Building blocks for exascale computing at GFDL

Aparna Radhakrishnan, V.Balaji, Thomas Jackson, Maike Sonnewald

6th ENES HPC Workshop, May 29, 2020

PRINCETON UNIVERSITY

NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION
NOAA
U.S. DEPARTMENT OF COMMERCE

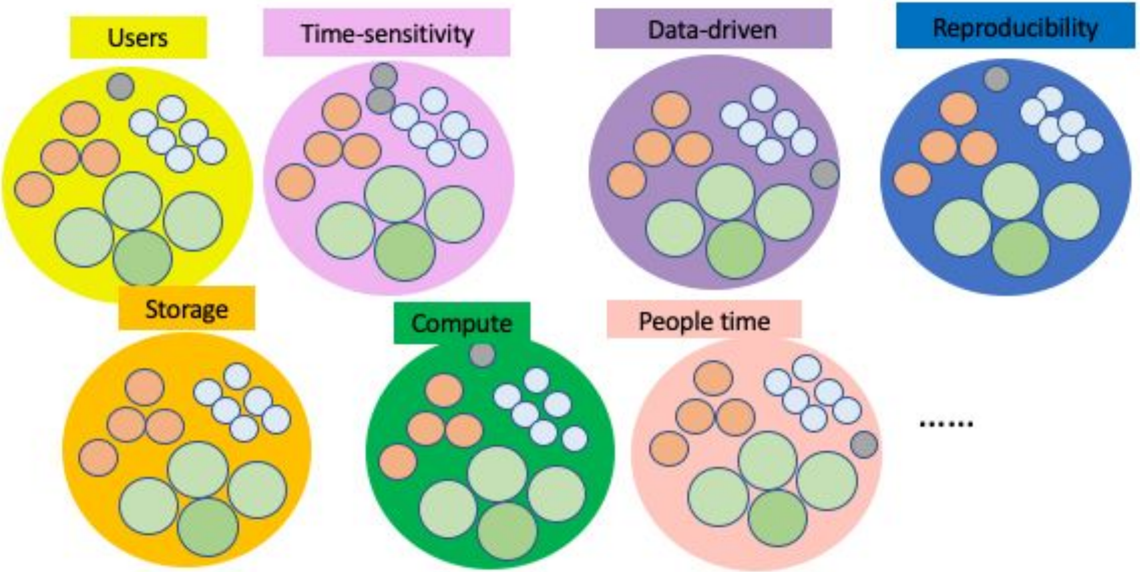
Acknowledgment

Special thanks to..

- Spencer Clark, Raphael Dussin, John Krasting, Vaishali Naik for their valuable input
- Perry Boh for his fantastic previous related work at GFDL
- Ming Zhao for CMIP6 HighResMIP data

and many more.

Factors influencing data analysis



INFRASTRUCTURE

Building blocks for analysis: The GIST


photo credit: HV

- Gentle Learning curve
- Interoperability
- Scalability Shareability
- Traceability

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.

