EOSDIS Zarr Store - Fast, direct access to NASA EOSDIS data in the cloud

2020 ESIP Summer Meeting

Patrick Quinn
EED-2 Software Engineer
patrick@element84.com

This work was supported by NASA/GSFC under Raytheon Technologies contract number NNG15HZ39C. This document does not contain technology or Technical Data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.
Acronym Soup

- EOSDIS – Earth Observing System Data and Information System. The collective systems that, among other responsibilities, archive and distribute NASA Earth observation data
- HDF5 – Hierarchical Data Format, version 5. A standard hierarchical multidimensional data format
- NetCDF4 – Network Common Data Form, version 4. A standard data format built on top of HDF5
- OPeNDAP DMR++ - Open-source Project for a Network Data Access Protocol Dataset Metadata Response. A standardized metadata response describing the contents of a data file as produced by the Hyrax data server
EOSDIS Zarr Store

Cloud Optimized Format Study
Cloud Optimized Format Study

- **Goal** – Qualitatively and quantitatively examine the pros and cons of different formats for storing data on the cloud
- **Approach** – Compare data formats which were optimized for the cloud to the existing standards which store the majority of current NASA EOSDIS data

<table>
<thead>
<tr>
<th>Cloud-Optimized Data Formats</th>
<th>Currently Used Data Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud-Optimized GeoTIFF</td>
<td>GeoTIFF</td>
</tr>
<tr>
<td>HDF in the Cloud</td>
<td>NetCDF-4</td>
</tr>
<tr>
<td>Parquet</td>
<td></td>
</tr>
<tr>
<td>Zarr</td>
<td></td>
</tr>
</tbody>
</table>

Study available at: [https://ntrs.nasa.gov/search.jsp?R=20200001178](https://ntrs.nasa.gov/search.jsp?R=20200001178)
Evaluation Criteria

- Data access **performance** to support common forms of analysis, including time series, shape-based averaging, regridding and data intercomparison.
- **Compatibility** with existing off the shelf tools, including Panoply, gdal, nco, Jupyter/xarray, ArcGIS and QGIS.
- Ability to support **fine-grained requests** from S3 via range-get or other means.
- Ability to comply with community **metadata conventions**
- Availability of **independent libraries** to read the data in C/C++, Fortran, Python and R
- Comparative **cost** of data preparation, storage and analysis, adjusted for lossless compressibility as appropriate.
- Ability to **represent several different data types** / structures including imagery, swath, trajectory, point cloud, Platte-Carre and Sinusoidal grids, in situ and airborne
- Ability to verify data **integrity** upon reformatting and ongoing
- **Self-describability**, i.e., ability to include complete sets of both descriptive and structural metadata
- **Open specification**
- Number of **independent implementations** of read/write API
- **Standards-body approval**

Study available at: [https://ntrs.nasa.gov/search.jsp?R=20200001178](https://ntrs.nasa.gov/search.jsp?R=20200001178)
Sample Performance Results

Percentage of File Size Read - MOD11A2 (Tile Grid)

Benchmark Execution Time - 3IMERGHH (Global Grid)

Study available at: https://ntrs.nasa.gov/search.jsp?R=20200001178
High Level Conclusions

• Found no one-size-fits-all solution but provided some broad guidelines to inform choices
• Regardless of selected format and compression algorithm, optimize files for partial access over HTTP using the Range header
• Choose Cloud Optimized GeoTIFF over standard GeoTIFF when a TIFF is desired
• Recognized the need for the stability, interoperability and self-describing nature of NetCDF-4, but wanted the performance of zarr

Study available at: https://ntrs.nasa.gov/search.jsp?R=20200001178
EOSDIS Zarr Store

