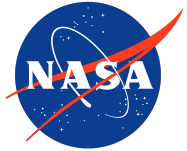
PANGEO



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
- Planned upstream Zarr enhancements

Problem of accessing legacy file formats

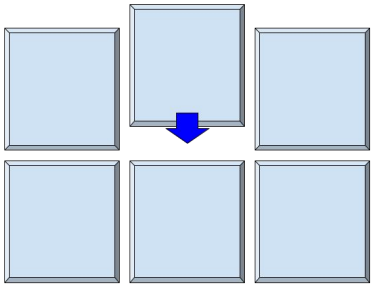
- All these old files (netCDF, HDF5, GRIB, …)
  - Want to put them on the cloud

- 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
  - 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

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
- Variable-length chunks

[C] Worthy

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

Structure of the problem: "Chunk Manifests"

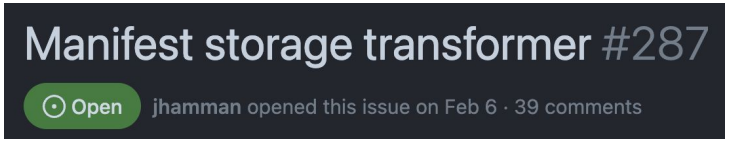
- A zarr store is metadata + compressed chunks of bytes
- But kerchunk has shown that these bytes can live elsewhere
- Simplest representation is Zarr store with a list of paths to bytes
  - i.e. a "chunk manifest" for each array

```
{
    "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…

Future of Zarr: Chunk manifests ZEP



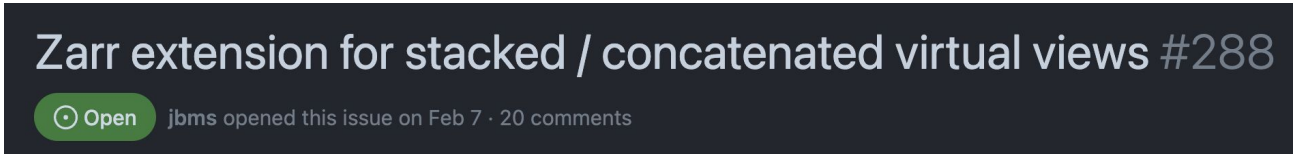
- New type of Zarr store containing chunk `manifest.json` files

- Formalize via a ZEP, then implement reading arbitrary byte ranges in zarr readers

```
a/zarr.json  <- group metadata
a/foo/zarr.json  <- array metadata
a/foo/manifest.json <- array manifest
...
b/baz/zarr.json <- array metadata
b/baz/c/1/1 <- "regular" chunk
...
```

- Means virtual zarr stores that can be read in any language
  - Opens the door to e.g. javascript visualization frameworks pointing at netCDF files…

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 alternative to kerchunk
  - Some rough edges but progressing quickly
  - Can be used today to write kerchunk-format references
  - Uses xarray API so should be intuitive

- Plan is to upstream sidecar formats as Zarr enhancements

Go try it! https://github.com/TomNicholas/VirtualiZarr

Bonus: Finding byte ranges

- Currently using kerchunk (e.g. `SingleHdf5ToZarr`)
- Could rewrite these readers too
  - Could give performance advantages
- Parallelize opening like `xr.open_mfdataset(..., parallel=True)`
- Could also read from other sidecar formats
  - E.g. NASA's DMR++