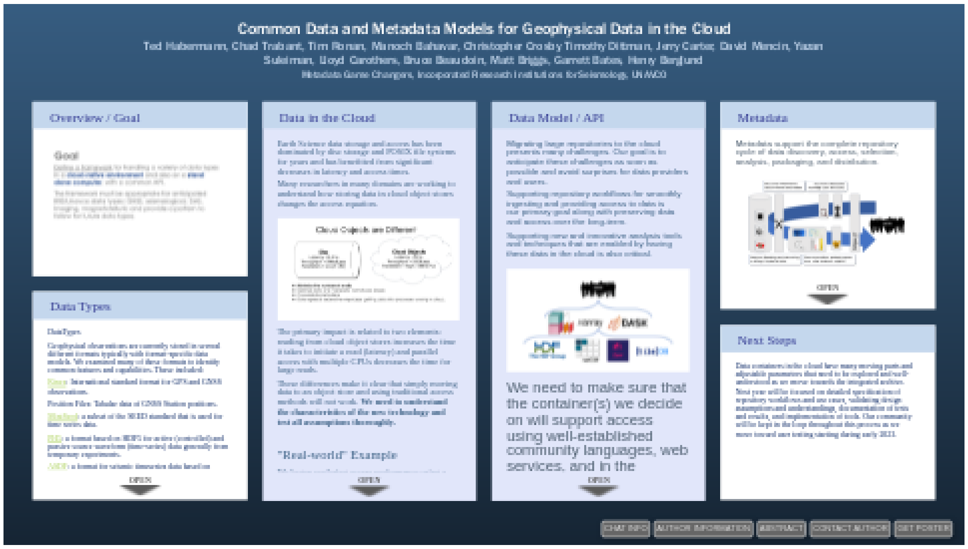
Common Data and Metadata Models for Geophysical Data in the Cloud



Ted Habermann, Chad Trabant, Tim Ronan, Manoch Bahavar, Christopher Crosby, Timothy Dittman, Jerry Carter, David Mencin, Yazan Suleiman, Lloyd Carothers, Bruce Beaudoin, Matt Briggs, Garrett Bates, Henry Berglund

Metadata Game Changers, Incorporated Research Institutions for Seismology, UNAVCO

PRESENTED AT:

OVERVIEW / GOAL

Goal

Define a framework for handling a variety of data types in a **cloud-native environment** and also on a **stand alone computer**, with a common API.

The framework must be appropriate for anticipated IRIS/Unavco data types: GNSS, seismological, DAS, imaging, magnetotelluric and provide a pattern to follow for future data types.

DATA TYPES

DataTypes

Geophysical observations are currently stored in several different formats typically with format-specific data models. We examined many of these formats to identify common features and capabilities. These included:

Rinex (https://github.com/geospace-code/georinex): International standard format for GPS and GNSS observations.

Position Files: Tabular data of GNSS Station positions.

MiniSeed (http://ds.iris.edu/ds/nodes/dmc/data/formats/miniseed/): a subset of the SEED standard that is used for time series data.

PH5 (http://ds.iris.edu/ds/nodes/dmc/data/types/ph5/?EXP): a format based on HDF5 for active (controlled) and passive source waveform (time-series) data generally from temporary experiments.

ASDF (https://readthedocs.org/projects/asdf-definition/downloads/pdf/latest/): a format for seismic timeseries data based on HDF5 and several seismic metadata formats (QuakeML and SEIS-PROV).

