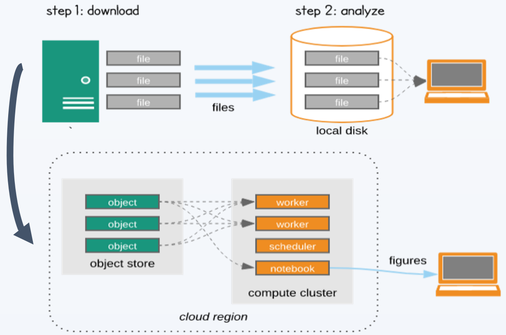
Introducing Kerchunk: Cloud-performant reading of NetCDF4/HDF5/Grib2 using the Zarr library

Lucas Sterzinger[1], Martin Durant[2], Rich Signell[3], Chelle Gentemann[4], Julia Kent[5], Kevin Paul[5]

[1]UC Davis; [2]Anaconda Inc; [3]USGS; [4]Farallon Institute; [5]NCAR

Paradigm Shift

- As datasets get larger and larger, downloading data for local analysis is less feasible
- Cloud computing allows for computation to happen adjacent to permanent data storage within the same cloud region

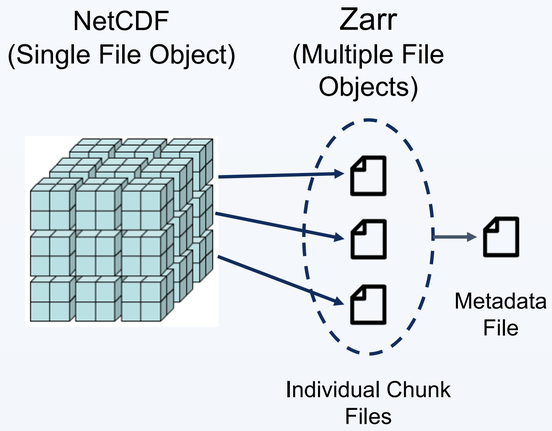


Not all data are created equal

- Many institutions are moving their data to the cloud
- Most of this data is still in its archival form
  - NetCDF/HDF/GRIB/FITS
- While these formats enjoy support from a wide range of tools and programming languages, they are not optimized for the cloud
- Zarr data format is focused on optimizing large datasets for cloud access
- But for data repositories already in the cloud, **offering Zarr requires conversion and data duplication**

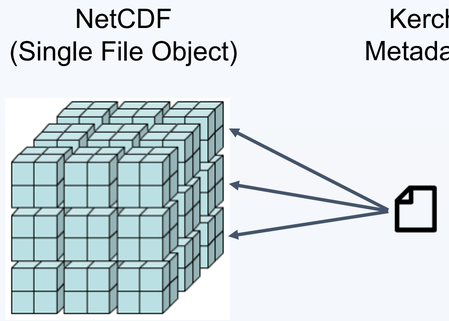
Zarr

Zarr works by splitting chunks of data into individual file objects, which are then described by a plaintext metadata JSON.



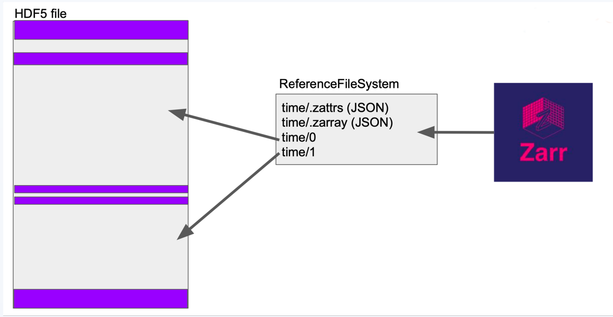
Kerchunk

Kerchunk works by using the same Zarr metadata format, but instead of pointing to individual chunk file objects the metadata points to byte-ranges within the original data file



ReferenceFileSystem

Behind the scenes, accessing Kerchunked data relies on fsspec's ReferenceFileSystem, creating a virtual filesystem in which the Zarr API can navigate, and maps would-be Zarr chunk files to the byte ranges generated by Kerchunk



The following table compares accessing 24 hours of GOES-16 satellite data on AWS and performing a workflow (code linked here; generating a movie, RGB composites, timeseries and spatial gradients) between the native NetCDF4 data on the cloud, converting to Zarr and uploading to the cloud, and using reference files generated with Kerchunk.

Accessing the native NetCDF was a slow process. Zarr was fast, but at the expense of conversion and storage needed to host the new files (52 GB, the size of the original NetCDF4 files). Kerchunk had near the same benchmark times as native Zarr, but with the benefit of *only generating 416 MB of additional files.*

Format	Preprocess	Open w/ Xarray	Workflow	Extra Storage
NetCDF4	0 min	10 min	40 min	0 GB
Zarr	1 h 38 min	30 sec	4 min 10 sec	52 GB
Kerchunk	1 h 25 min	35 sec	4 min 30 sec	**416 MB**

Click this link (or scan the QR code) for links to Kerchunk, tutorials, code samples, and more