data inventory catalog (updated regularly)

Example script that uses the above data catalogs (intake-esm in this example) to fit into our analysis seamlessly.

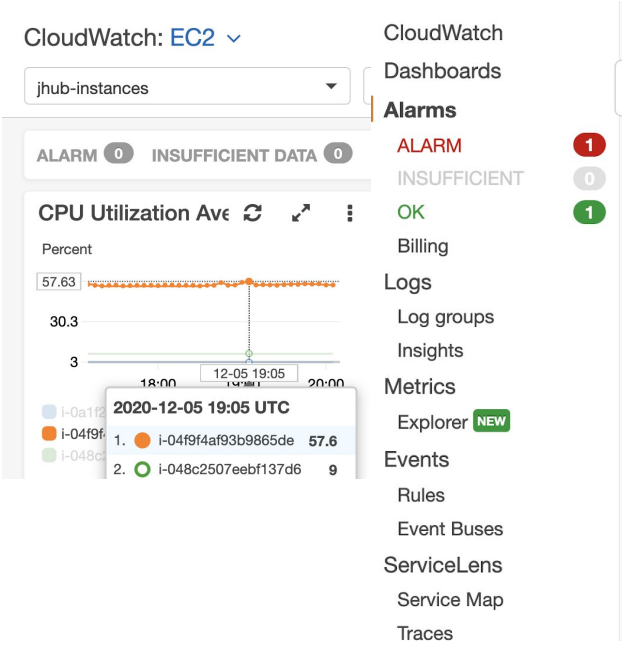
Example notebook to access S3 using Python

This example: Uses intake-esm, Python3, xarray to analyze a sample dataset in S3 bucket

Tools to monitor resources and applications

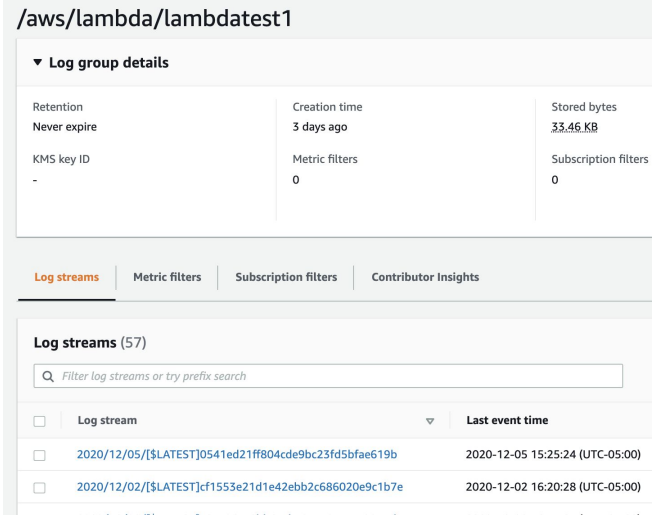
Amazon Cloudwatch

To monitor services and provide a unified view of the operational health of the AWS resources

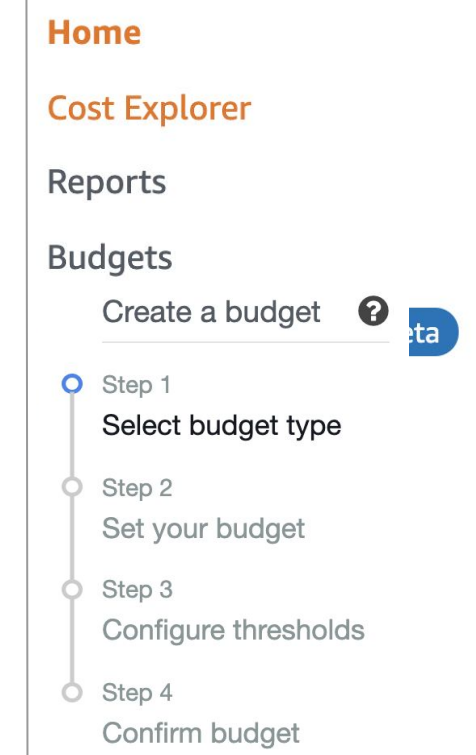


AWS Lambda

serverless compute service: lets you run code without provisioning servers

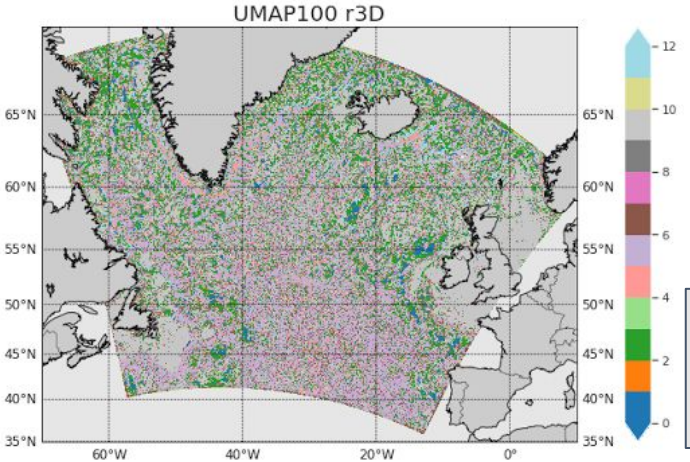


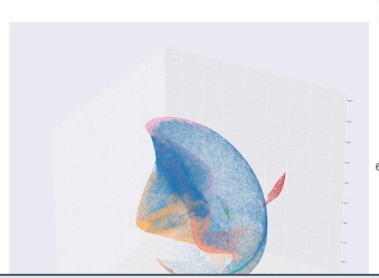
AWS Cost Management

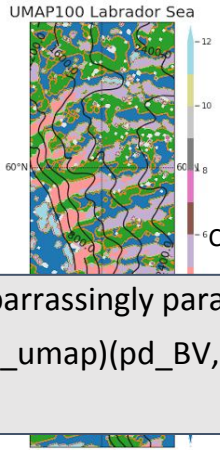


Ongoing/Future work

- Cost optimizations and instance configurations. **e.g** requester pays cost model.
- Further testing and performance tuning
- Robust strategies for configuring Dask array chunking
- Explore use of cloud based analysis in more challenging ML applications
- Complete deployment of Earth System Grid Federation software stack in AWS/EKS.
- Encourage users to use official CMIP6 data citation from further_info_url global attribute of NetCDF data in S3.







Cr: Sonnewald et al, 2020

Lazy-greedy user, Lazy dask, Embarrassingly parallel code-or-not

comps.append(dask.delayed(run_umap)(pd_BV, neighbors, dist,n_comp))

BV budget by Le Corre et al. 2019, using the Regional Oceanic Modelling System (ROMS, Shchepetkin and McWilliams (2009))

Thank you!

Many thanks to GFDL, ESGF, Pangeo community and our AWS colleagues and collaborators.

Additional references

- https://docs.aws.amazon.com/AmazonS3/latest/dev/example-walkthroughs-managing-access-example2.html
- https://jupyter.org/
- https://docs.aws.amazon.com/eks/latest/userguide/getting-started.html
- https://zero-to-jupyterhub.readthedocs.io/en/latest/kubernetes/amazon/step-zero-aws-eks.html
- https://docs.aws.amazon.com/
- https://github.com/aradhakrishnanGFDL/gfdl-aws-analysis/tree/community/examples