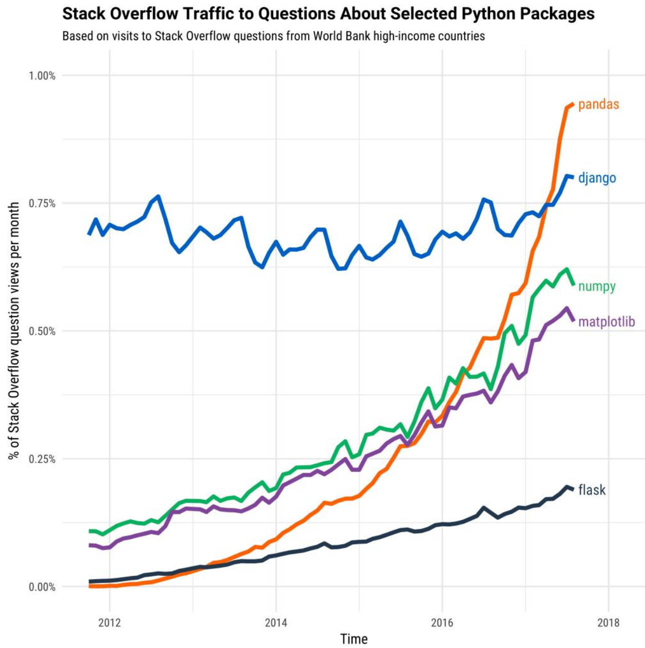
https://github.com/pydata/xarray

DASK

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

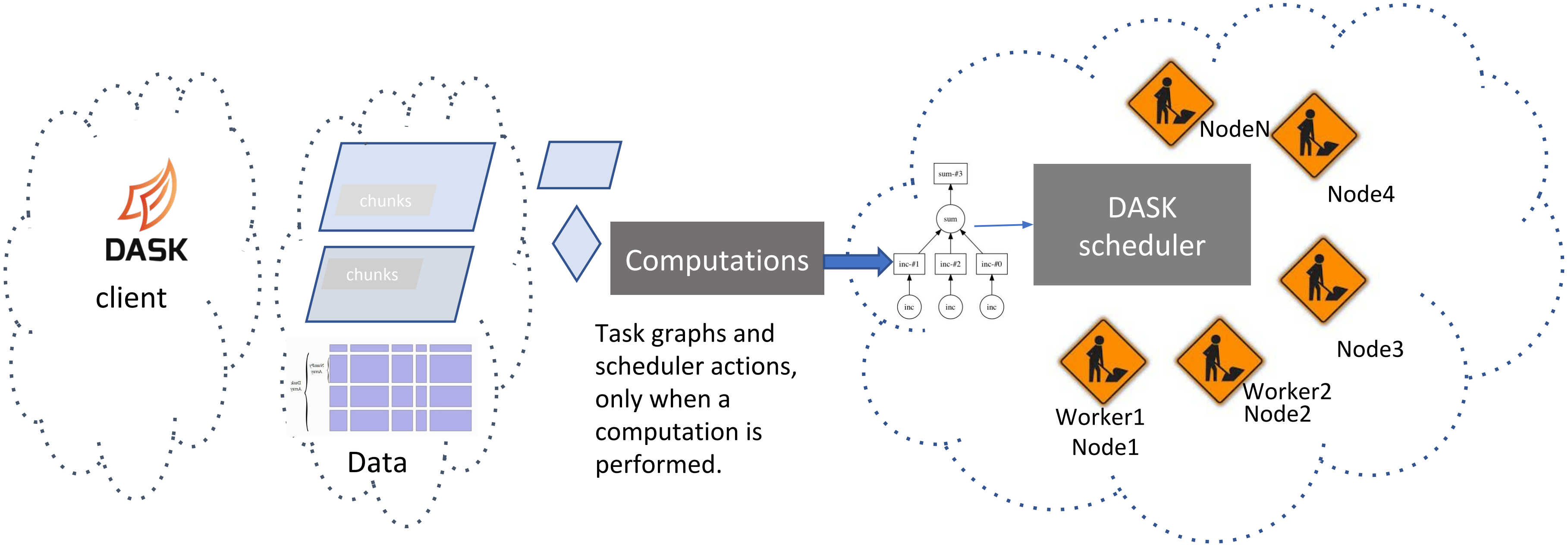


Dask, originally developed by Matthew Rocklin
https://dask.org/

Scalable
Inter-operable
Gentle learning curve

DASK

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



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.

Pluggable storage

zarr.DirectoryStore, zarr.ZipStore,
zarr.DBMStore, zarr.LMDBStore, zarr.SQLiteStore,
zarr.MongoDBStore, zarr.RedisStore,
zarr.ABSStore, s3fs.S3Map, gcsfs.GCSMap, ...

Zarr Scalable Storage of Tensor Data for Use in Parallel and Distributed Computing, SciPy 2019, A. Miles
https://zarr.readthedocs.io/en/stable/

Buckets / example_rdussin / OM4p5_sample_store / uo

.zarray	335 B	application/octet-stream	Standard
.zattrs	393 B	application/octet-stream	Standard
0.0.0.0	25.81 MB	application/octet-stream	Standard
1.0.0.0	25.81 MB	application/octet-stream	Standard