Accessing NetCDF via zarr-python
We provide this

Typical NetCDF4 Data File

Variable A

Variable B

Variable C

Variable D

Coastline Image © OpenStreetMap contributors
We provide this

Typical NetCDF4 Data File

Variable A

Variable B

Variable C

Variable D
We provide this

I want this

To get it, libraries need this

Typical NetCDF4 Data File

File Metadata

Dimension 1 Metadata

Dimension 1 Data

Dimension 2 Metadata

Dimension 2 Data

Variable A

Variable A Metadata

Variable A Data

Variable B

Variable B Metadata

Variable B Data

Variable C

Variable C Metadata

Variable D

Variable D Metadata
We provide this

I want this

To get it, libraries need this

Typical NetCDF4 Data File

File Metadata

Variable A Metadata

Variable A Data

Variable B Metadata

Variable B Data

Dimension 1 Metadata

Dimension 1 Data

Dimension 2 Metadata

Dimension 2 Data

The Challenge:
We need a small percentage of the file, but we don’t know where to find the bytes we need without reading a significant percentage of the file.
The Challenge:
We need a small percentage of the file, but we don’t know where to find the bytes we need without reading a significant percentage of the file.
<table>
<thead>
<tr>
<th>Zarr Metadata</th>
<th>Variable A Metadata</th>
<th>Variable B Metadata</th>
<th>Variable C Metadata</th>
<th>Variable D Metadata</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Metadata</td>
<td>Dimension 1 Metadata</td>
<td>Dimension 1 Metadata</td>
<td>Dimension 2 Metadata</td>
<td>Dimension 2 Metadata</td>
</tr>
<tr>
<td></td>
<td>Byte Offsets of Chunks of Data (USGS + The HDF Group Prototype)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Library updates:**

1. Read all up front in a single, fast read.
2. Directly read only data of interest

---

**Typical NetCDF4 Data File**

<table>
<thead>
<tr>
<th>File Metadata</th>
<th>Dimension 1 Metadata</th>
<th>Dimension 1 Data</th>
<th>Dimension 2 Metadata</th>
<th>Dimension 2 Data</th>
</tr>
</thead>
</table>

**Variable A**

<table>
<thead>
<tr>
<th>Variable A Metadata</th>
<th>Variable A Data</th>
</tr>
</thead>
</table>

**Variable B**

<table>
<thead>
<tr>
<th>Variable B Metadata</th>
<th>Variable B Data</th>
</tr>
</thead>
</table>

**The Challenge:**

We need a small percentage of the file, but we don’t know where to find the bytes we need without reading a significant percentage of the file.
```python
%%time
ncfile = fsspec.open('s3://pangeo-data-uswest2/esip/adcirc/adcirc_01d.nc',
anon=False, requester_pays=True)

CPU times: user 70 µs, sys: 8 µs, total: 78 µs
Wall time: 79.2 µs

%%time
store = fsspec.get_mapper('s3://hdf5-zarr/adcirc_01d.nc.chunkstore', anon=True)
chunk_store = FileChunkStore(store, chunk_source=ncfile.open())

ds2 = xr.open_zarr(store, consolidated=True, chunk_store=chunk_store)

CPU times: user 100 ms, sys: 20.4 ms, total: 121 ms
Wall time: 341 ms

%%time
max_var2 = ds2['zeta'].max(dim='time').compute()

CPU times: user 11 s, sys: 1.01 s, total: 12 s
Wall time: 22.1 s
```

More details and examples

“Cloud-Performant NetCDF4 / HDF5 Reading with the Zarr Library”
by Rich Signell (USGS), Aleksandar Jelenak (The HDF Group), and John Readey (The HDF Group)

https://medium.com/pangeo/cloud-performant-reading-of-netcdf4-hdf5-data-using-the-zarr-library-1a95c5c92314

(Or search “medium netcdf zarr”)
Optimizations to EOSDIS Access
The Challenge:
We need a small percentage of the file, but we don’t know where to find the bytes we need without reading a significant percentage of the file.
The Challenge:
We need a small percentage of the file, but we don’t know where to find the bytes we need without reading a significant percentage of the file.
The Challenge:
We need a small percentage of the file, but we don’t know where to find the bytes we need without reading a significant percentage of the file.
Optimization 1: Allow Reading EOSDIS NetCDF from Zarr Python

EOSDIS Cloud

EOSDIS Zarr Storage Library (New)

Zarr Python Library

OPeNDAP DMR++

Typical EOSDIS Data File
Zarr Read Data
Optimization 2: Cache Handshaking Information

By default, for every request, often hundreds or more...
Optimization 2: Cache Handshaking Information

By default, for every request, often hundreds or more
Zarr Read Data
Optimization 3: Combine Requests for Near-Adjacent Ranges

Zarr Read Data
Optimization 4: Parallelize once we have an S3 URL

Zarr Read Data
Optimization 5: Cache outputs for future reads

Zarr Read Data
EOSDIS Zarr Store

Examples
EOSDIS Zarr Store

Conclusions
Conclusions

Cloud access optimization demands read latency optimization

1. Combine many smaller reads into fewer larger reads (metadata & data)
2. Skip expensive network actions such as auth or redirects, if possible, via cookies and caching
3. Parallelize reads
Questions?

Patrick Quinn
EED-2 Software Engineer
patrick@element84.com
This work was supported by NASA/GSFC under Raytheon Technologies contract number NNG15HZ39C.

in partnership with