earthmover

Earth system data in the 2020s and beyond

Zarr/Xarray/Icechunk/Kerchunk and all that

OOI FB & DSC | Nov 2024

about me

Deepak Cherian, PhD

FORWARD ENGINEER

- Physical oceanographer
- Open source maintainer & community leader

PRIOR EXPERIENCE

NCAR Operated by UCAR

WOODS HOLE OCEANOGRAPHIC INSTITUTION

OPEN SOURCE CONTRIBUTIONS

xarray

icechunk

Zarr

PANGEO

CF xarray

flox

earthmover

Earthmover PBC

MISSION STATEMENT

To empower people to use scientific data to solve humanity’s greatest challenges

OUR TEAM COMPOSITION

- ✓ Accomplished scientists.
- ✓ Open source community leaders.
- ✓ Seasoned engineers who have built production-grade scientific data infrastructure at top companies.

Ryan Abernathey
CO-FOUNDER & CEO

Joe Hamman
CO-FOUNDER & CTO

OUR PRIOR EXPERIENCE

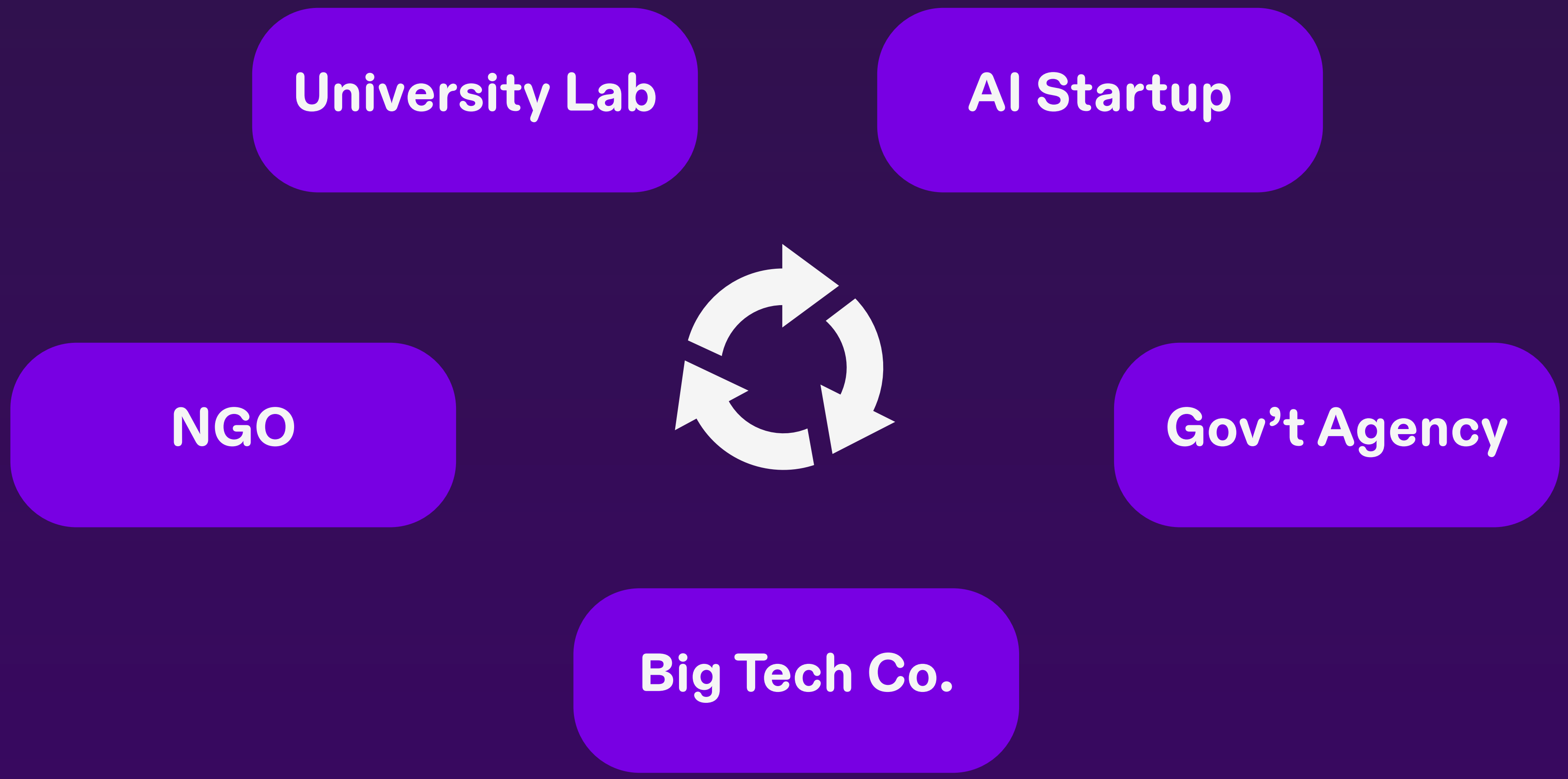
Google
aws
UNIVERSITY of WASHINGTON
MIT
NCAR
planet.
COLUMBIA UNIVERSITY
THE CLIMATE CORPORATION

OUR OPEN-SOURCE LEADERSHIP

PANGEO
Zarr
xarray

our vision

planetary-scale data federation in the cloud



University Lab

AI Startup

NGO

Gov’t Agency

Big Tech Co.

Seamless exchange of data across academia, government, and industry will empower reproducible open science and accelerate climate change solutions.

Our open source footprint

- ✓ Xarray: N-D labeled arrays and datasets in Python
  - → Analyze array data using metadata (e.g. dimension names)
  - → `dataset.mean(dim=“time”)` instead of mean(data, axis=0)

- ✓ Zarr: Chunked-compressed N-D Arrays
  - → Highly scalable cloud-native data format and API
  - → Widely used in all cloud object stores (S3, GCS, ABS, etc.)
  - → Some drawbacks around collaboration

- ✓ Icechunk: A new, open-source transactional storage engine for Zarr
  - → Consistency guarantees
  - → Version control for dataset

The prompt

From Jim Potemra

Zarr as a data model for the cloud

the relationship between netCDF and zarr/xarray.

Right now, OOI pushes out data in netCDF, but NSF is asking about "AI/ML" ready data and about "big data" issues.

