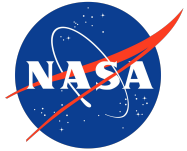
PANGEO



developmentSEED (carbon)plan

VirtualiZarr: Create virtual Zarr stores using xarray syntax

*Tom Nicholas
(and lots of help!)

[C] Worthy

*tom@cworthy.org
@TomNicholas

What I will talk about:

- Problem: Cloud-friendly access to archival data
- Solution: Zarr virtualization
- `VirtualiZarr` package
- Icechunk integration

Who am I?

xarray

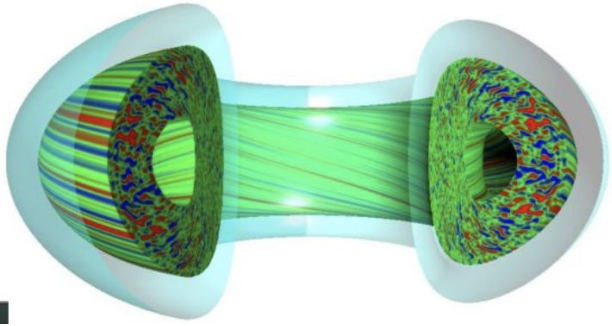
2016 — Ph.D. in Plasma Physics for Nuclear Fusion

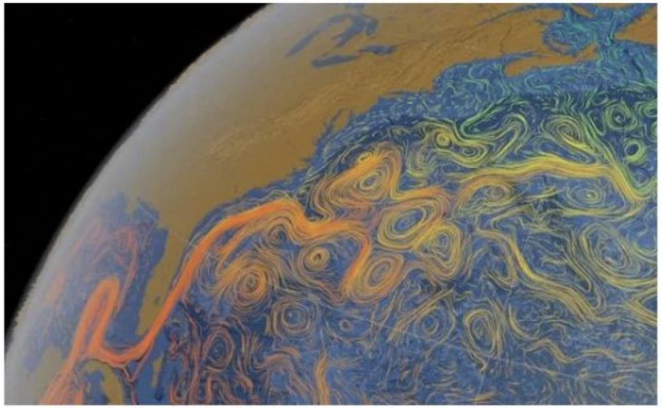
2018 — Open-source contributor (Xarray maintainer)

2021 — Pivot to oceanography at Columbia U.

2023 — Join [C]Worthy

2024 — Wrote VirtualiZarr





Zarr

Problem of accessing legacy file formats

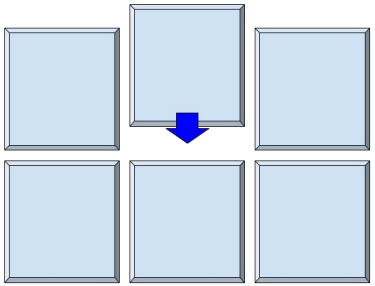
- All these old files (netCDF, HDF5, GRIB, …)
  - Want to put them on the cloud

- Zarr is a much better format
  - Separation between data and metadata
  - Scalable
  - Cloud-optimized access patterns

- But data providers **don't want to change** formats
  - Also ideally avoid data duplication

Solution: Zarr Virtualization!

kerchunk



- Create a **Zarr-like layer** over the top
  - Zarr metadata + pointers to byte ranges inside the legacy files
  - Sidecar files which sit alongside original legacy files

- In the same bucket or anywhere and point to bucket URL
  - Potential here to create catalogue-like meta-stores...

Solution: Zarr Virtualization!

Advantages:

- No data duplication (only metadata)
- Original files unchanged
- Cloud-like access can be just as performant as Zarr

Disadvantages:

- Can't change chunk size
- Can’t guarantee consistency if original files changed
- Requires mapping from legacy format to Zarr data model
  - Imposes limitations - come back to this

Kerchunk + fsspec approach

- Makes a *zarr-like* layer

- Kerchunk currently:
  - Finds byte ranges using e.g. `SingleHdf5ToZarr`
  - Represents them as a nest `dict` in-memory
  - Combines dicts using `MultiZarrToZarr`
  - Writes them out in kerchunk reference format (json/parquet)

- Result is sidecar files which behave like a zarr store…
  - … when read through `fsspec`

Issues with Kerchunk + fsspec approach

- Concatenation is complicated
  - Store-level abstractions make many operations hard to express
  - `MultiZarrToZarr` is bespoke and overloaded

- In-memory `dict` representation is complicated, bespoke, and inefficient

- Output files are not true Zarr stores,
  - Can only be understood by `fsspec` (i.e. currently only in python…?)

Challenge: CWorthy datasets

- Large (many many files / chunks)
  - So any inefficiency is painful
- Multi-dimensional
  - (So `MultiZarrToZarr` gets tricky to use along many dimensions)
- Staggered grids
  - Very hard to express correct concatenation operations with store-level operations
- Stretch goal: Variable-length chunks

Ideal approach

- Make concatenation as easy as `xr.concat`
- Efficient in-memory representation of chunks that scales well
- Output true Zarr stores that follow some Zarr spec
- Handle virtual data and “real” data equally

Structure of the problem: "Chunk Manifests"

- A zarr store is metadata + compressed chunks of bytes
- But kerchunk has shown that these bytes can live elsewhere
- Simplest representation is Zarr store with a list of paths to bytes
  - i.e. a "chunk manifest" for each array

```
{
    "0.0.0": {"path": "s3://bucket/foo1.nc", "offset": 100, "length": 100},
    "0.0.1": {"path": "s3://bucket/foo2.nc", "offset": 200, "length": 100},
    "0.1.0": {"path": "s3://bucket/foo3.nc", "offset": 300, "length": 100},
    "0.1.1": {"path": "s3://bucket/foo4.nc", "offset": 400, "length": 100},
}
```

