Transforming NOAA Data to Analysis Ready Format in the Cloud for ML

Timothy Smith, NOAA PSL
Daniel Abdi, CIRES/NOAA GSL
Sergey Frolov, NOAA PSL

Mariah Pope, EPIC

Matt Long, Parallel Works

Motivation: Emulators doing well, rely on available data

Weather emulators are achieving prediction skill that demands attention.

These emulators rely on high quality, openly available data, like ERA5.

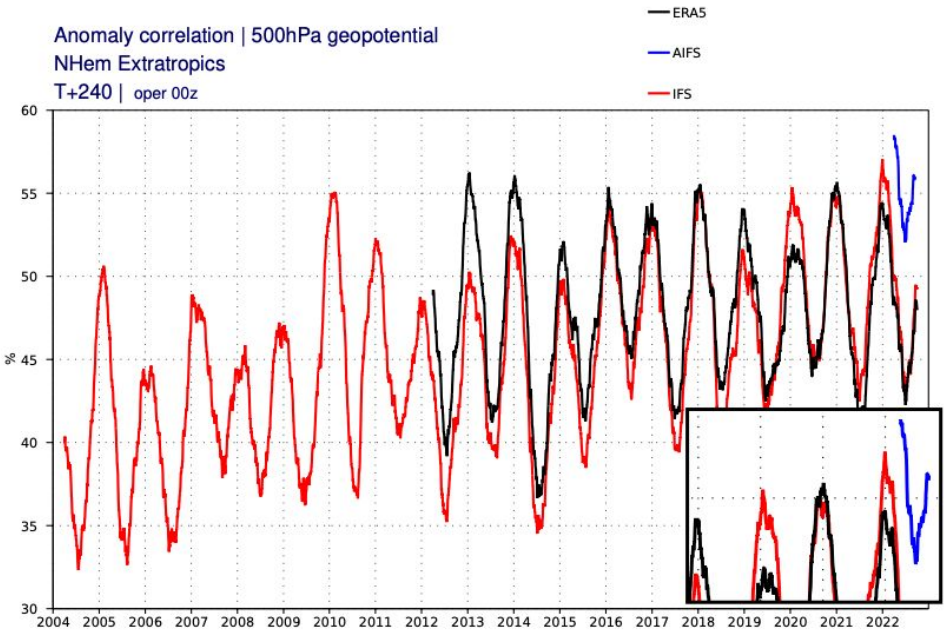
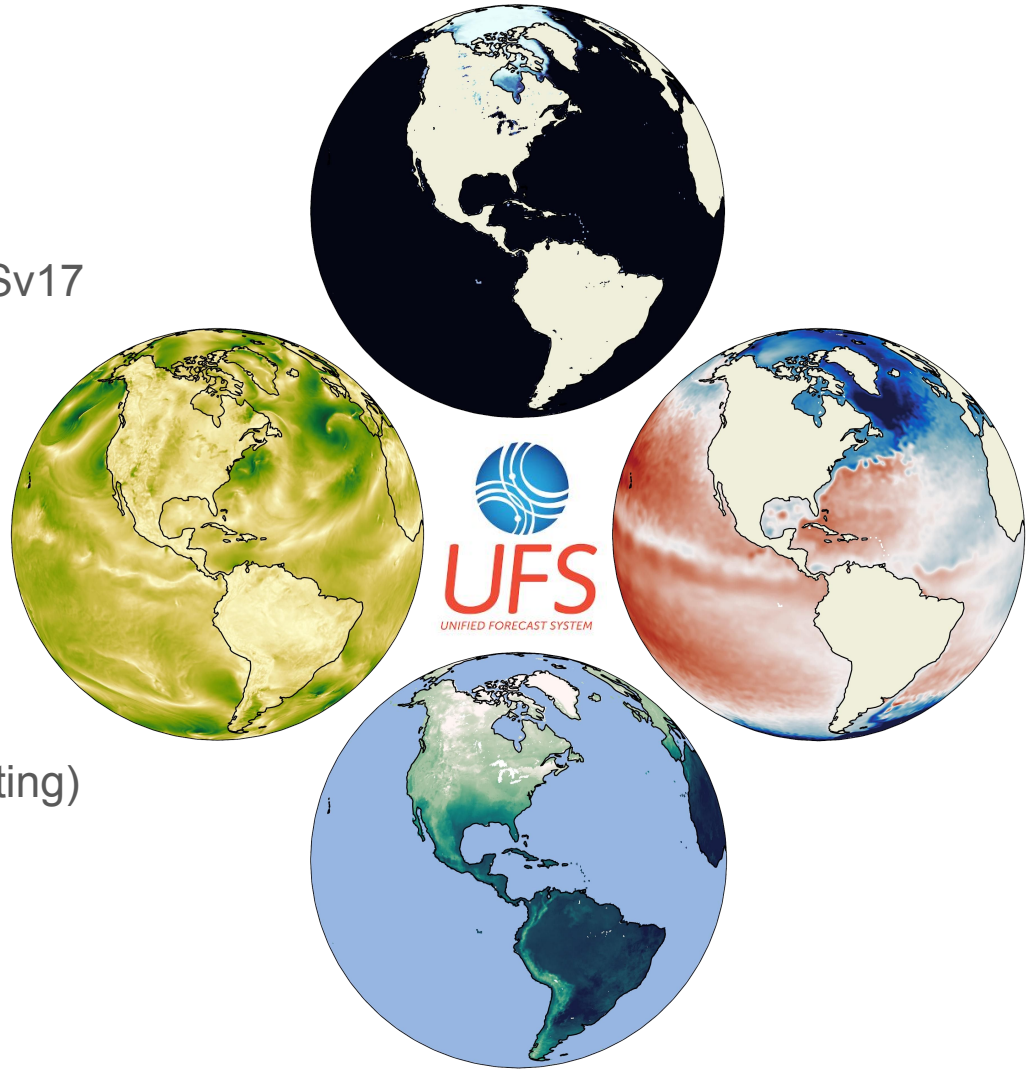


Figure from the AIFS paper
(Lang et al., 2024)