Timeline

PANGEO

<2018

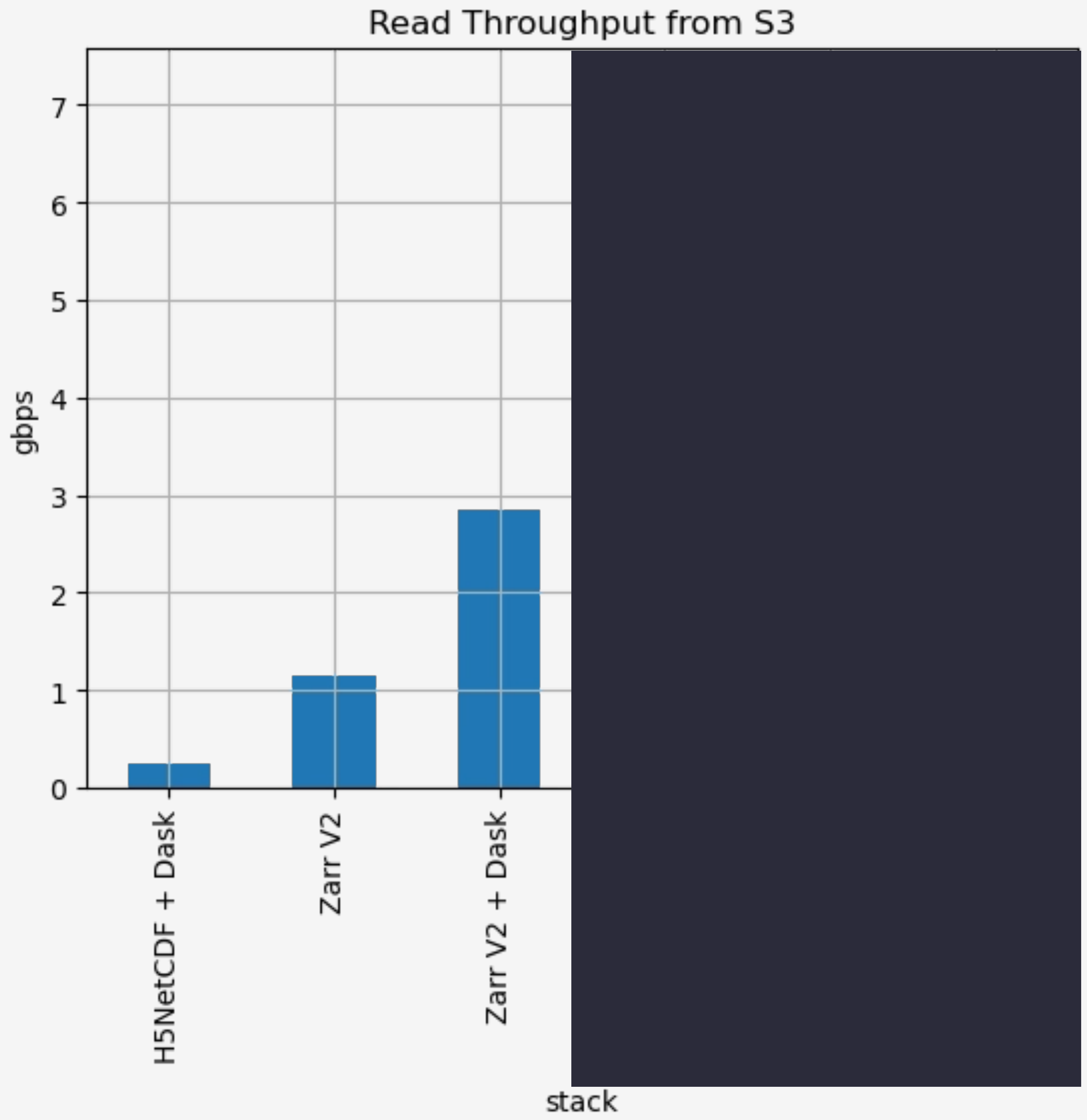
HDF

- ✓ Not cloud optimized (though this is changing)
- ✓ Hierarchical (groups)
- ✓ Wide adoption

>2018

Zarr

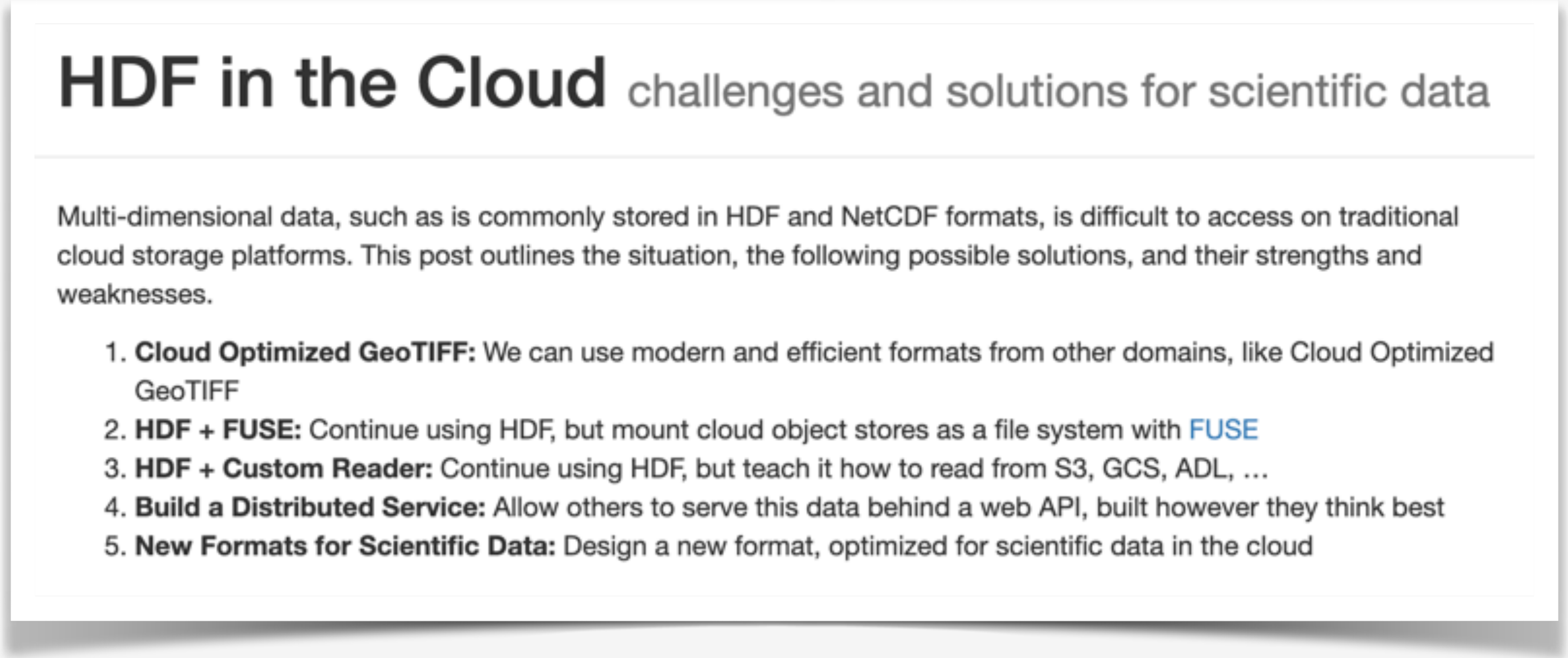
- ✓ Open-source
- ✓ Cloud-native
- ✓ Hackable
- ✓ Scalable
- ✓ Not transactional



[redacted]

Arrays 🤨 Cloud

- Most cost-effective and performant shared stored in cloud is **object storage**, not **file storage**
- With object storage, data are read and written via high-latency HTTP calls
- Archival file formats designed for POSIX storage either don't work at all or have very poor performance with object storage.
- Though HDF is moving to address this.
- Note netCDF4 is HDF5.



