Gridded Environmental Data in the Cloud: NCEI Data Access Perspective

Mark Capece[1,2]
[1]General Dynamics Information Technology, Falls Church, VA, USA; [2]Data Access Branch, Data Stewardship Division, NCEI, Asheville, NC, USA

Introduction

The development of affordable cloud storage has revolutionized data storage and distribution, offering enhanced durability and nearly unlimited capacity. To ensure that the NCEI's archives can continue to scale to meet the demands of data providers and consumers, NESDIS has initiated an effort to migrate a portion of its mission capabilities to the cloud in the next five years. The underlying infrastructure, including the foundations for end-to-end data ingest and dissemination in the AWS cloud, is being developed through the NESDIS Common Cloud Framework (NCCF) project.[1]

NCEI's Data Access Branch development team is responsible for the Distribution and Access component of the NCCF, ensuring that data is properly stored for efficient search, discovery, and retrieval. Their mission is to meet the following access requirements for gridded data in AWS S3.

1. Aggregation and sub-setting with OPeNDAP-like services[2]
2. Usage of cloud-optimized data formats
3. Minimizing egress from the cloud
4. Monitoring how the data is being accessed

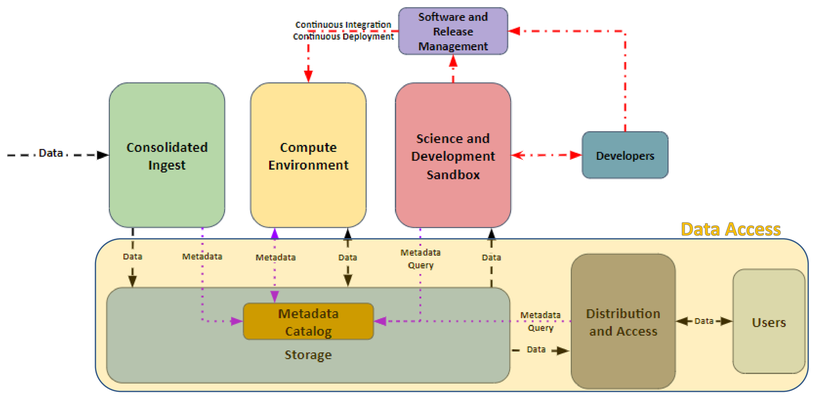

Fig 1. Architecture of the NESDIS Common Cloud Framework

Cloud-Ready Data Formats

- Gridded data consists of multidimensional arrays with geospatial coordinates.
- Marine, satellite, and model datasets are often gridded.
- Gridded data is traditionally constructed as GRIB, NetCDF, and HDF files.
- Cloud readiness entails organizing data arrays into smaller pieces that can be accessed individually in parallel computing applications.
- HDF5 and NetCDF4 organizes arrays into chunks within the files themselves.
- Zarr is a standout cloud-ready data format that stores chunks as separate objects in S3, enabling rapid, parallel read/write operations for accelerated data analysis.[3,4]
- Zarr support is being added to the NetCDF library.[5,6]
- Zarr is currently under review as a Community Standard through the Open Geospatial Consortium.[7]
- GRIB, NetCDF, and HDF files can be converted to Zarr using the Xarray library.
- Zarr arrays can expand to include data from multiple GRIB, NetCDF, or HDF files.

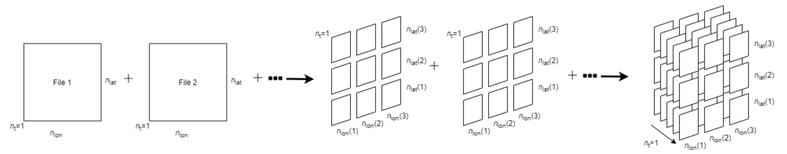

Fig 2. Converting NetCDF to Zarr divides files into smaller chunks and aggregates arrays along a temporal dimension

OPeNDAP Servers

THREDDS, ERDDAP, and Hyrax are well-established OPeNDAP Java applications used on-premises at NCEI. ZarrDAP is a Python Flask application developed in-house at NCEI to handle OPeNDAP requests for Zarr-formatted datasets in S3. Each application was deployed in Amazon Elastic Kubernetes Service.