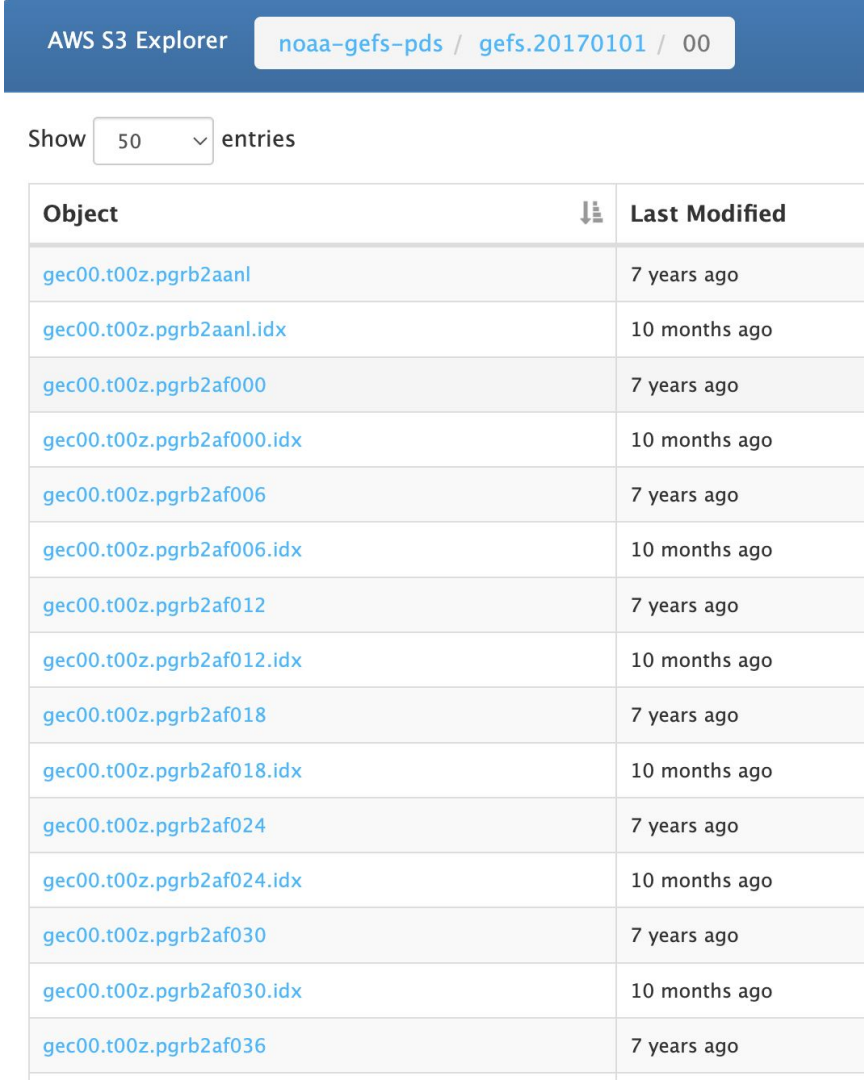
NOAA's UFS Replay

- Produced to initialize GEFSv13/GFSv17 reforecasts
- "Reanalysis-like": nudged toward ERA5/ORAS5
- Fully coupled ufs-hr1
  - Atmosphere: FV3
  - Ocean: MOM6
  - Sea Ice: CICE6
  - Land: NoahMP
  - Wave: WW3
- ¼ degree horizontal resolution
- Spans 1994-2024 (continually updating)
- Approximately 1 Pb of data
- More info here: psl.noaa.gov/data/ufs_replay/



Most of NOAA's Data is not in Analysis Ready, Cloud Optimized (ARCO) Format

How can we improve access to the vast amounts of high quality NOAA data?

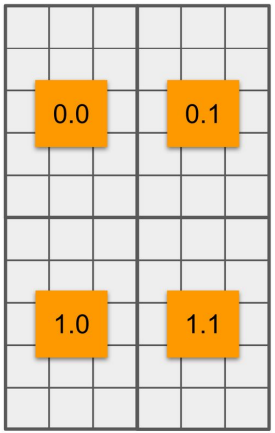


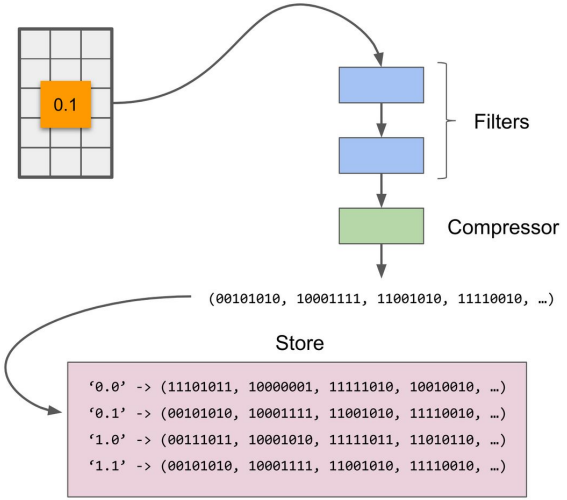
The Zarr Data Format

- A collection of separate files (chunks), which can be written/read independently
  - High throughput I/O
- Ideal for cloud workflows + bucket storage, but works well on HPC too

From wikipedia

E.g., array with shape (10, 6) and chunk shape (5, 3) has 4 chunks in a 2 by 2 chunk grid, with chunks identified by the keys ‘0.0’, ‘0.1’, ‘1.0’, ‘1.1’.






```
[perlmutter partial-one-degree]$ ls forecasts.zarr/
fhr  latitude   longitude  member  pressure  sh2  t   t2m  u10  v10          w
gh   lead_time  lsm        orog    q         sp   t0  u    v    valid_time
[perlmutter partial-one-degree]$ ls forecasts.zarr/t/ | head
0.0.0.0.0.0
0.0.0.1.0.0
0.0.0.10.0.0
0.0.0.11.0.0
0.0.0.12.0.0
0.0.0.13.0.0
0.0.0.14.0.0
0.0.0.15.0.0
0.0.0.16.0.0
0.0.0.17.0.0
```

The Zarr Data Format

- A collection of separate files (chunks), which can be written/read independently
  - High throughput I/O
- Ideal for cloud workflows + bucket storage, but works well on HPC too