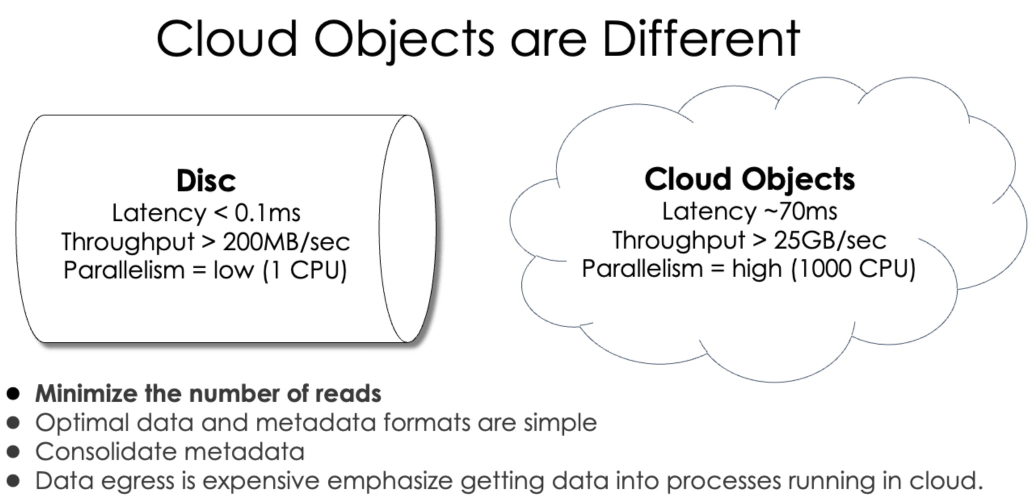
DAS:

MTH5 (https://mth5.readthedocs.io/en/latest/): an HDF5 data container for magnetotelluric time series data.

DATA IN THE CLOUD

Earth Science data storage and access has been dominated by disc storage and POSIX file systems for years and has benefitted from significant decreases in latency and access times.

Many researchers in many domains are working to understand how storing data in cloud object stores changes the access equation.



The primary impact is related to two elements: reading from cloud object stores increases the time it takes to initiate a read (latency) and parallel access with multiple CPUs decreases the time for large reads.

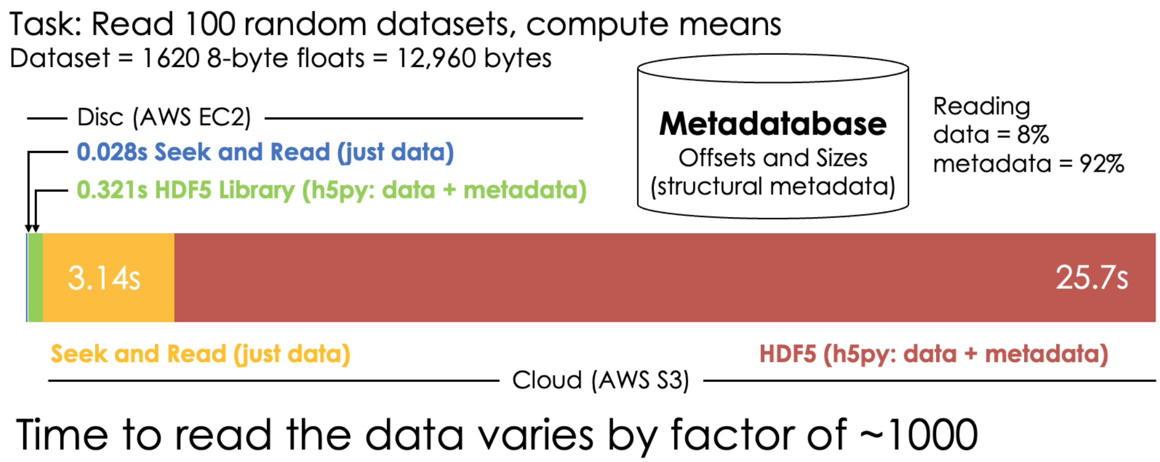
These differences make it clear that simply moving data to an object store and using traditional access methods will not work. **We need to understand the characteristics of the new technology and test all assumptions thoroughly.**

"Real-world" Example

We began exploring access performance using a simple HDF5 file with ~1.2 million seismograms from seismic events and ambient noise (Magrini et al., 2020 (https://www.sciencedirect.com/science/article/pii/S2666544120300010#appsec1)).

We read the data and metadata two ways: 1) both from the HDF file, using the HDF library and 2) read metadata (offsets and sizes of datasets in the file) from a database and data from the file using seek and read, i.e. without the HDF5 library. In this case the dataset offsets and sizes were determined using Kerchunk (https://github.com/fsspec/kerchunk) and stored outside of the object store in a database.

Both approaches were done with the data file 1) on a disc attached to an EC2 instance and 2) in an AWS object store.