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

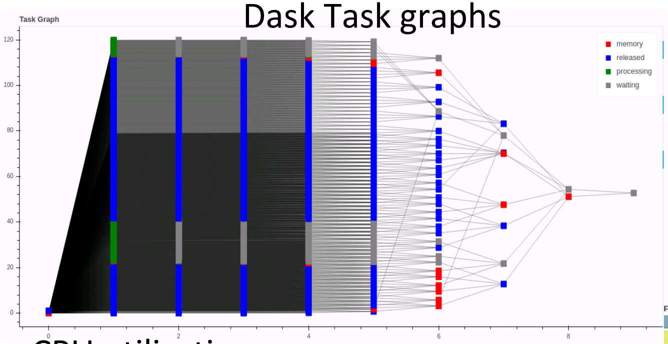
Examples: Dask and xarray

Use dask.distributed task scheduler and launch DASK using SLURMcluster

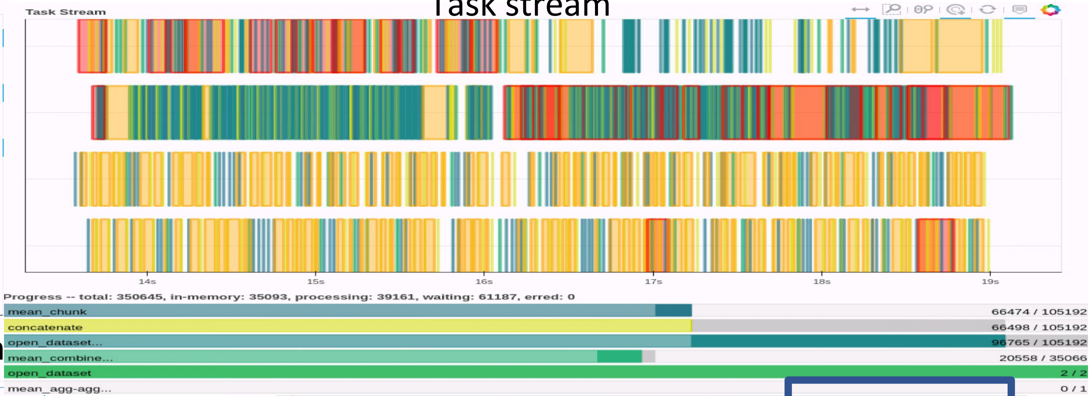
```
[3]: from dask.distributed import Client

#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

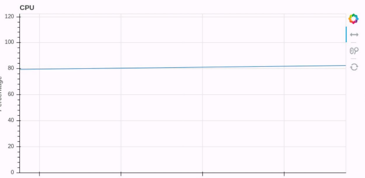
Dask Task graphs



Task stream



CPU utilization



Memory utilization

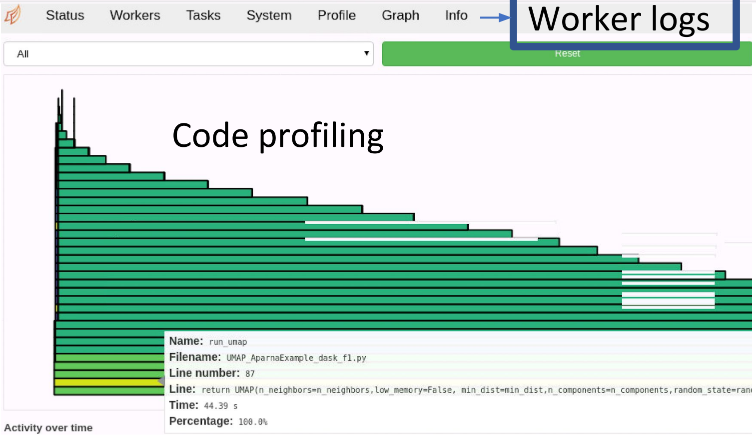