HDF in the Cloud challenges and solutions for scientific data

Multi-dimensional data, such as is commonly stored in HDF and NetCDF formats, is difficult to access on traditional cloud storage platforms. This post outlines the situation, the following possible solutions, and their strengths and weaknesses.

1. **Cloud Optimized GeoTIFF:** We can use modern and efficient formats from other domains, like Cloud Optimized GeoTIFF
2. **HDF + FUSE:** Continue using HDF, but mount cloud object stores as a file system with FUSE
3. **HDF + Custom Reader:** Continue using HDF, but teach it how to read from S3, GCS, ADL, …
4. **Build a Distributed Service:** Allow others to serve this data behind a web API, built however they think best
5. **New Formats for Scientific Data:** Design a new format, optimized for scientific data in the cloud

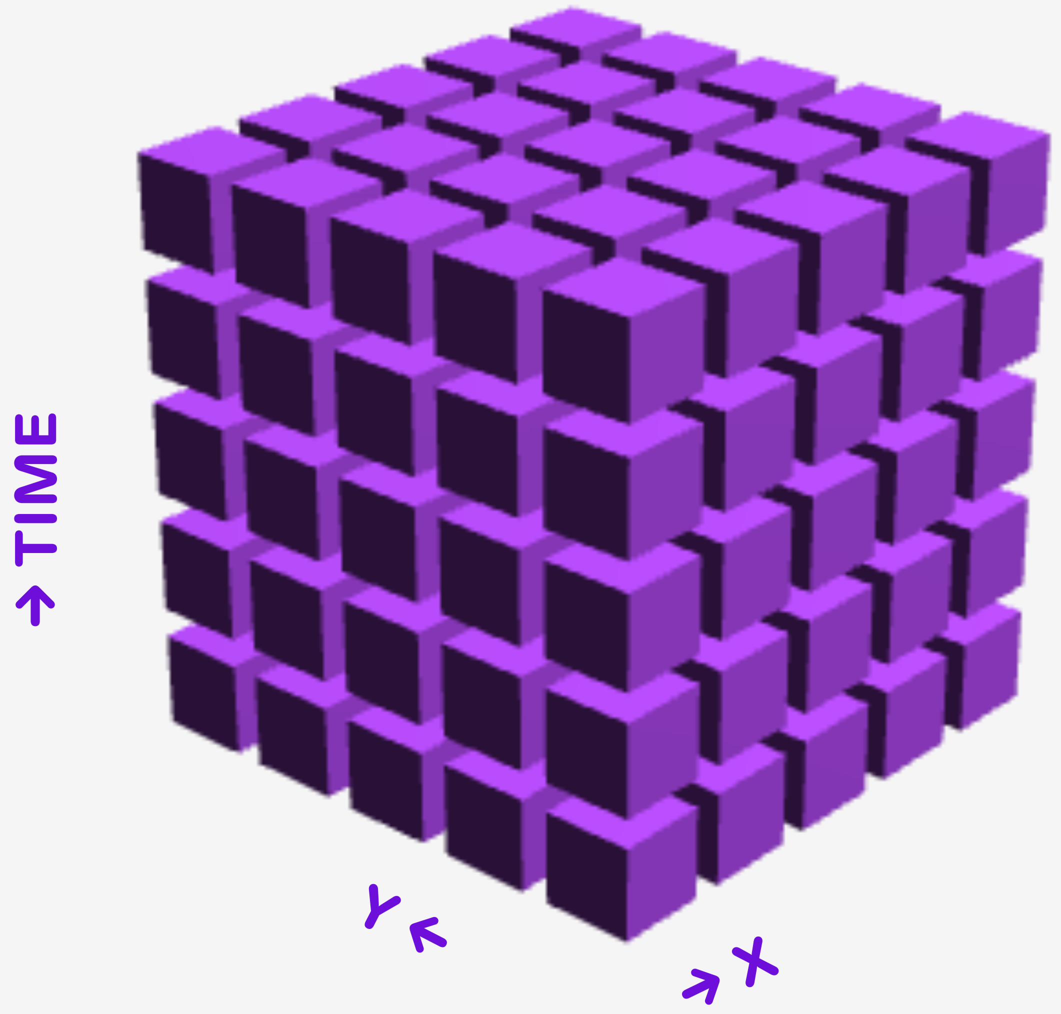
One of the original foci for the Pangeo Project!

https://matthewrocklin.com/blog/work/2018/02/06/hdf-in-the-cloud

Zarr data model

→ scalable, cloud-optimized array format

Zarr



Chunked, compressed
multidimensional array.

- ✓ **Arbitrary metadata (CF or otherwise)**
- ✓ Can be self-describing in the netCDF sense
- ✓ Support arbitrary hierarchies
  - ✓ Core structures are Array, Group
- ✓ Open source cloud-native format
- ✓ No limit to size of Arrays
- ✓ Read / write arrays in parallel
- ✓ Scalable with cloud object storage

EXAMPLE ZARR V3 KEYS

```
mygroup/zarr.json
mygroup/myarray/zarr.json
mygroup/myarray/c/0/0
mygroup/myarray/c/0/1
```