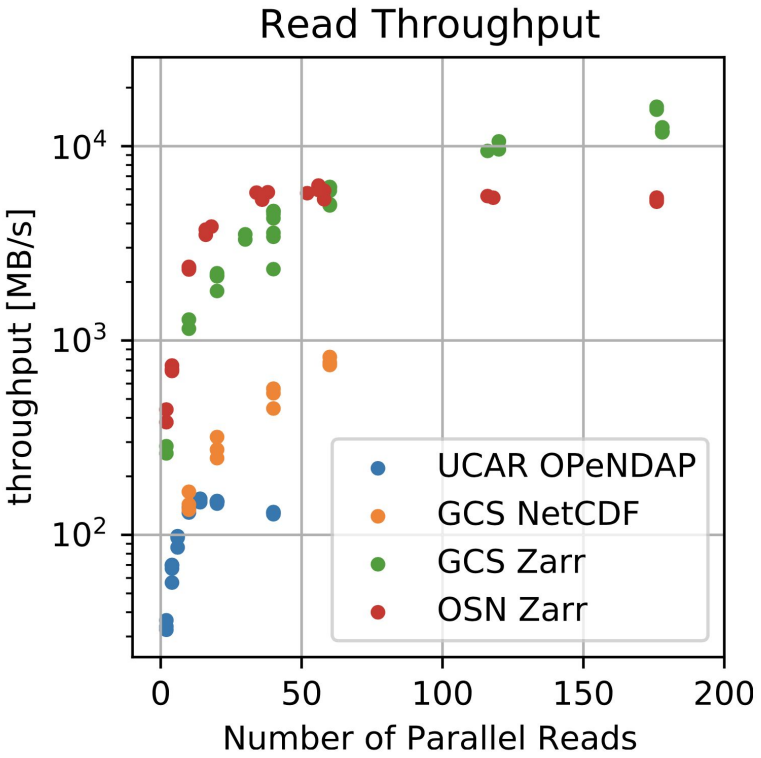


Figure from
Abernathey et al., (2021)

How do we move a Petabyte of data into the cloud?

1) 

xarray-beam

Most runners suited for cloud, but our resources are on-prem, or look like on-prem

2) 