CPU use per worker

1319.4 %

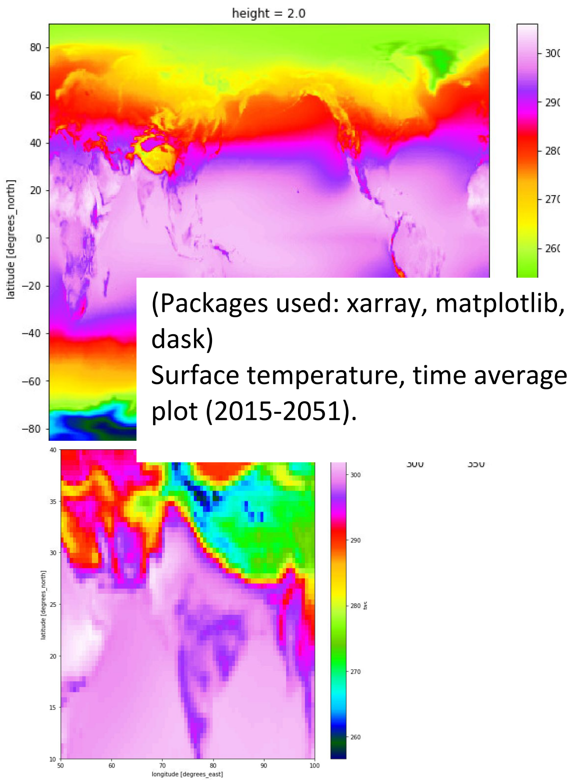
name	address	nthre	cpu	m m	mem	mem	num_	read_bytes	write_bytes
Total (8)		8	1319.4	25 G	504 (	39.0	248	689 KiB	224 KiB
0	inproc://140.208.147.176/35178/4	1	161.4 %	3 GiE	63 G	4.9 %	31	6 KiB	28 KiB
1	inproc://140.208.147.176/35178/5	1	161.3 %	3 GiE	63 G	4.9 %	31	6 KiB	28 KiB
2	inproc://140.208.147.176/35178/6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

Worker logs

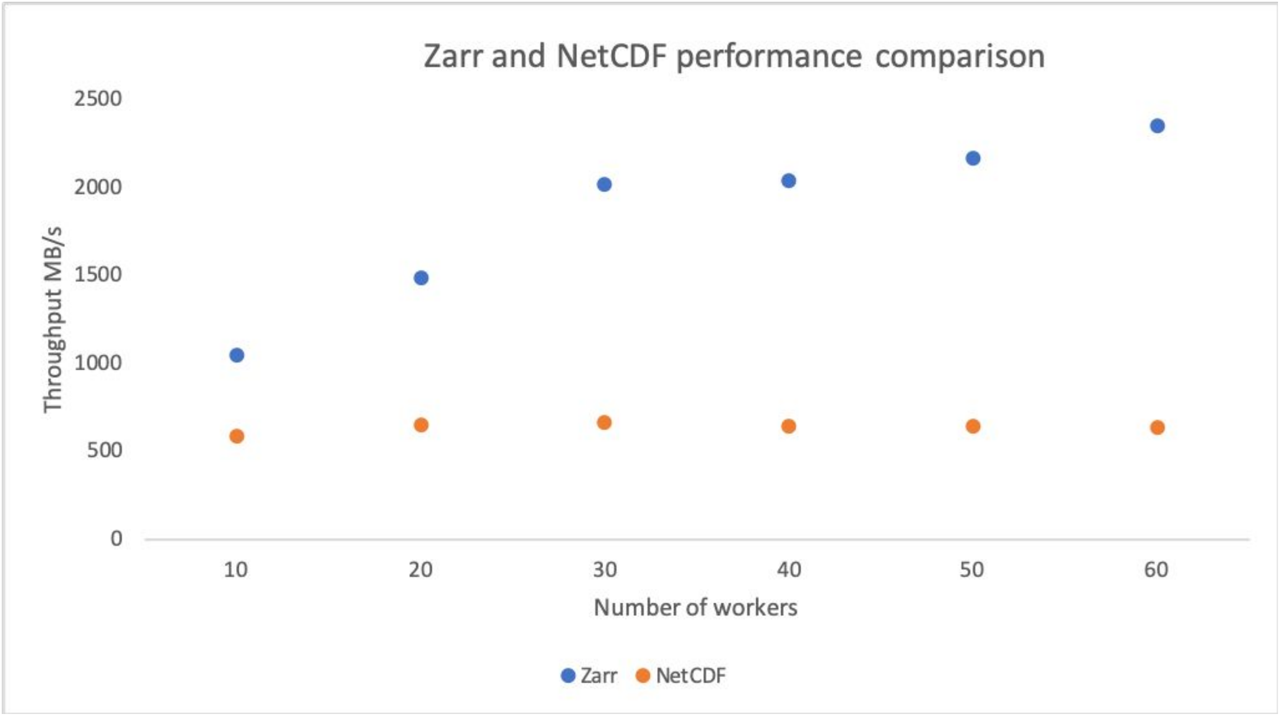
Code profiling