Zarr can be netCDF if you want it to be

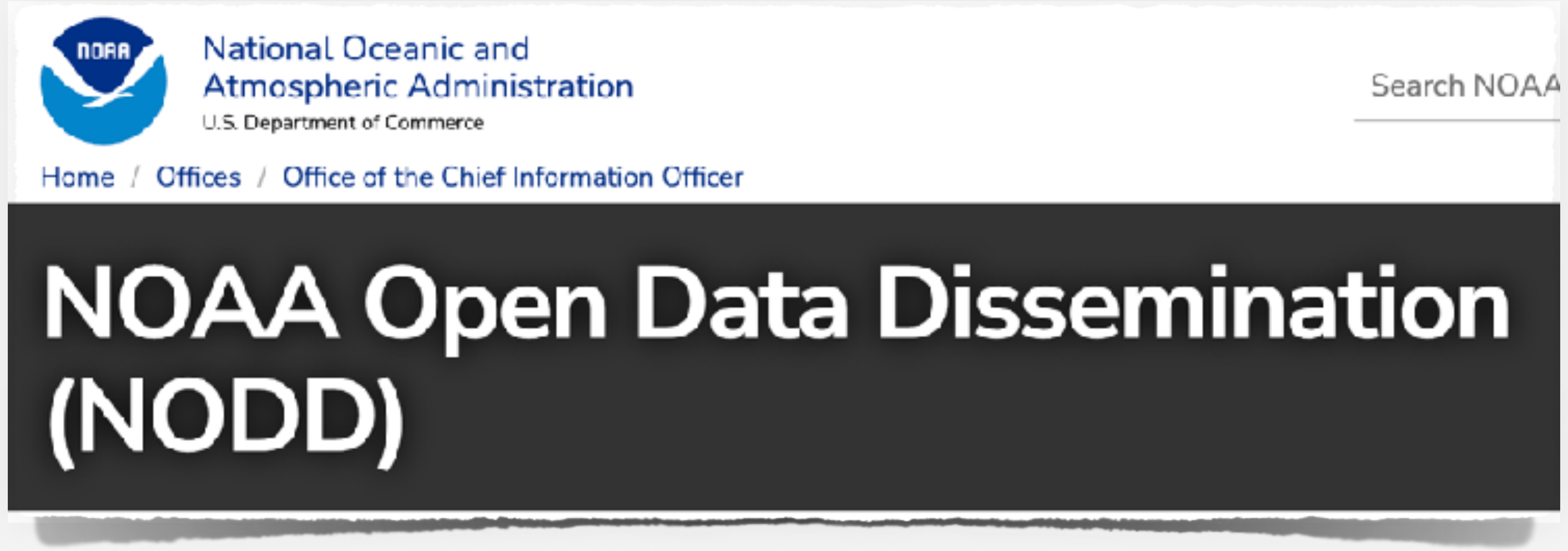
→ And more

- (DEMO)
  - https://app.earthmover.io/earthmover-demos/hrrr
  - More

“Lift and shift” is prevalent

→ Despite Zarr being great

- ✓ Large data providers are adopting cloud as a means of distributing data
- ✓ ...but are mostly just moving existing file archives to object storage without rethinking data architecture
- ✓ Result: cloud is being used like a big FTP server, limiting the impact of the cloud migration

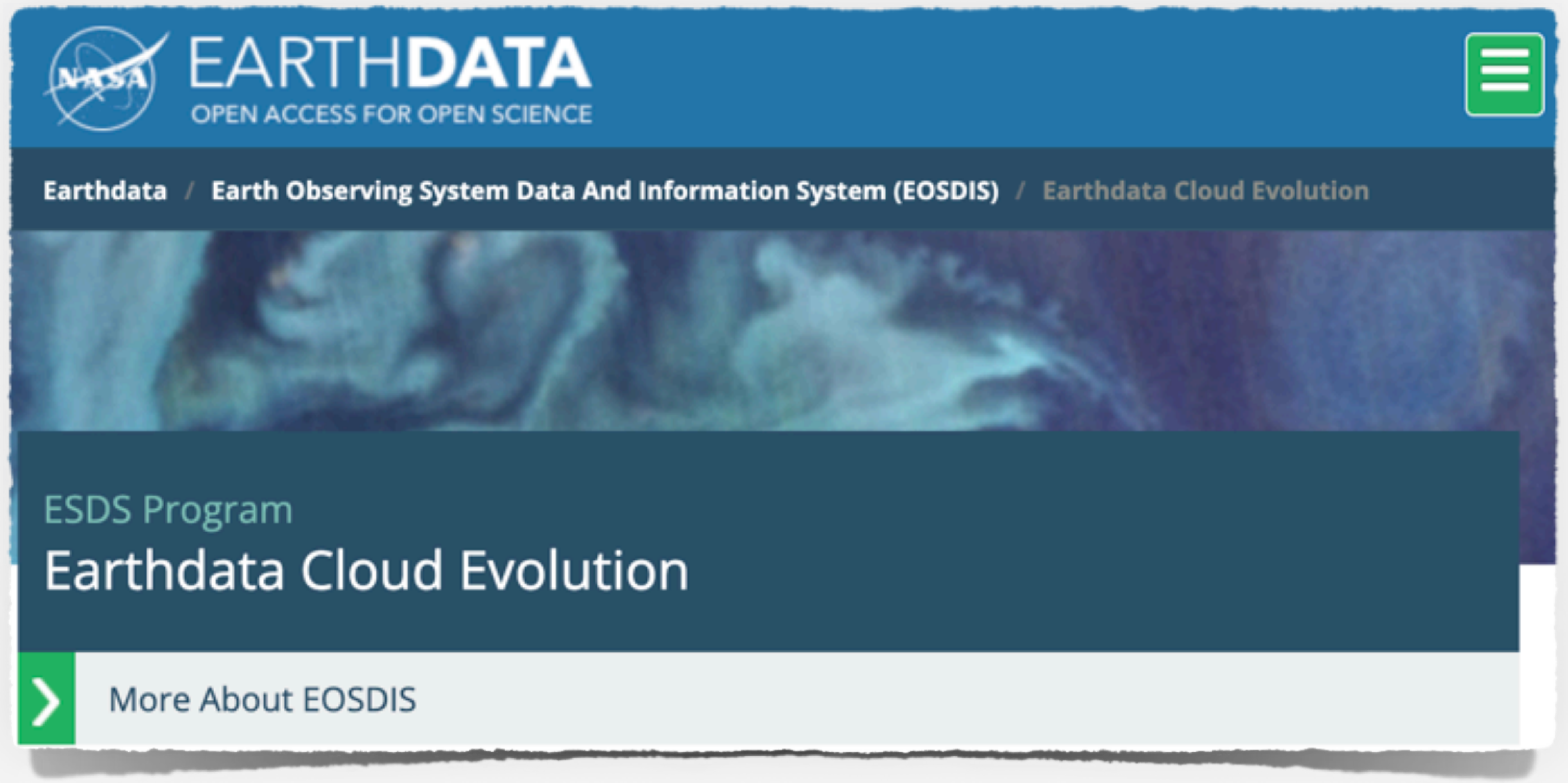


National Oceanic and Atmospheric Administration
U.S. Department of Commerce

Search NOAA

Home / Offices / Office of the Chief Information Officer

NOAA Open Data Dissemination (NODD)



EARTHDATA
OPEN ACCESS FOR OPEN SCIENCE

Earthdata / Earth Observing System Data And Information System (EOSDIS) / Earthdata Cloud Evolution