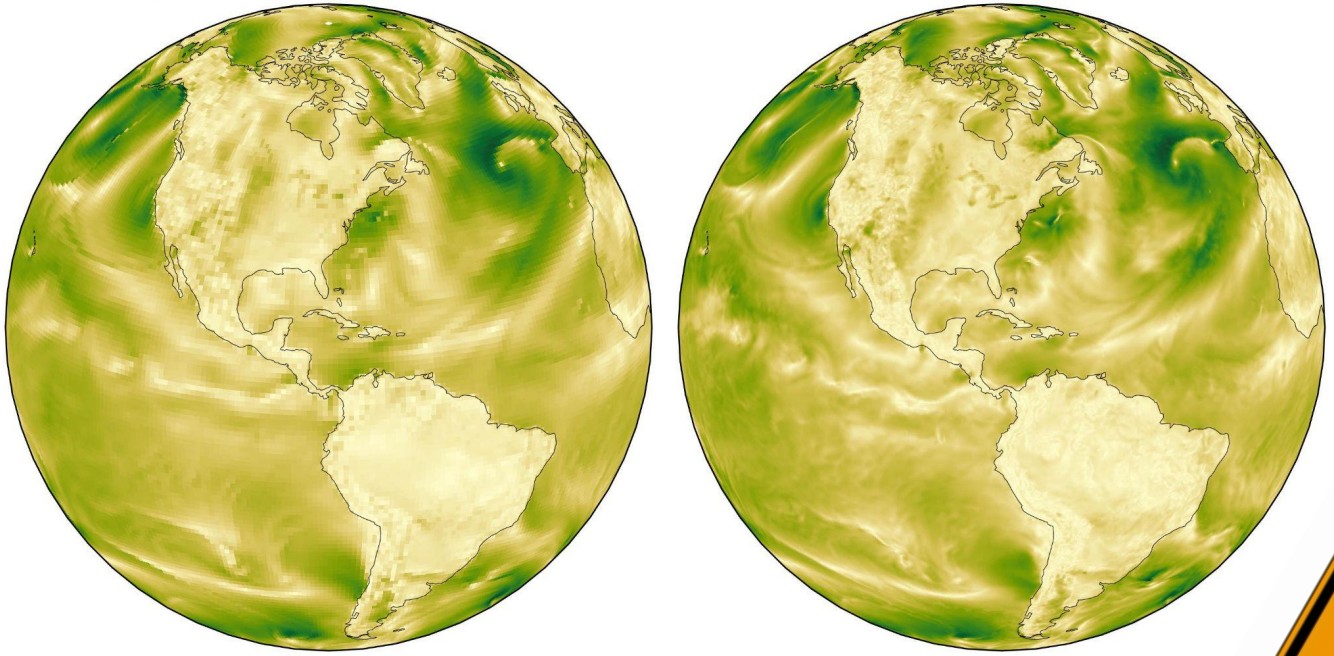
dask-jobqueue


Trouble seeing this scale for very large workloads

… and more

GraphUFS, 2022-12-11T09 + 24h

ERA5, 2022-12-12T09

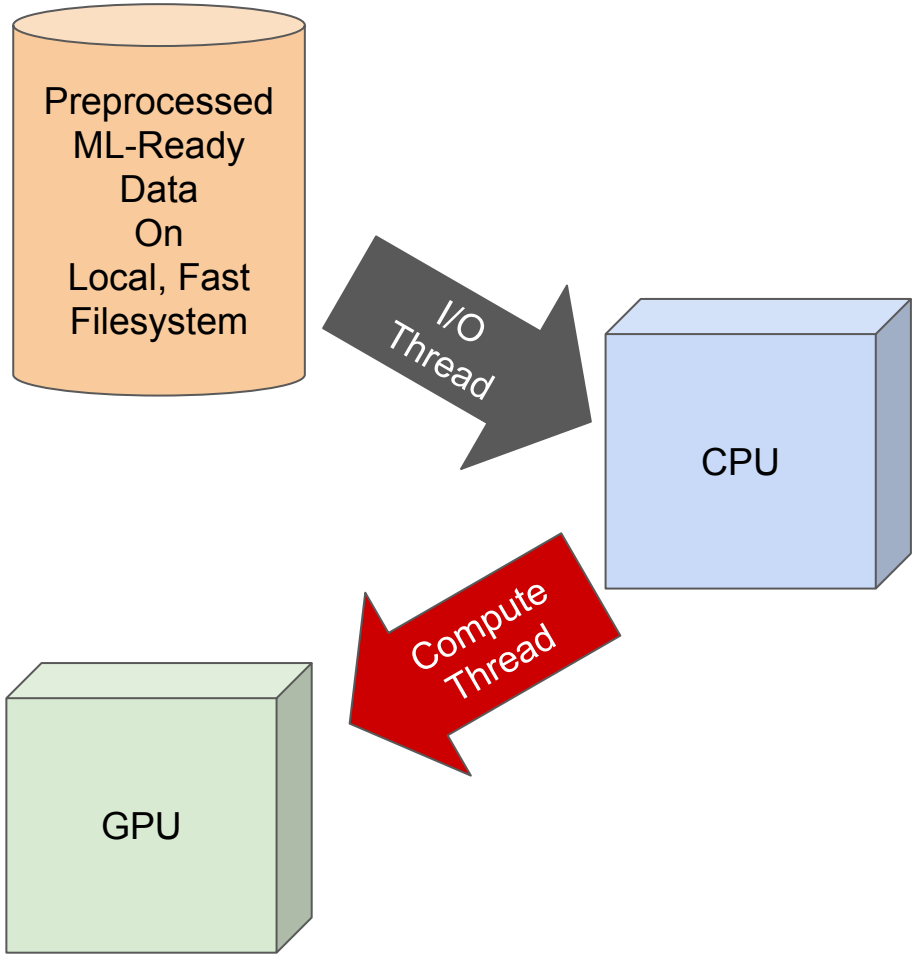


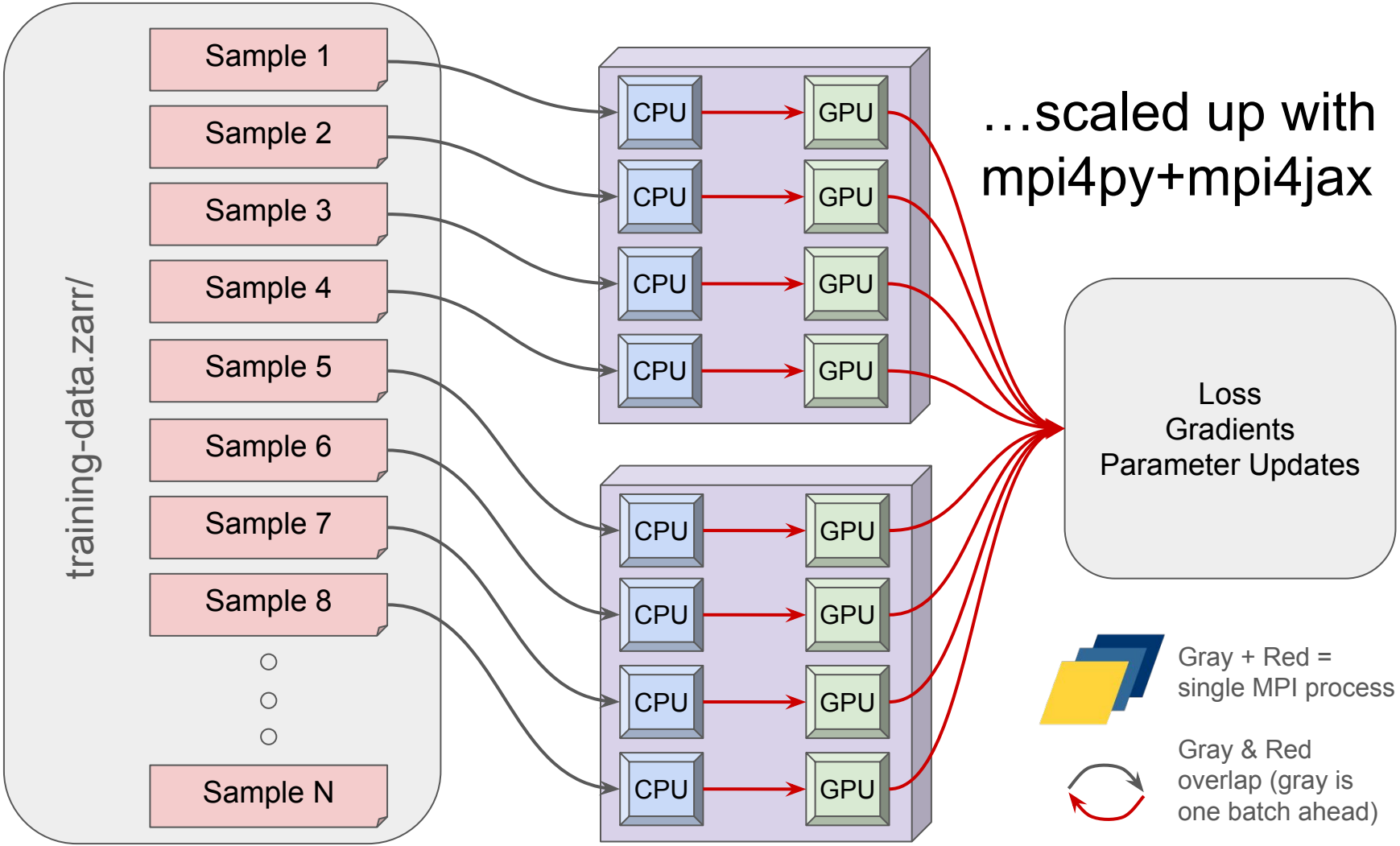
A detour into our ML emulator developments using the GraphCast code base…