Preliminary Benchmarking Results



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

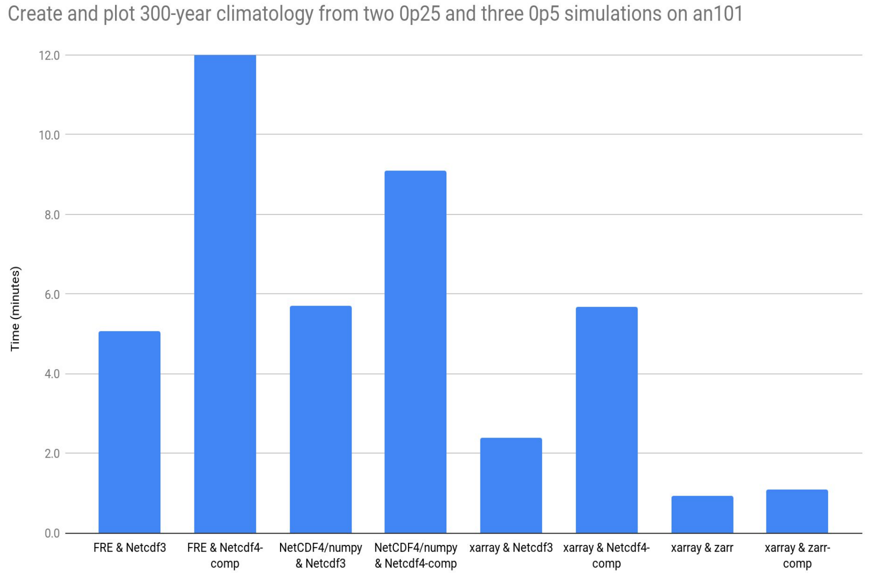


Ack: Zhao, Ming; et al.
https://doi.org/10.22033/ESGF/CMIP6.2262
2015-2051

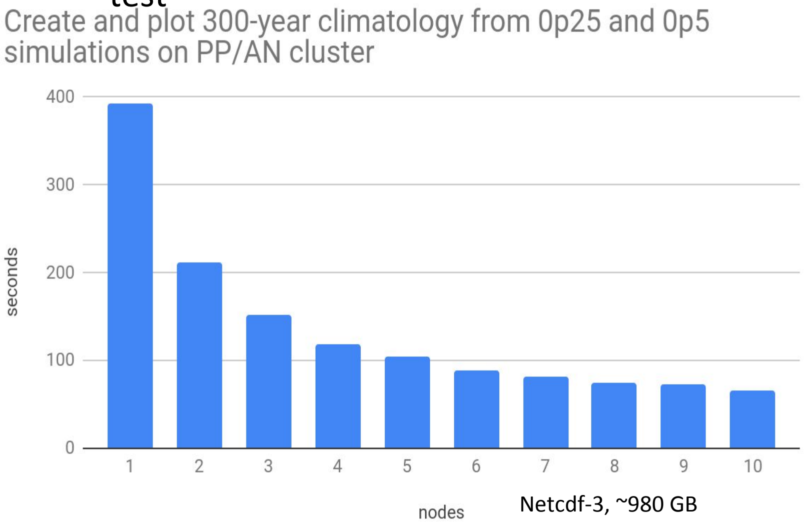
Ref. https://github.com/aradhakrishnanGFDL/enes2020

Preliminary Benchmarking Results (contd..)

2D tos SST notebook timing varying method & data format



Resource scaling test



Netcdf-3, ~980 GB

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
```