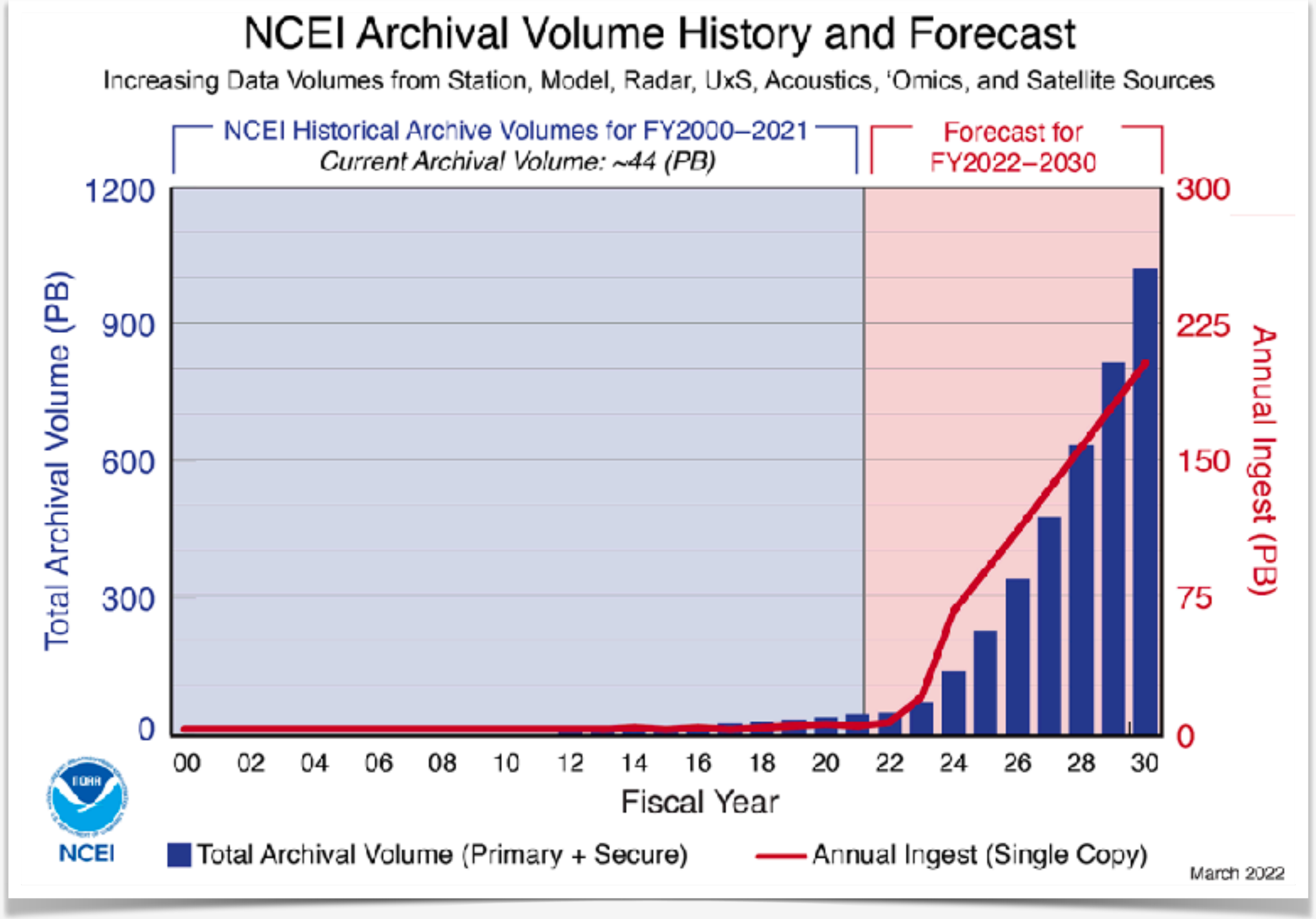
ESDS Program

Earthdata Cloud Evolution

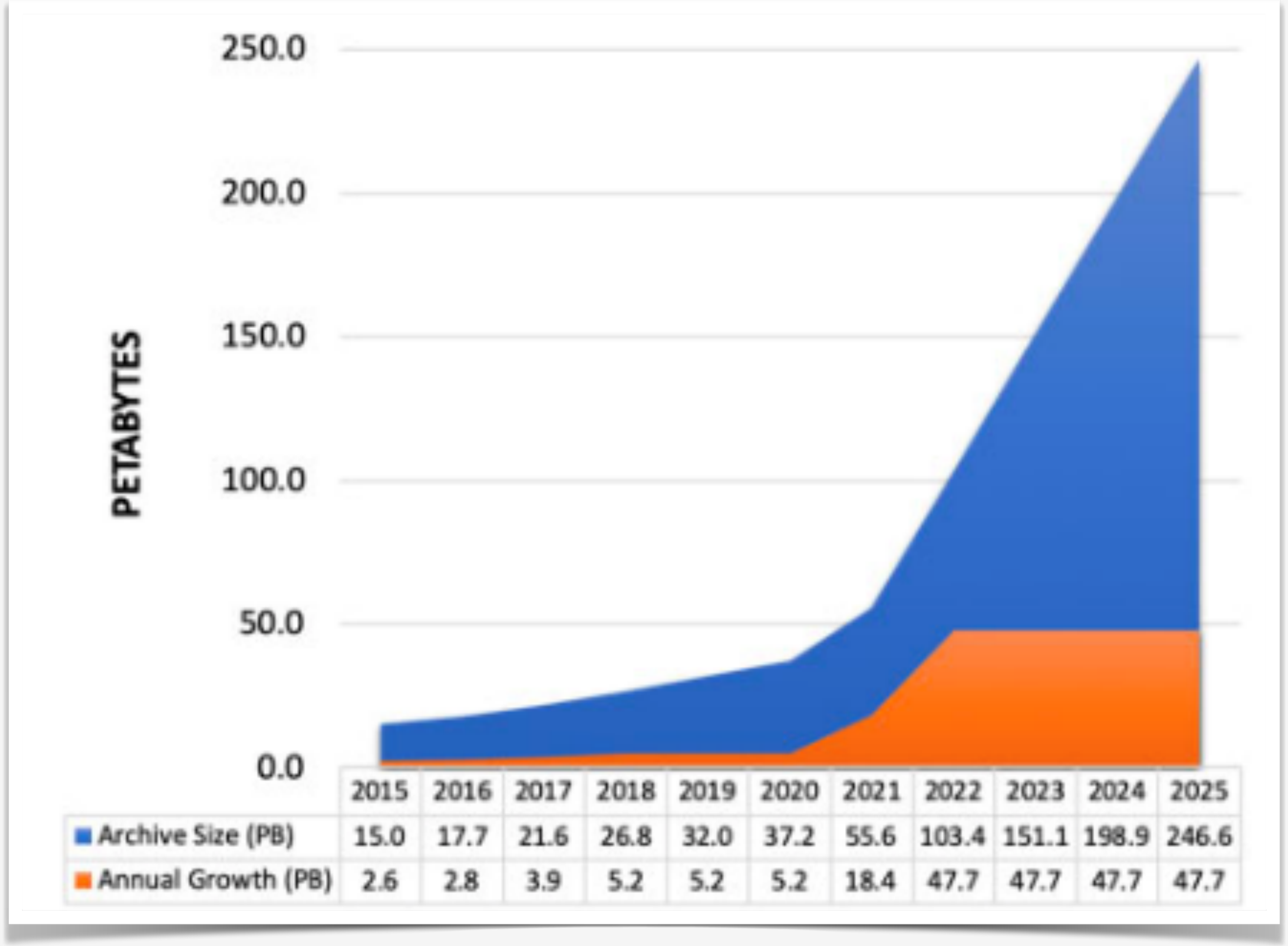
More About EOSDIS

The Gravity of Archival Formats

NOAA NCEI



NASA EOSDIS



	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025
🟦 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
🟧 Annual Growth (PB)	2.6	2.8	3.9	5.2	5.2	5.2	18.4	47.7	47.7	47.7	47.7

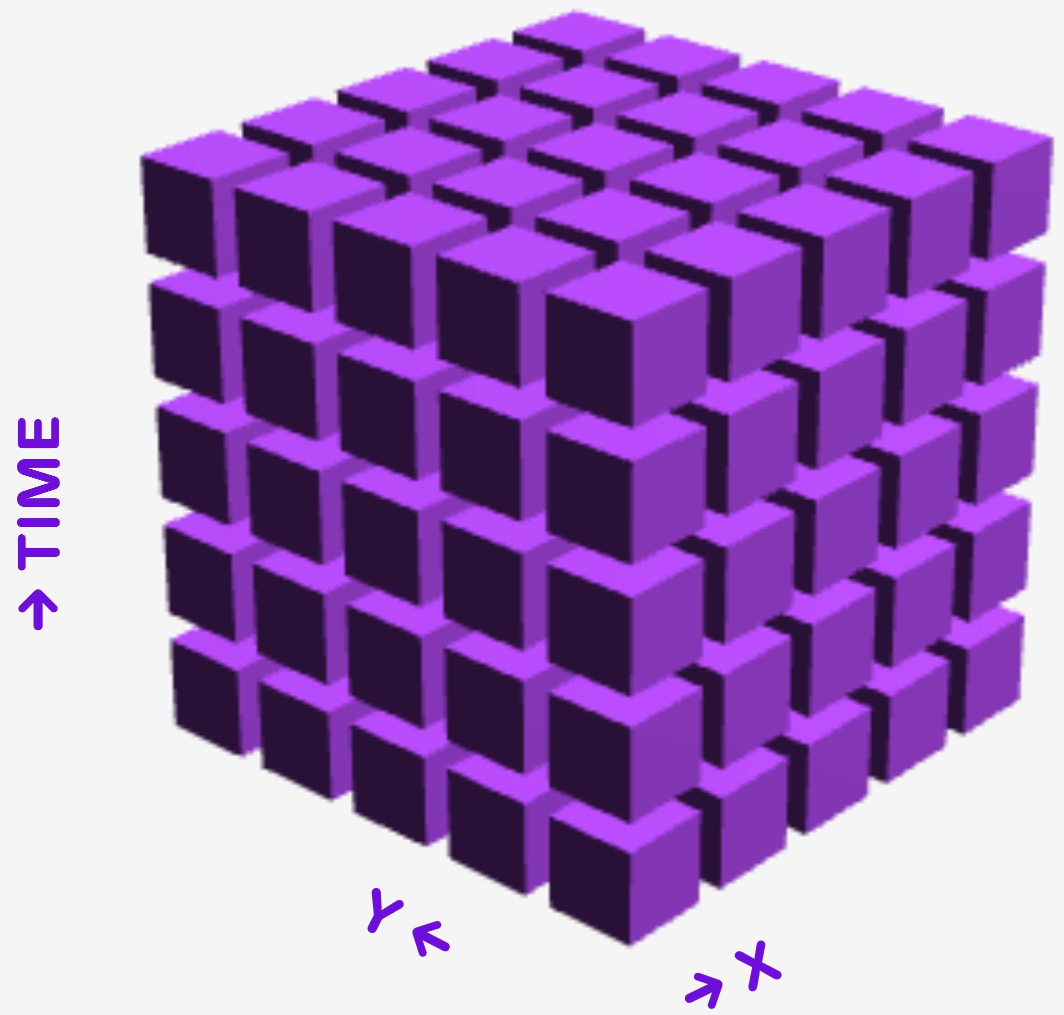