Homegrown Dataloader

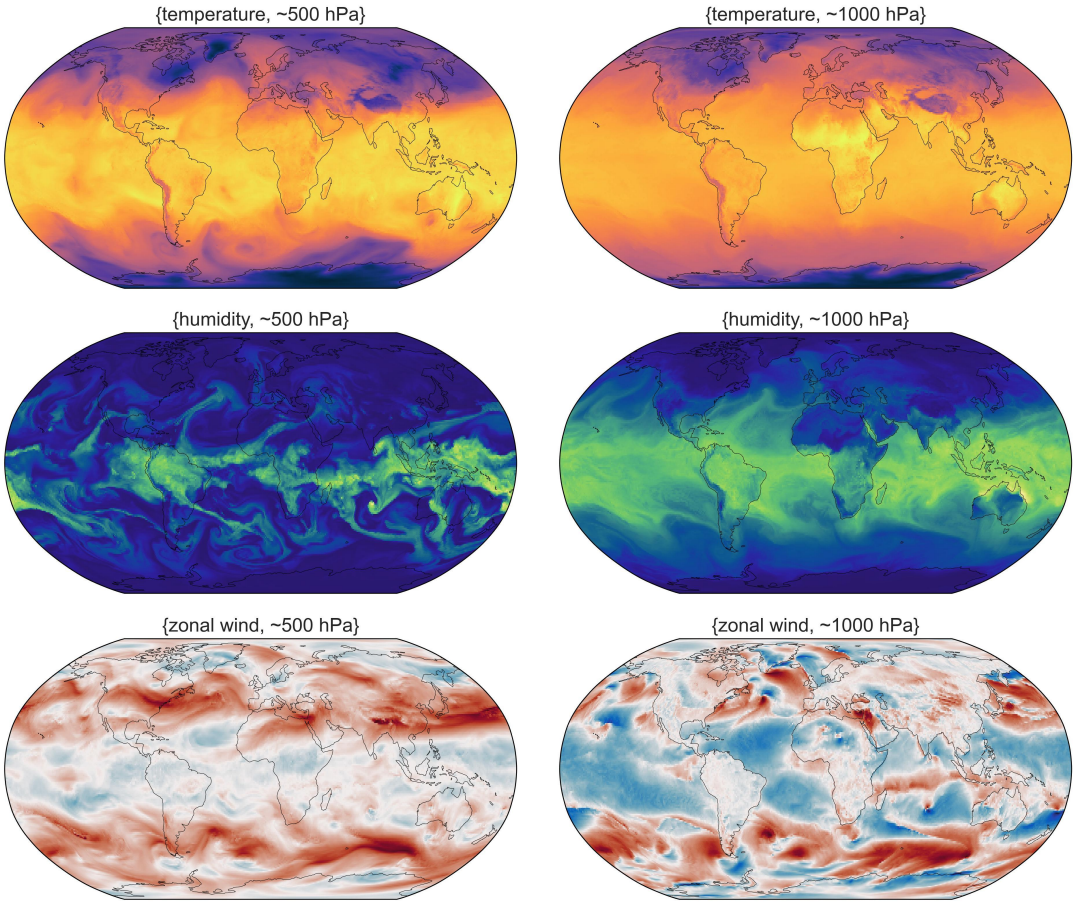
JAX ecosystem does not have any built in dataloading tools

JAX cannot use multiprocessing based dataloaders (e.g. pytorch)

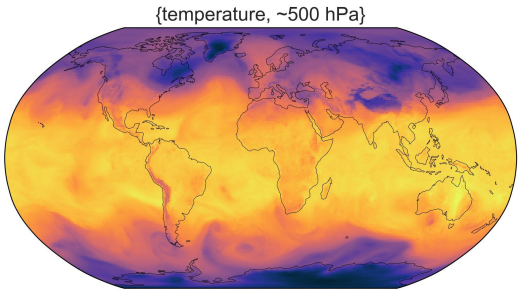


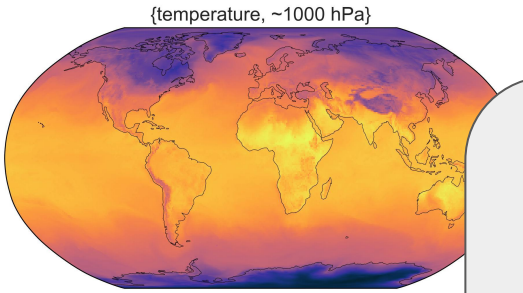

training-data.zarr/
Sample 1
Sample 2
Sample 3
Sample 4
Sample 5
Sample 6
Sample 7
Sample 8
Sample N
CPU
GPU
…scaled up with
mpi4py+mpi4jax
Loss
Gradients
Parameter Updates
Gray + Red =
single MPI process
Gray & Red
overlap (gray is
one batch ahead)

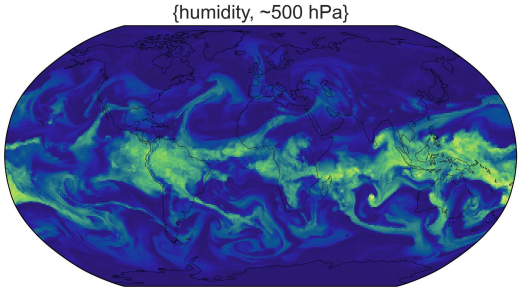
Dataloading at Scale Usually Requires Stacking

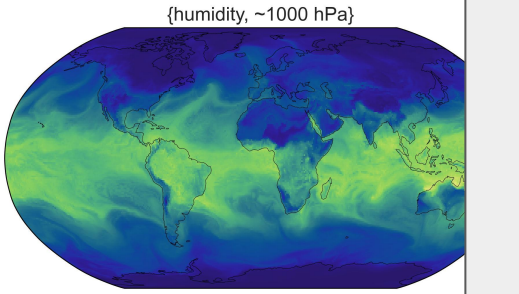


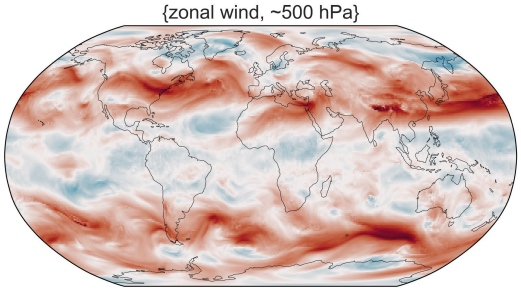
Dataloading at Scale Usually Requires Stacking