Data exploration

GFDL Unified Data Archive

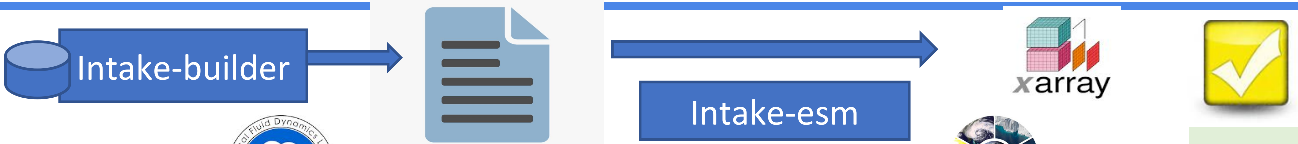
A centralized location at GFDL for data published to the Earth System Grid Federation (ESGF) (i.e. CMIP5, CMIP6) and for other reanalysis datasets.

Ack: K.Rand, GFDL

Cloud optimized containerized workflow for data hosting and computing

Data/Metadata **cataloging capability** from multiple sources

- intake-esm is a data cataloging utility that uses **ESM collection file** and is built on top of intake, pandas, and xarray
- Intake-builder- A python API to build custom intake catalogs from multiple data sources/formats.



Traceability

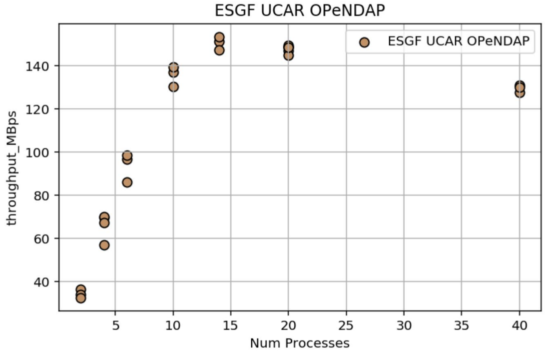
Building blocks for analysis: The GIST

- Gentle Learning curve
- Interoperability
- Scalability Shareability
- Traceability

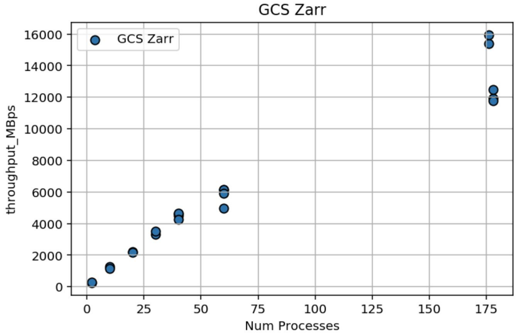
Community-driven development collaborations

Community-driven development

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



Throughout bandwidth of UCAR ESGF OPeNDAP server in MB/s, as a function of the number of parallel processes used. Note that the throughput saturates at around 140 MB/s.



Throughput of reading Zarr data from Google Cloud Storage with a Dask Kubernetes cluster, as a function of the number of processes in the cluster.

PANGEO

Mission

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.

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

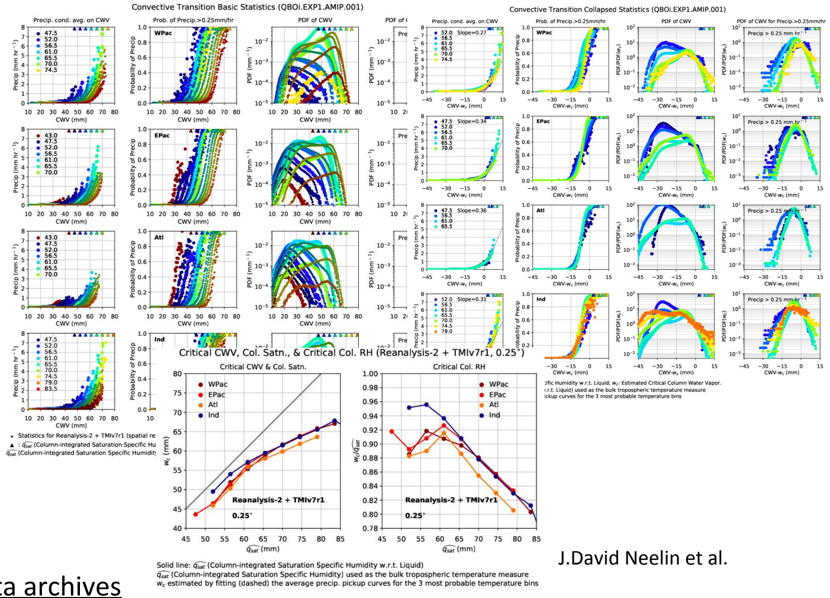
Model Development TaskForce (MDTF)

The MDTF project combines community expertise and Federal model development through process-oriented diagnostics that span weather to climate timescales

Write once, Run often: MDTF provides a unified framework to move from inspiration driven research to industrial strength research.

Improve the scientists' data analysis experience by automating the data exploration and analysis (several supported languages) management phases.