1. Catalogs and configurations are stored in CodeCommit repositories.
2. Docker images are built by injecting these catalogs and configurations into base images supplied by the application developers and stored in Elastic Container Registry.
3. A Kubernetes cluster is deployed in Elastic Kubernetes Service.
4. Application containers are deployed as autoscaling Kubernetes pods in Fargate profiles.
5. S3 always hosts the authoritative data copy from which these applications serve.
6. For Hyrax only, DMR++ metadata records must be persisted on a shared Elastic File System volume, because Hyrax lacks a version-controllable catalog file like THREDDS and ERDDAP.
7. Application load balancers distribute user traffic across parallel instances. Network load balancers expose the applications outside of the Virtual Private Cloud network.
8. To an external user, applications operate the same as on-premises.
9. To monitor usage, application logs are shipped to CloudWatch.

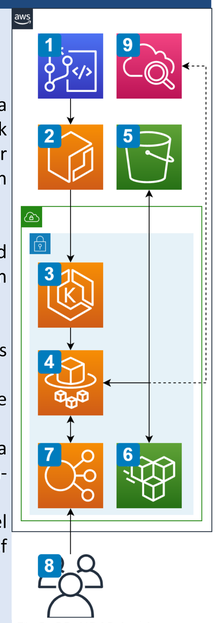

Fig 3. OPeNDAP Architecture

Tests with a Python OPeNDAP client have indicated that ERDDAP consistently outperforms other OPeNDAP servers for both initial DAS/DDS reads and DODS binary downloads for NetCDF4 files from S3, while THREDDS had the worst performance.

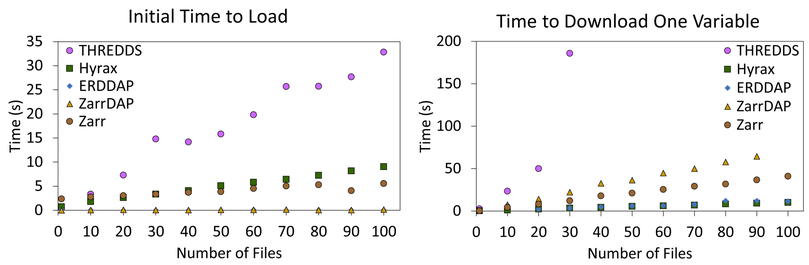

Fig 4. OPeNDAP server performance in EKS for a NetCDF4 dataset. "0 s" points were recorded at <200 ms.

Tbl 1. Qualitative comparison of THREDDS, ERDDAP, Hyrax, and ZarrDAP servers

Server	Pros	Cons
THREDDS	• Works with NetCDF3 and NetCDF4 in S3 • OPeNDAP, WMS, and NcML services	• No S3 bucket scan • Poor aggregation performance
ERDDAP	• Works with NetCDF3 and NetCDF4 in S3 • OPeNDAP, WMS, NCSS, and NcML services • Uses other OPeNDAP servers as sources • Excellent aggregation performance	• No S3 bucket scan
Hyrax	• Works with NetCDF4 in S3 • OPeNDAP, WMS, NCSS, and NcML services • Can emulate S3 bucket scan	• Does not work with NetCDF3 in S3 • Requires management of DMR++ files in EFS
ZarrDAP	• Works with NetCDF3, NetCDF4, and Zarr • OPeNDAP service • Can emulate S3 bucket scan	• No WMS, NCSS, or NcML services • No on-the-fly NetCDF aggregations • Requires Zarr to be pre-generated

JupyterHub

- Users download approximately 1 PB of data per month from NCEI.
- 1 PB of data egress from AWS to the internet would cost $55,000/month.
- Users should be encouraged to work within AWS to reduce data egress.
- DaskHub is a Dask-integrated JupyterHub cluster in Kubernetes.[8]
- JupyterHub is a sandbox environment with pre-populated credentials and libraries that supports Python, R, and Julia.
- Downloads to Jupyter notebook servers in EKS do not incur egress costs.
- Dask clusters accelerate data analysis of Zarr stores in S3.