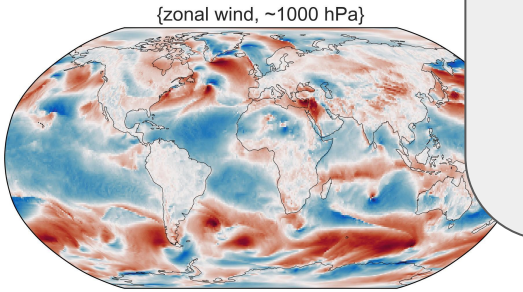




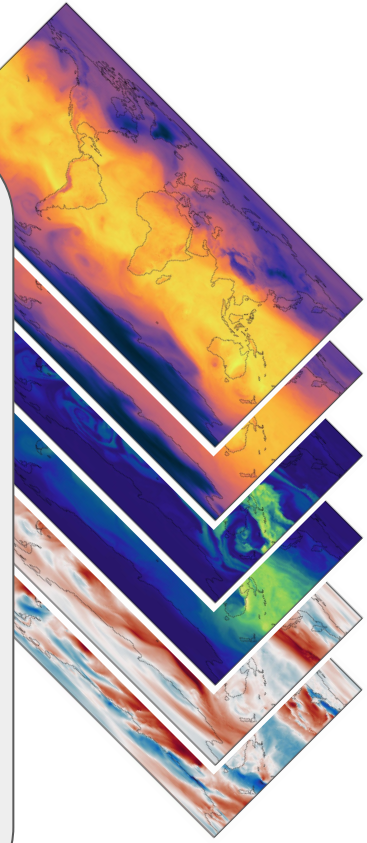






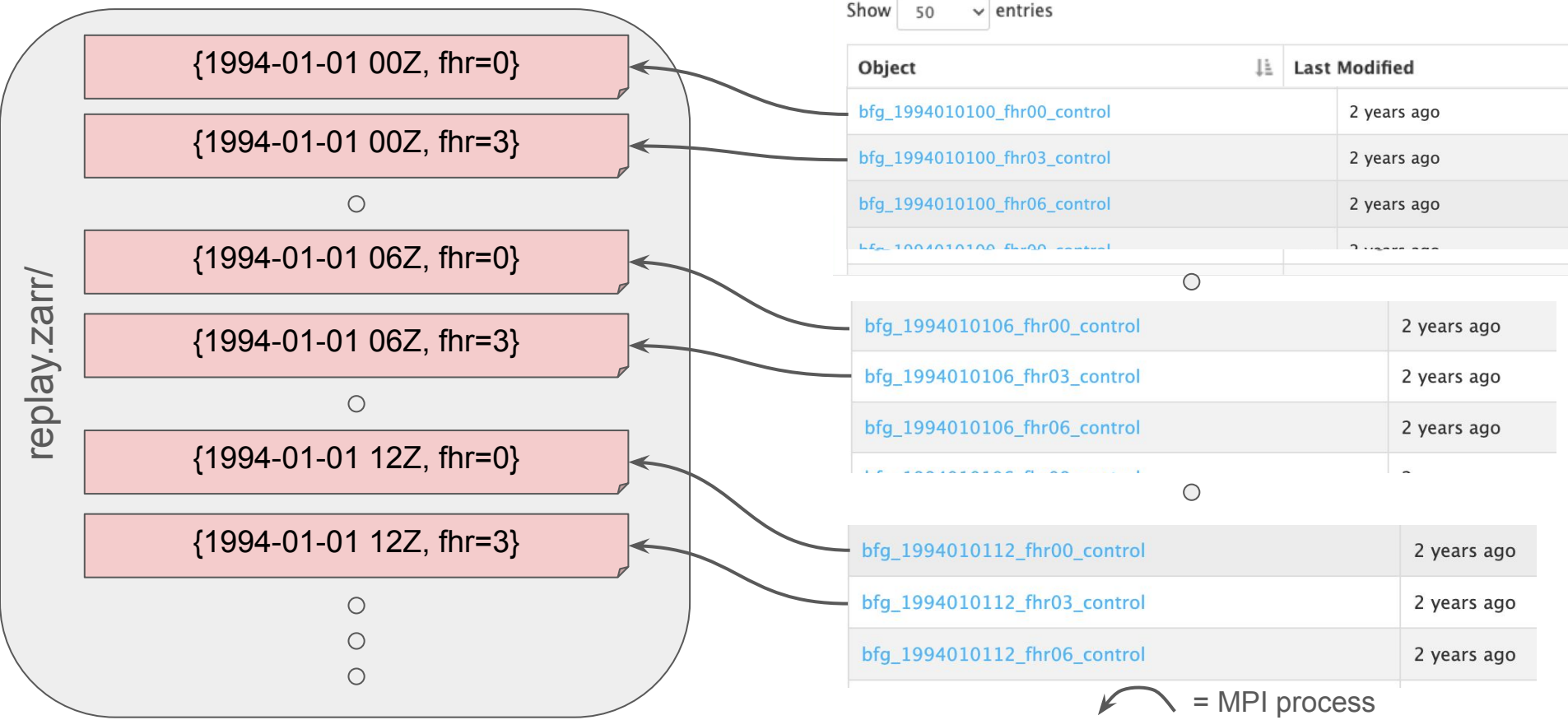


While creating the Dataloader, we have defined the mapping from a full multivariate dataset to an enumeration of the smallest chunk that we care about

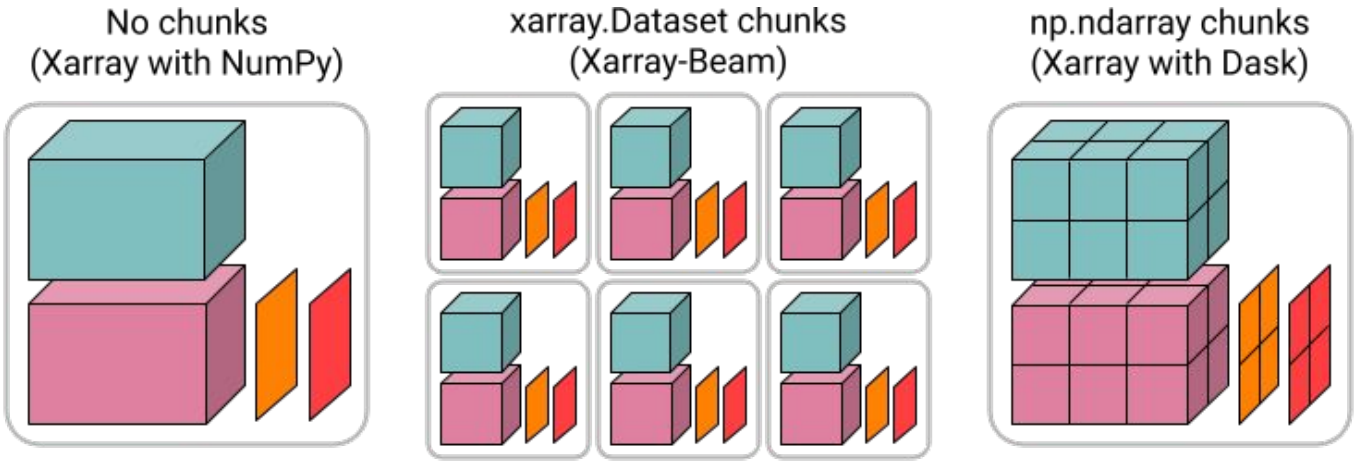


Dimensions:
[sample, lat, lon, channel]