H5py/HDF (metadata and data)

Disc: 0.321s

Object Store: 25.7s

Seek/Read (data), metadata in database

Disc: 0.028s

Object Store 3.14s

The fasted access (0.028s) was directly seeking to and reading just the data (i.e. no metadata) from files without the HDF5 library. The slowest access (25.7s) was reading metadata and data from an object using the HDF5 library (i.e. no optimization).

This simple comparison demonstrates that reading metadata that is distributed in multiple locations in a file, i.e requiring multiple reads, is slow because of the latency associated with each object store read.

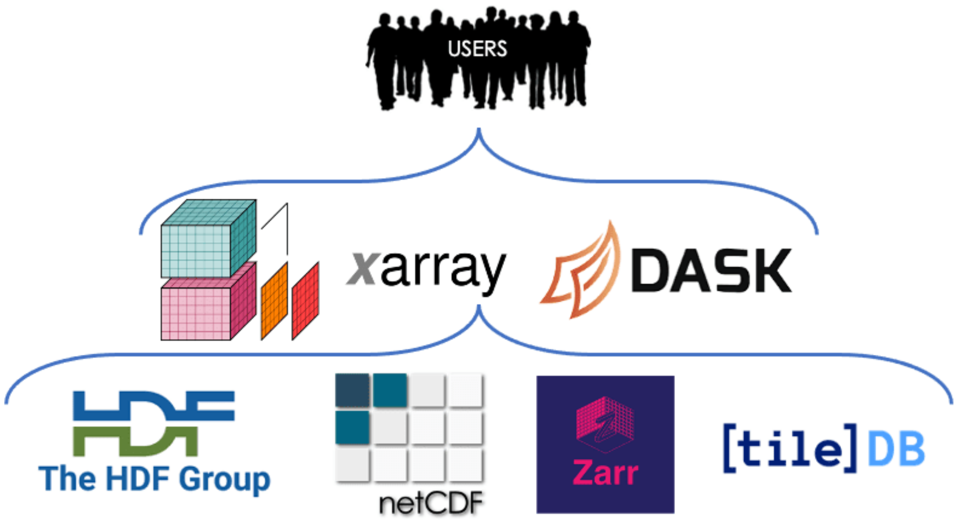
This observation has been reported by many others exploring data in the cloud and is the basis for metadata consolidation as a technique for improving performance in object stores. It also suggests that any operation that involves multiple reads must be done in parallel, i.e. with multiple workers, to minimize the effect of the latency.

DATA MODEL / API

Migrating large repositories to the cloud presents many challenges. Our goal is to anticipate these challenges as soon as possible and avoid surprises for data providers and users.

Supporting repository workflows for smoothly ingesting and providing access to data is our primary goal along with preserving data and access over the long-term.

Supporting new and innovative analysis tools and techniques that are enabled by having these data in the cloud is also critical.

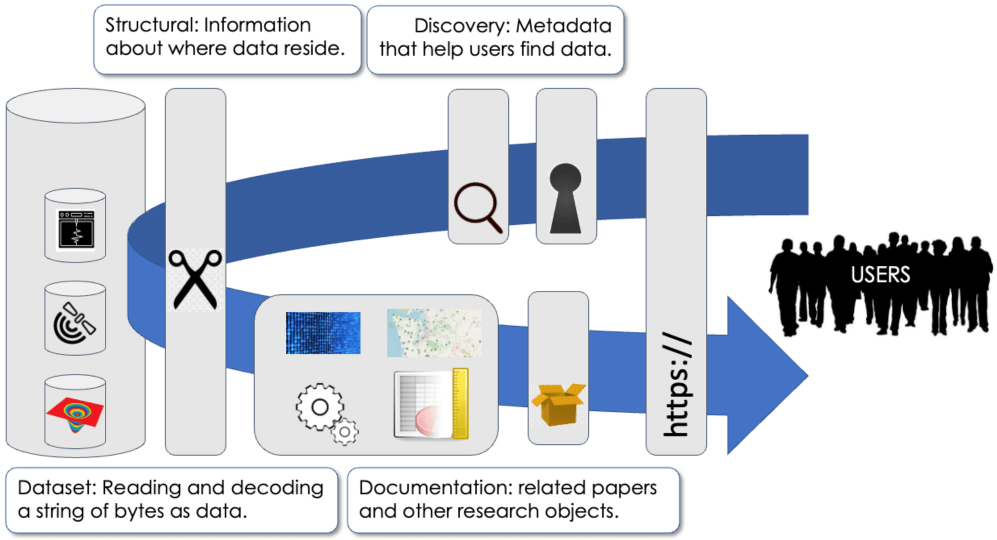


We need to make sure that the container(s) we decide on will support access using well-established community languages, web services, and in the future, direct interfaces.