Almost none of this is Zarr. Mostly NetCDF, GRIB, and TIFF.

The kerchunk idea: everything is a zarr store

→ It’s really great that Zarr is a hackable community python project.

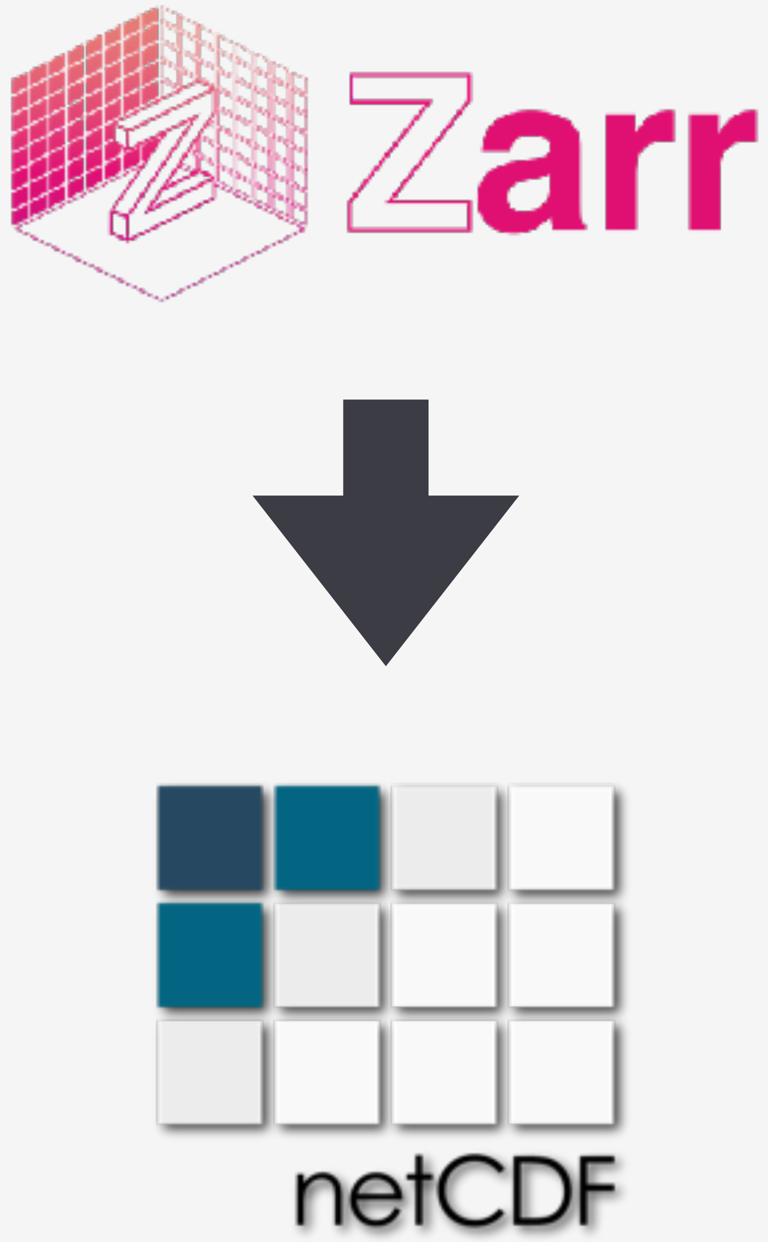
- ✓ **Nearly** all array formats store (un)compressed, chunked binary blobs
  - ✓ Chunks may just be different files (GRIB, Zarr)
  - ✓ Or internal to a single file (netCDF4/HDF5/TIFF)
  - ✓ Or both (folder of netCDF4 files)