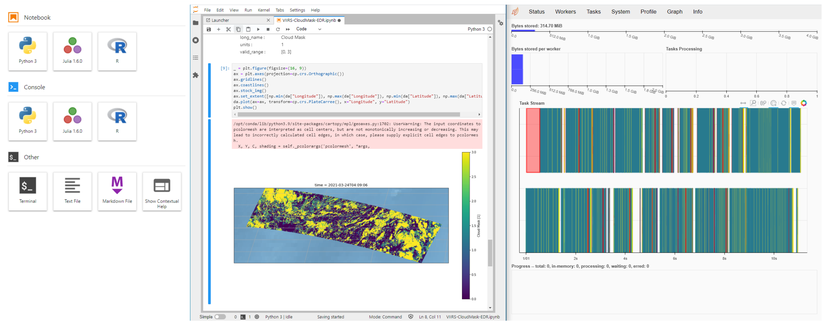

Fig 5. (Left) Jupyter notebook Launcher options. (Center/Right) Side-by-side Jupyter notebook with Dask dashboard in EKS

Conclusions

- Conventional gridded data formats should be converted to cloud-ready formats like Zarr in order to take advantage of parallel computing.
- ERDDAP and ZarrDAP were the most promising OPeNDAP servers for data stored in S3.
- Hyrax's use of DMR++ records, while good for performance, is unattractive for server administration and management.
- CPU and memory limitations of Fargate containers in EKS may bottleneck OPeNDAP server performance.
- Deploying to more powerful EC2 nodes instead of Fargate can provide more CPU and memory resources, which may improve OPeNDAP server performance.
- Traditional OPeNDAP servers may be less efficient tools for data dissemination than Zarr in S3, especially when used with Dask.
- Jupyter notebook servers may not only help educate users on how to work with data, but also reduce the costs of storage in the cloud.
- Future work will focus on subscriptions and ordering from a cloud archive, as well as exploring the implications of Zarr support in the NetCDF library.

References

1. Dalal, M.; Kent, J.. "Himawari-8: Enabling access to key weather data." *AWS Public Sector Blog*, 27 April 2020, https://aws.amazon.com/blogs/publicsector/himawari-8-enabling-access-key-weather-data/
2. "About OPeNDAP." *OPeNDAP*, https://www.opendap.org/
3. Abernathy, R.P.; Hamman, J.; Miles, A. Beyond netCDF: Cloud native climate data with zarr and xarray. In: American Geophysical Union, Fall Meeting 2018; December 2018.
4. Signell, R.P.; Pothina, D. Analysis and visualization of coastal ocean model data in the cloud. *J. Mar. Sci. Eng.* 2019, 7: 110.
5. Fisher, W.; Heimbigner, D. NetCDF in the Cloud: modernizing storage options for the netCDF data model with zarr. In: 22nd EGU General Assembly; May 2018.
6. "NetCDF 4.8.0." *News@Unidata*, 2 April 2021, https://www.unidata.ucar.edu/blogs/news/entry/netcdf-4-8-0
7. "Public Comment sought on draft Zarr Storage Specification 2.0 OGC Community Standard." *OGC Press Release*, 29 June 2021, https://www.ogc.org/pressroom/pressreleases/4497
8. Augspurger, T. "Announcing the DaskHub Helm Chart." *Dask Working Notes*, 31 August 2018, https://blog.dask.org/2020/08/31/helm_daskhub