We are exploring several data containers with cloud-optimization and standard user APIs in mind. These include HDF5, netCDF, and Zarr (possibly with Kerchunk for data indexing) as well as TileDB.

METADATA

Metadata support the complete repository cycle of data discovery, access, selection, analysis, packaging, and distribution.



Access to metadata embedded in cloud objects can be slow so alternative metadata architectures need to be identified and tested.

The Zarr format addresses this problem by extracting metadata and consolidating it into structured json files that are stored with the data in the cloud. These files allow reading all of the metadata to be read in a single read. This is typically done independently from reading the data to support lazy access to the data needed for analysis or distribution.

Kerchunk extracts structural (offsets and sizes of data blocks) and dataset (data type, byte-order, dimensions, ...) from HDF5/netCDF4 files. These metadata can be stored in databases and used to access the data without the HDF5/netCDF4 libraries.

More metadata details:

Discovery: Metadata that help users find data includes locations, times, station identifiers, and values for various search facets (instrument type, data type, …). These metadata are stored separately from the data, typically in a database, with an interface for user discovery and selection.

Structural: Information about where data reside, typically a path or object identifier, an offset to the data from the beginning of the object or file (could be zero), and a size of the data in bytes. They are used to access data that has been selected by users using other metadata and, therefore, act as an index for the archive data.

Dataset: Reading and properly decodong a string of bytes as data requires metadata that gives the datatype, byte-order, size, dimensionality and ordering of the dataset. In some cases, other information is required, for example, compression methods, fill_values, or other format-specific metadata.

Documentation: All information required to make data usable, understandable, trustworthy, and reproducible.

NEXT STEPS

Data containers in the cloud have many moving parts and adjustable parameters that need to be explored and well-understood as we move towards the integrated archive. Next year will be focused on detailed specification of repository workflows and use cases, validating design assumptions and understandings, documentation of tests and results, and implementation of tools. Our community will be kept in the loop throughout this process as we move toward user testing starting during early 2023.

AUTHOR INFORMATION

Ted Habermann, ted@metadatagamechangers.com[1],

Chad Trabant [2], Tim Ronan[2], Manoch Bahavar[2], Christopher Crosby[3], Timothy Dittman[3], Jerry Carter[2], David Mencin[3], Yazan Suleiman[2], Lloyd Carothers[4], Bruce Beaudoin[4], Matt Briggs[4], Garrett Bates[4],

Henry Berglund[3]

1. Metadata Game Changers, 2. IRIS Data Management Center, 3. UNAVCO Inc., 4. IRIS/PASSCAL

ABSTRACT

UNAVCO and IRIS, major repositories for global geodetic and seismic data, are in the process of joining their operations to form a unified facility for supporting the broad spectrum of geophysical observations and science required to help understand and predict the behavior of Earth Systems. This process would be complicated in a static data management environment, but both repositories are also migrating archives and services to the cloud as part of the merger. To simplify and unify archive data management, the organizations are collaborating to create common data and metadata models for observations from a wide variety of instruments and disciplines. For data, the initial focus has been on the xArray data model, already used in the geodetic and magnetotelluric communities, which can be implemented with several disc- and cloud-native approaches (HDF5, netCDF4, and Zarr). For metadata, the SensorML Standard developed by the Open Geospatial Consortium is being explored because 1) SensorML accommodates the large parameter space associated with instrument metadata required to use and trust complex observations and 2) the ability to extend the standard when required. The merger of two large repositories combined with migration to the cloud requires careful identification and on-going testing of a wide variety of assumptions about data management systems. This presentation will focus on lessons learned so far.