- ✓ **Idea:** We can use Zarr to read all of the above as long as we know where the chunks are in the file, and know how to decode the compressed bytes.
  - ✓ Build a manifest that records the chunk locations (“references”) and codec, use Zarr to read everything.
  - ✓ Note potentially ~10-100 million chunks per array
  - ✓ ncML style virtual aggregations, but turbocharged



Existing Solution: Kerchunk / VirtualiZarr

→ “Virtual” Zarr datasets on top of existing archival data



✓ Pros

- Leverage the PB of existing data
- “Remix” individual granules into one logical dataset

✓ Cons

- Still limited by internal chunking of those files
- Have have to keep track of “chunk references" somehow
- Limited library support for using references

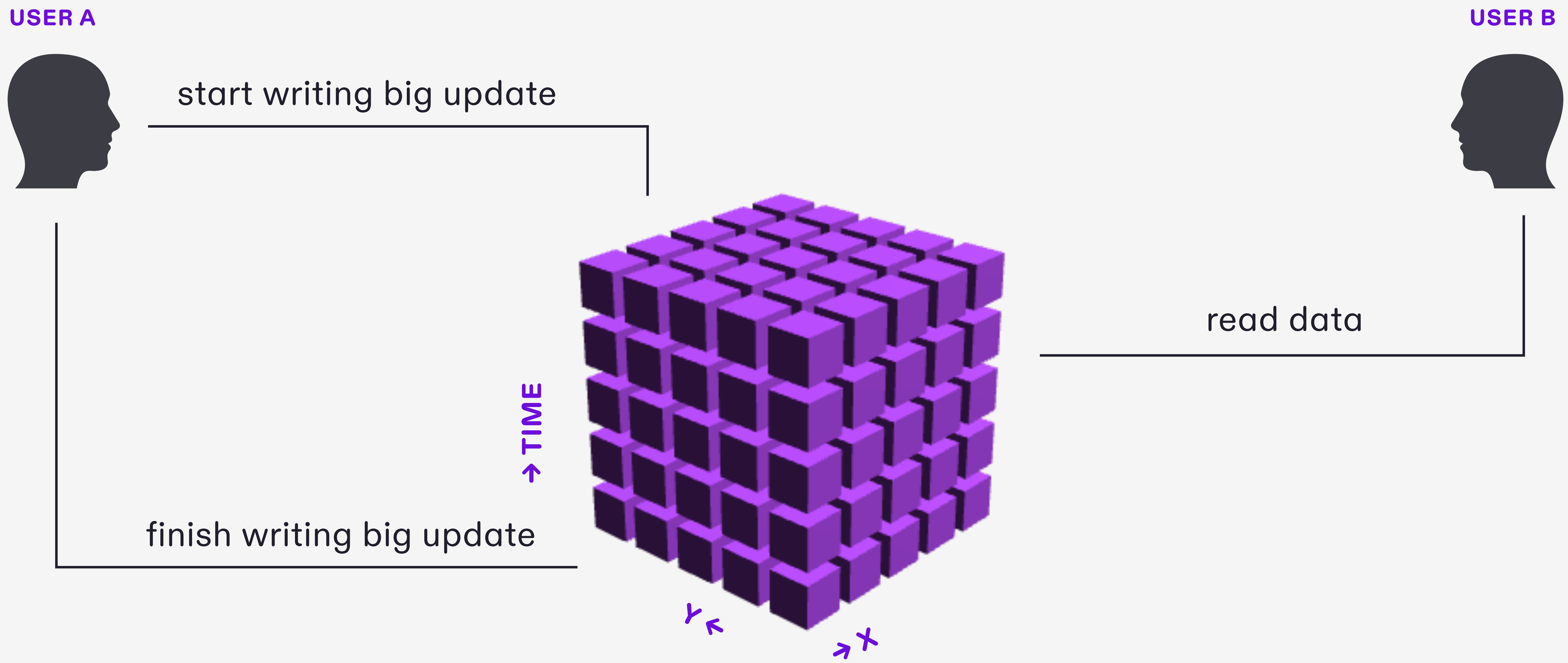
What do our clients care about?

→ Very similar to govt/academic space

- ✓ ML: perf, perf, perf, perf, perf
  - ✓ Will pre-process and maintain multiple copies of the data to enable this.
  - ✓ Scale like the object store (5000 reqs/sec)
- ✓ Data team:
  - ✓ Correctness: consistent views of the data.
  - ✓ Version control as they iterate on a dataset
- ✓ All teams:
  - ✓ Collaborative features.

Consistency issues with Zarr

→ Zarr is not designed for “multiplayer mode”



These issues stem from the fact that Zarr is not a file format.
Zarr dataset is spread over many files. (More like a database.)

Summary of challenges with the status quo

- ✓ Archival array file formats (HDF5, NetCDF, GRIB) are not designed for cloud object storage and are slow
- ✓ Exabytes of data are still being produced in archival formats
- ✓ Zarr is nice, but has some limitations
  - Data must be transformed
  - Not safe to use in multiplayer mode
  - Hard to detect and fix corrupted data
- ✓ Kerchunk / VirtualiZarr have great potential to retroactively optimize the existing archives
  - ...but no major agency has actually put this in production yet.