Jackson T, Krasting J, Dong W, MDTF,2020

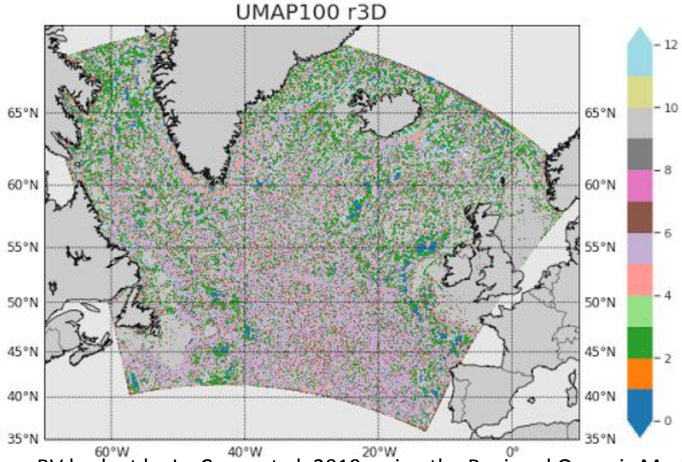


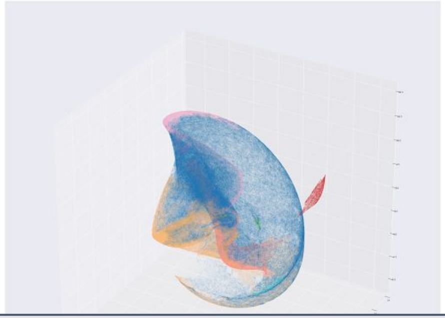
J.David Neelin et al.

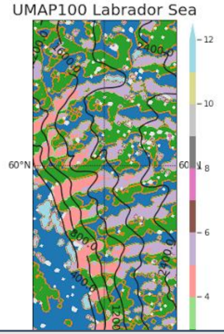
Ref.
https://github.com/NOAA-GFDL/MDTF-diagnostics
V.Balaji et al, Deploying user-developed scientific analyses on federated data archives
A.Radhakrishnan et al, From inspiration-driven research to industrial-strength research, 2014

Future work: Challenges (continued..)

- More testing and performance tuning
- Robust strategies for configuring Dask array chunking
- Explore use of Dask in more challenging ML applications





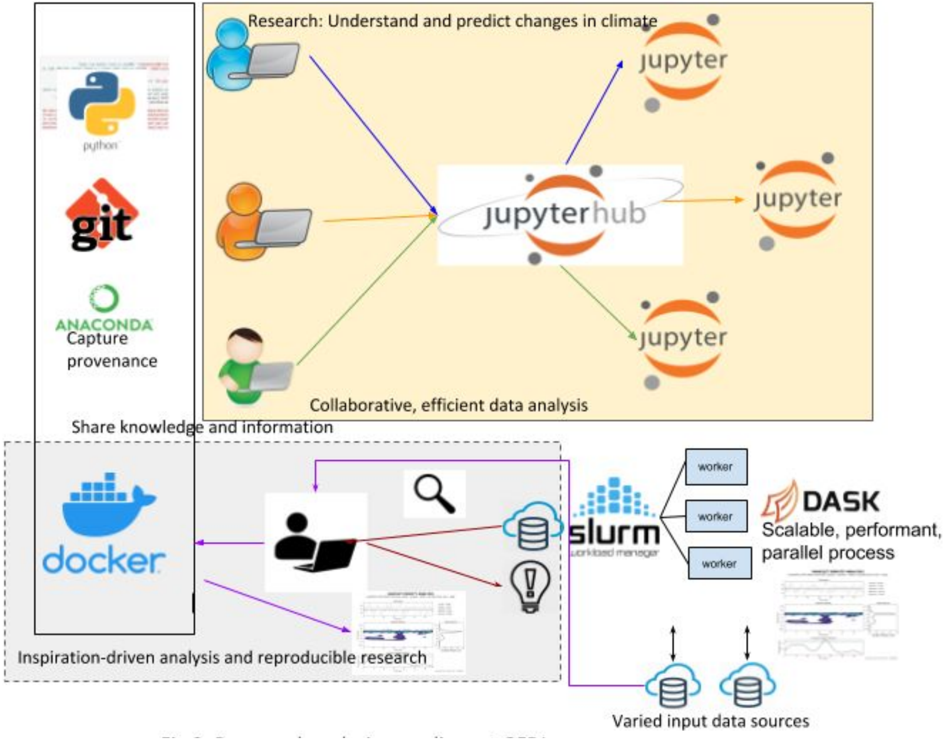


Lazy-greedy user, Lazy dask, Embarrassingly parallel code-or-not

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! Back to building blocks.



A.Radhakrishnan et al, Building blocks for exascale computing at GFDL, Presented at 6th ENES HPC Workshop 2020