Stacked Mapping = Enumerated, Multi-index



Workflow: monster dataset from many small datasets



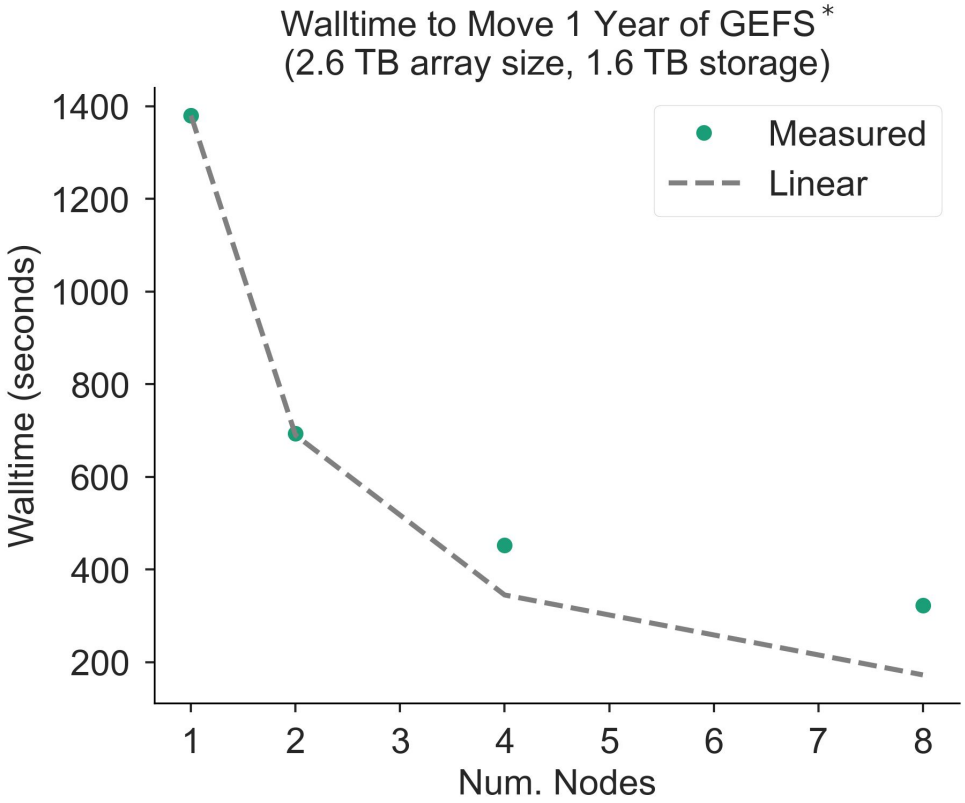
From xarray-beam docs

When generating **very large** datasets, viewing the entire dataset (e.g. with dask) creates scheduling challenges, and does not appear to be robust. Workflows that manage small subsets of the dataset appear scalable & more robust.

Strong Scaling Tests on Perlmutter (DOE HPC)

- Moving 1 year of NOAA's Global Ensemble Forecast System (GEFS), with
  - All ensemble members
  - "The usual" ML weather emulator variables
  - Forecast hour 0 & 6
- Lustre file system suboptimal for zarr, but still works well
- Test case is flipped, moving cloud to on-prem filesystem, but still illustrative

Performance is promising:
1 Pb in 35 hours with 8 nodes??
🚀



https://noaa-ufs-gefsv13replay-pds.s3.amazonaws.com/index.html#1deg/1994/01/1994010100/

NOAA's UFS Replay


```
In [1]: import xarray as xr

In [2]: atm = xr.open_zarr(
            "gs://noaa-ufs-gefsv13replay/ufs-hr1/0.25-degree/03h-freq/zarr/fv3.zarr",
            storage_options={"token": "anon"},
        )

In [3]: atm

Out[3]: xarray.Dataset
```

▸ Dimensions: (**time**: 87020, **grid_yt**: 768, **grid_xt**: 1536, **pfull**: 127)

▸ Coordinates: (6)

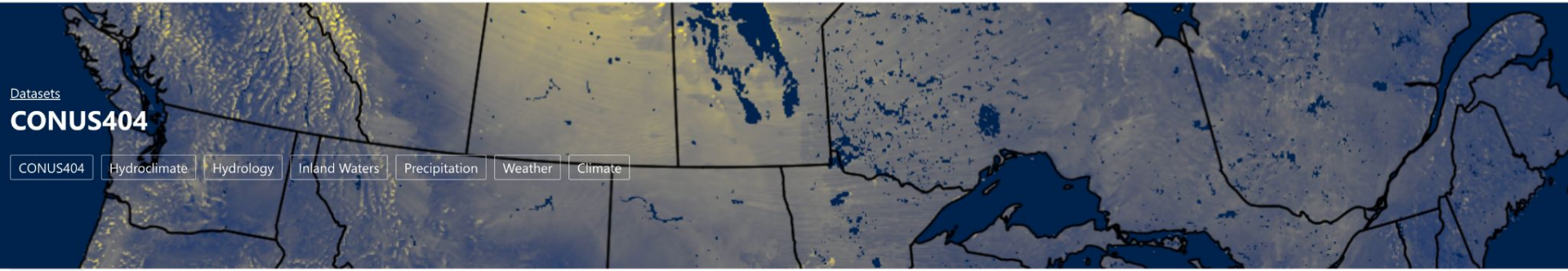
▾ Data variables:

acond	(time, grid_yt, grid_xt)	float32	dask.array<chunksize=(1, 768, 1536)...
albdo_ave	(time, grid_yt, grid_xt)	float32	dask.array<chunksize=(1, 768, 1536)...
alnsf	(time, grid_yt, grid_xt)	float32	dask.array<chunksize=(1, 768, 1536)...
alnwf	(time, grid_yt, grid_xt)	float32	dask.array<chunksize=(1, 768, 1536)...

[Sample Notebook ... & more info]

~1 Pb

Thanks to [NOAA Open Data Dissemination (NODD)]



Datasets

CONUS404

CONUS404 | Hydroclimate | Hydrology | Inland Waters | Precipitation | Weather | Climate

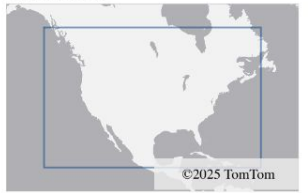
Overview | Example Notebook

Overview

CONUS404 is a unique, high-resolution hydro-climate dataset appropriate for forcing hydrological models and conducting meteorological analysis over the conterminous United States. CONUS404, so named because it covers the CONterminous United States for over 40 years at 4 km resolution, was produced by the Weather Research and Forecasting (WRF) model simulations run by NCAR as part of a collaboration with the USGS Water Mission Area. The CONUS404 includes 42 years of data (water years 1980-2021) and the spatial domain extends beyond the CONUS into Canada and Mexico, thereby capturing transboundary river basins and covering all contributing areas for CONUS surface waters.