ICECHUNK

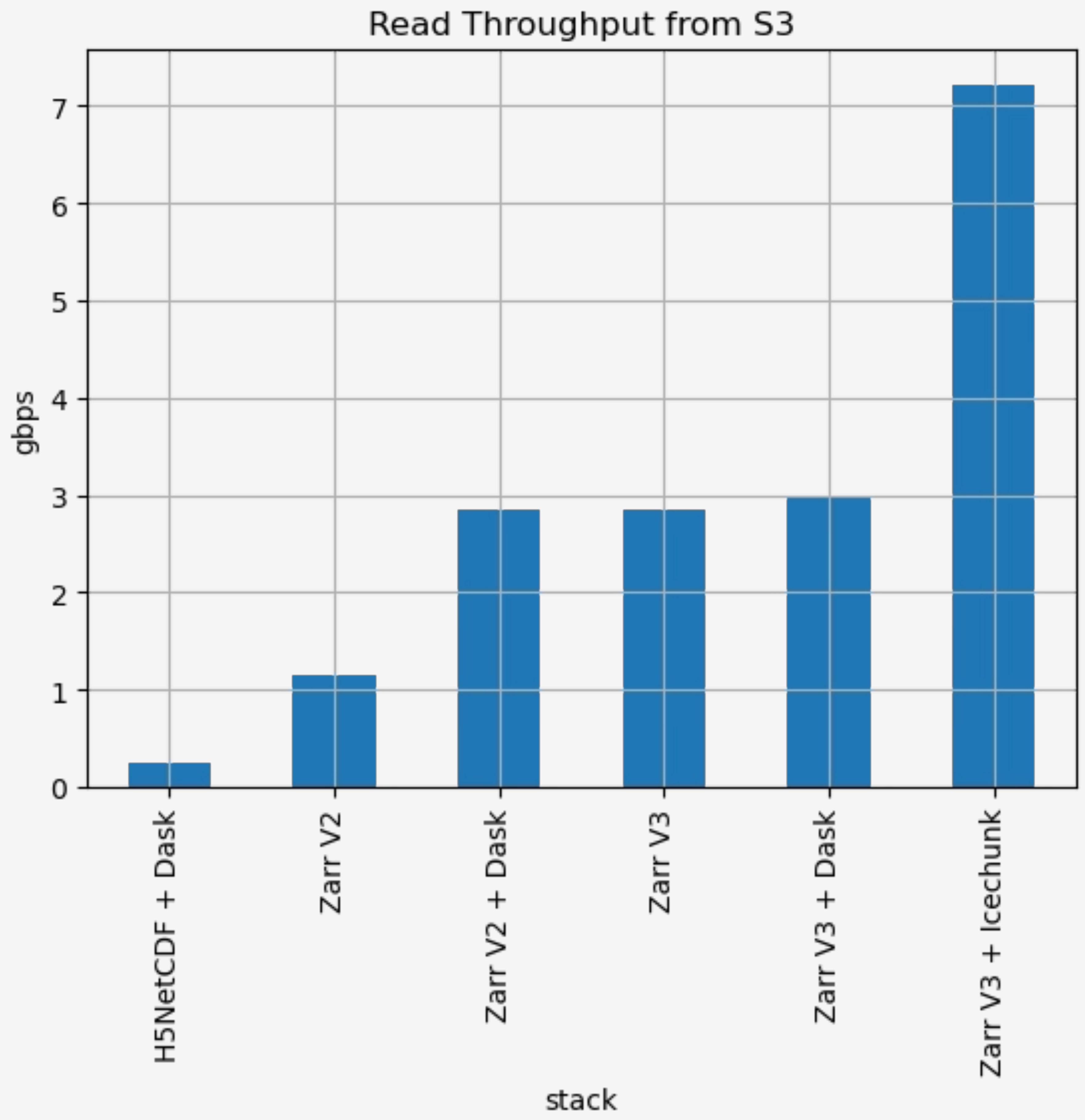
open-source, cloud-native transactional tensor storage engine

icechunk

→ icechunk.io

ICECHUNK

- ✓ **open-source,** cloud-native transactional array storage engine
  - ✓ **designed** for use on cloud object storage
  - ✓ works together with Zarr
  - ✓ Handles virtual references (the kerchunk idea)
  - ✓ implementation in Rust (with Python bindings)
    - ✓ perf, perf, perf
  - ✓ Heavily inspired by the tabular world (Apache Iceberg)
    - ✓ safe multiplayer mode: atomic updates (transactions)
    - ✓ and versioning
  - ✓ earthmover.io/blog/icechunk



Zarr data model for the cloud

→ Prompt 1

- The CF/netCDF data model can be expressed in multiple formats
    - Historically HDF5 was blessed (netCDF4)
    - But Zarr works too (ncZarr)
    - Or Zarr + icechunk

- They’re all the “same”
    - Compressed chunks as binary blobs
    - Arbitrary metadata
    - Arbitrary hierarchies of groups and arrays

the relationship between netCDF and zarr/xarray.

→ Prompt 2

- Xarray is an analysis library, it can read netCDF/Zarr/Icechunk/NCZarr/GRIB etc.
    - (Xarray’s job is to make the file format an uninteresting detail to the analyst, much like NCL)
- Xarray is commonly used to read from netCDF, stitch together a data cube, and write to Zarr
- Zarr can be netCDF if you want it to (DEMO)
    - https://app.earthmover.io/earthmover-demos/hrrr

"AI/ML" ready data and "big data" issues.

Prompt 3

- ✓ These are ill-defined terms.
- ✓ Query patterns can be quite orthogonal (for AI/ML and even generally)
  - ✓ Time series analysis: Small in space, large in time
  - ✓ Spatial (map) analysis: Large in space, small in time

- ✓ **Zarr v3/Icechunk** is our best bet yet
  - ✓ perf is a step change up.
  - ✓ Designed to scale to millions of references but more work to be done.
  - ✓ We are thinking hard

Summary

deepak@earthmover.io

- ✓ **What's old is ~~new again~~ now cloud-optimized.**
  - ✓ Chunked arrays in files now on object storage.
  - ✓ High-performance parallelized reading libraries
  - ✓ We can serve Zarr as netCDF, JSON, CSV etc. using standard APIs
    - ✓ Opendap, WMS tiles for maps, OGC EDR for position queries etc. etc.

- ✓ **Earthmover PBC is working hard to make all of this easy™**
  as a cloud SaaS platform,
  built on high-quality open-source foundation
  - ✓ Xarray, Zarr, Icechunk, Xpublish

- ✓ **What challenge are you looking to address?**

questions?

→ deepak@earthmover.io

Icechunk File Layout

All data and metadata files are stored within a root directory (typically a prefix within an object store) using the following directory structure.

- `$ROOT/refs/` reference files
- `$ROOT/snapshots/` snapshot files
- `$ROOT/attributes/` attribute files
- `$ROOT/manifests/` chunk manifests
- `$ROOT/chunks/` chunks