VirtualiZarr package

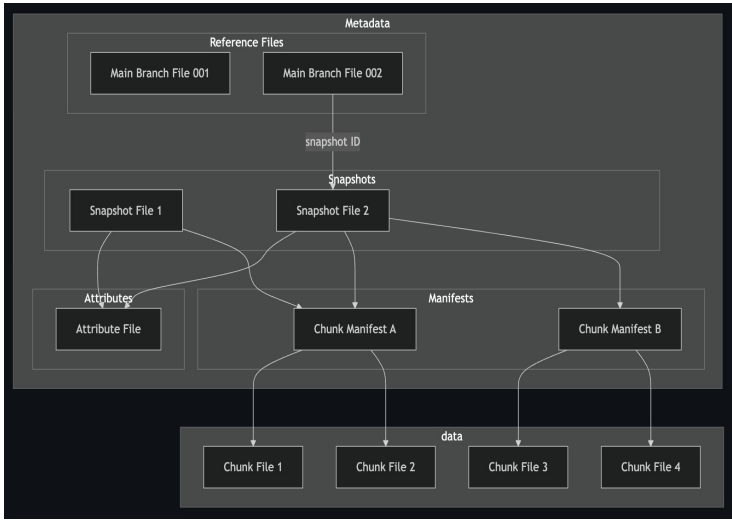
- Combining references to files == **array concatenation problem**
- We already have xarray API for concatenation
- So make an array type that xarray can wrap
- `ManifestArray`, which wraps a `ChunkManifest` object
  - Demo…

Virtual references in Icechunk

- Open-source, version-controlled zarr store
  - Basically Apache Iceberg but for arrays
  - Created by Earthmover

- Can hold virtual references…

- Commit virtual refs to netCDF files into Icechunk, tracking changes, and read any version of data as Zarr!!




```
vds.virtualize.to_icechunk(store)
ds = xr.open_zarr(store)
```

https://icechunk.io/

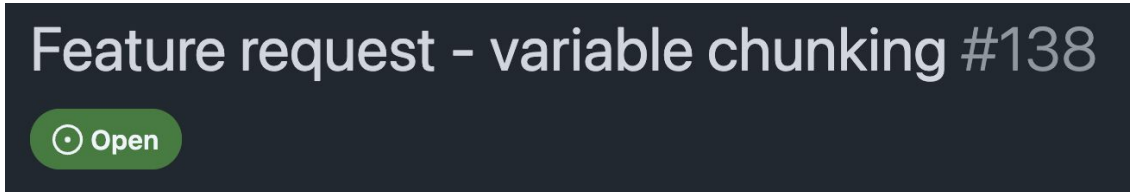
Pangeo cloud data access stack

2018
- `netCDF -> fsspec -> xr.open_mfdataset('*.nc')`
  - Slow to open and combine, slow to read
  - Or… you duplicate your entire dataset as Zarr

2021
- `netCDF -> kerchunk -> kerchunk json -> xr.open_dataset('refs.json', engine='kerchunk')`
  - Only combine once, faster to read, not duplicated, pain to use

2024
- `netCDF -> VirtualiZarr -> Icechunk -> xr.open_zarr(icechunkstore)`
  - Painless, even faster to read, and version-controlled!

Future of Zarr: Variable-length chunks



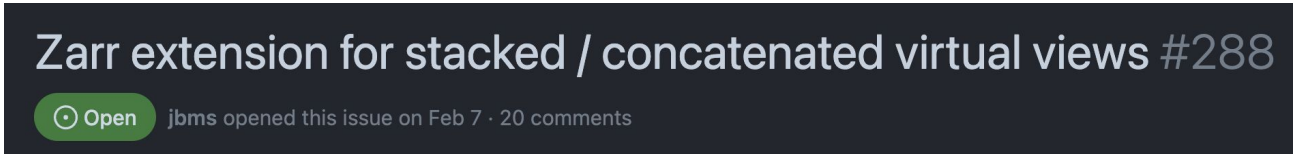
- Zarr data model currently requires all chunks in array are same length along one dimension

- Problem for common datasets
    - e.g. daily data chunked by month - {31, 28, 31, 20, …}

- Hoping for NASA funding to generalize this

developmentSEED

Future of Zarr: Virtual concatenation ZEP



- Single zarr array implies one set of codecs
  - So can't merge manifests from files with different encoding
- Need more general version of concatenation
  - Needs to also be serializable to Zarr store
- Idea: “Virtual concatenation” of zarr arrays

Conclusion

- Virtual Zarr stores over legacy data are a cool idea!

- VirtualiZarr package exists as successor to kerchunk
  - Still new but progressed rapidly
  - Uses xarray API so should be intuitive
  - Can be used today to write virtual references
    - As kerchunk format
    - Into version-controlled Icechunk stores

- Planned: upstream generalizations of Zarr's data model

Go try it! https://github.com/zarr-developers/VirtualiZarr

Bonus: Finding byte ranges

- Currently using kerchunk (e.g. `SingleHdf5ToZarr`)

- Could rewrite these readers too
  - Could give performance/reliability advantages

- Can also just convert byte ranges from other sidecar formats
  - e.g. NASA's DMR++

Bonus: Appending to existing data in icechunk

- Common to want to regularly append to existing dataset
  - e.g. forecast data, reanalysis

- Don’t want a new Zarr store every time you get new data…

developmentSEED

```
new_vds = vz.open_virtual_dataset('todays_weather.grib')

new_vds.virtualize.to_icechunk(icechunkstore,
append_dim='time')

icechunkstore.commit('dataset as of <todays-date>')
```