The CONUS404 dataset, produced using WRF version 3.9.1.1, is the successor to the CONUS1 dataset in ds612.0 (Liu, et al., 2017) with improved representation of weather and climate conditions in the central United States due to the addition of a shallow groundwater module and several other improvements in the NOAH-Multiparameterization land surface model. It also uses a more up-to-date and higher-resolution reanalysis dataset (ERA5) as input and covers a longer period than CONUS1.

Spatial Extent



Azure data region

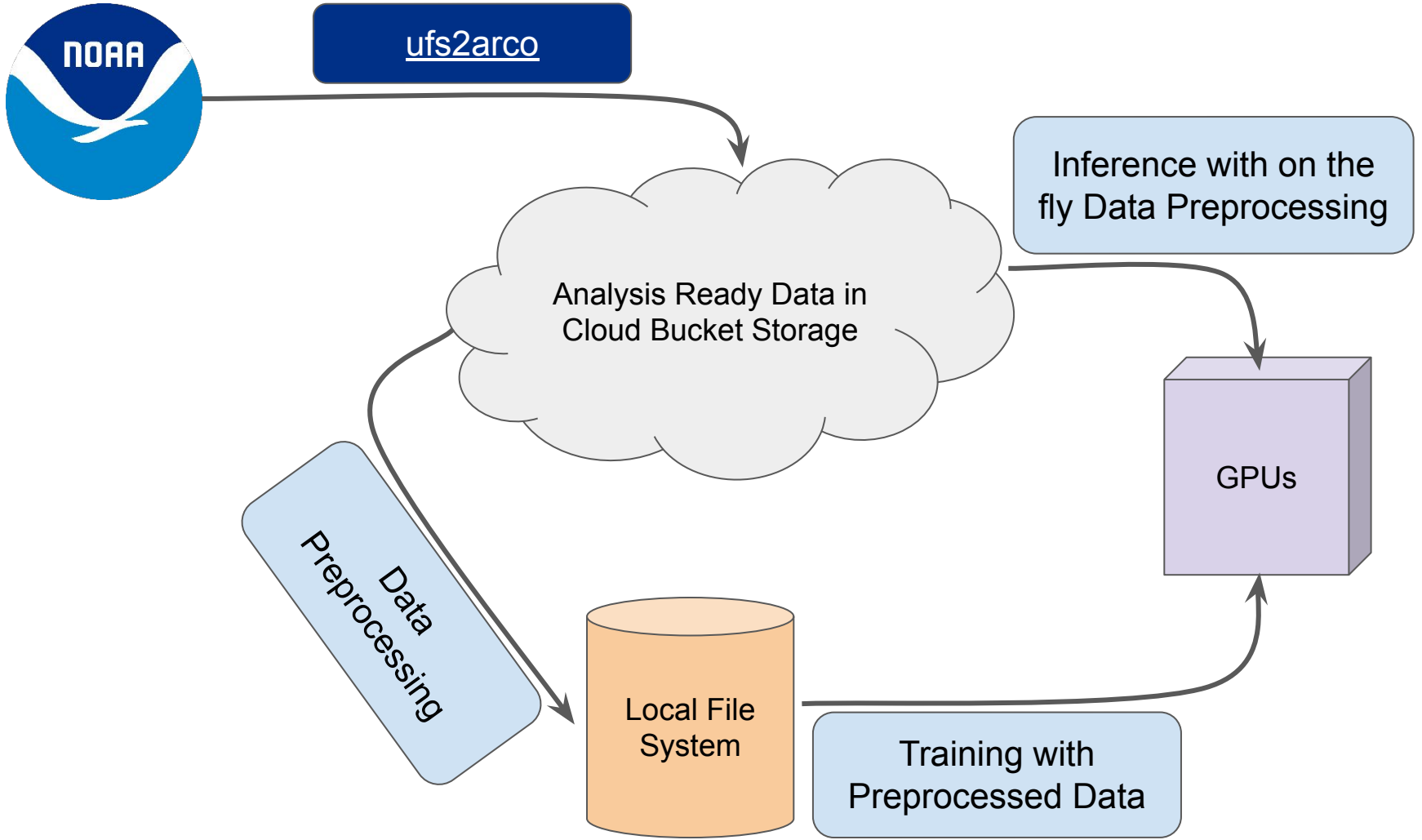
eastus

Temporal Extent

Planetary Computer Link
More info from NCAR and USGS

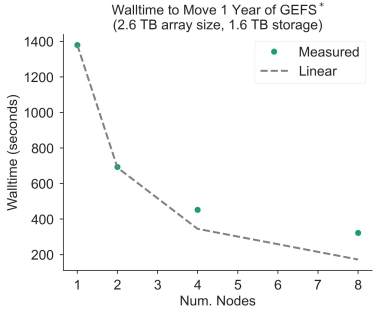
~2.3 Pb

Thanks to NOAA Open Data Dissemination (NODD)


ufs2arco
Inference with on the fly Data Preprocessing
Analysis Ready Data in Cloud Bucket Storage
GPUs
Data Preprocessing
Local File System
Training with Preprocessed Data

Key Takeaways

- By writing our own dataloader to work with JAX, found a way to harness on-prem resources for a scalable ARCO data mover
- Workflow: monster dataset from small datasets appears scalable
- Check out the Replay data on GCS & CONUS404 on Planetary Computer / Azure!
  - Big thanks to NOAA Open Data Dissemination (NODD)! 🚀



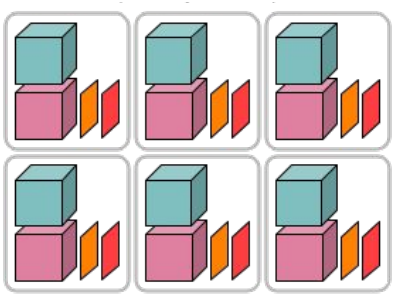
Future Developments

- Code is openly developed as a community resource & reference at github.com/NOAA-PSL/ufs2arco, but it is imperfect!

Robustness to missing data | Zarr v3 | Icechunk?

xarray-beam + Flink Runner + SLURM?

- Use this backbone to improve anemoi-datasets xarray-zarr workflow
- VirtualiZarr to improve access to existing datasets



From xarray-beam docs

Questions/Comments?
Let's chat!
tim.smith@noaa.gov
or on [LinkedIn]