EOSDIS

NASA'S EARTH OBSERVING SYSTEM
DATA AND INFORMATION SYSTEM

Data Optimization for the Cloud

Earthdata Cloud Services

Patrick Quinn
EED-2 / Element 84
patrick@element84.com

James Gallagher
EED-2 / OPeNDAP
jgallagher@opendap.org

This work was supported by NASA/GSFC under Raytheon Technologies contract number NNG15HZ39C.

Review

- What is ‘Cloud optimized’
- Why care

I have a file location. How do I get the data I care about from it?

1. **Locate** parts of the file I need to read to get the data I care about
2. **Read** the parts of the file that have my data
3. **Translate** bytes I got into the data I wanted, usually decompressing and throwing away things I had to read but didn't want

Files on a hard drive

- Latency is very low, under 0.1ms for SSD
- Throughput is very high, > 200 MB/s.
- Parallelism is low, usually one CPU

Consequences for file formats:

- Optimal formats favor many small reads getting only the necessary data.

EOSDIS

SSD: Solid State Drive
CPU: Central Processing Unit

Files in the cloud

- Latency is very high, 70ms or more for S3
- Throughput is very high, up to 25G/s
- Parallelism is very high

Consequences for file formats:

- Optimal formats emphasize using fewer reads, **even if it means fetching extra data only to discard it**.
  Too many reads makes it faster to download the whole file and process offline.

Cloud-Optimized =

Able to fetch lots of data in a small number of reads

But operating at a scale where you can't just fetch everything

How the server reads/subsets data from S3

OPeNDAP in the Cloud and the DMR++

DMR++: Dataset Metadata Response with extensions

Overview

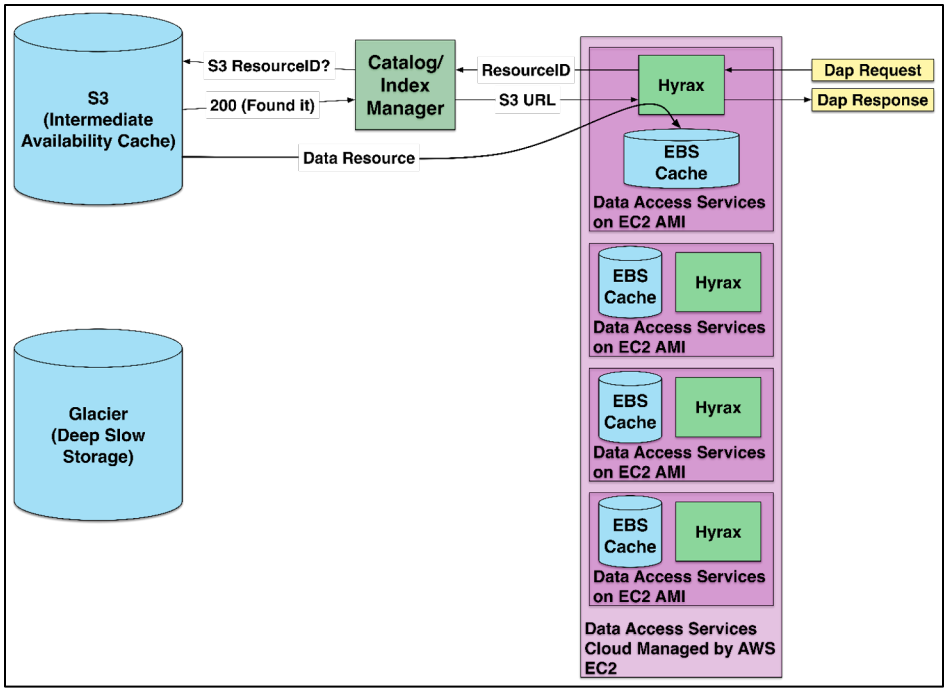
- Three possible ways to build ‘OPeNDAP in the Cloud’
- Let's talk about the ways each can be an optimal solution
- How the technique we chose works in practice

DMR == DAP4 Dataset Metadata Response

DMR++ == DMR with extra metadata

Arch 1: Baseline

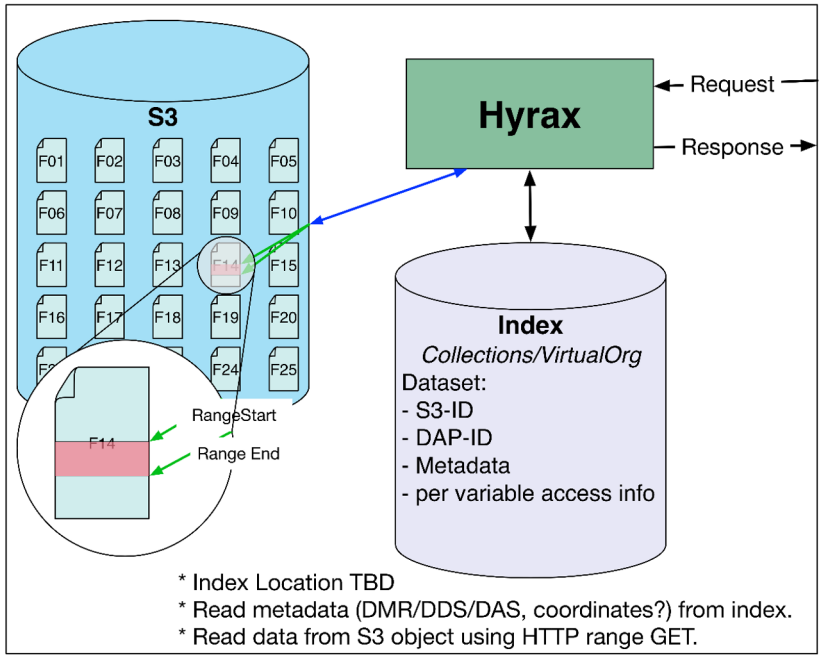
Access using whole files retrieved from Amazon's Simple Storage Service (S3)



EC2: Amazon's Elastic Computing
AMI: Amazon Machine Instance
EBS: Elastic Block Store

Arch 2: Files with HTTP Range GETs

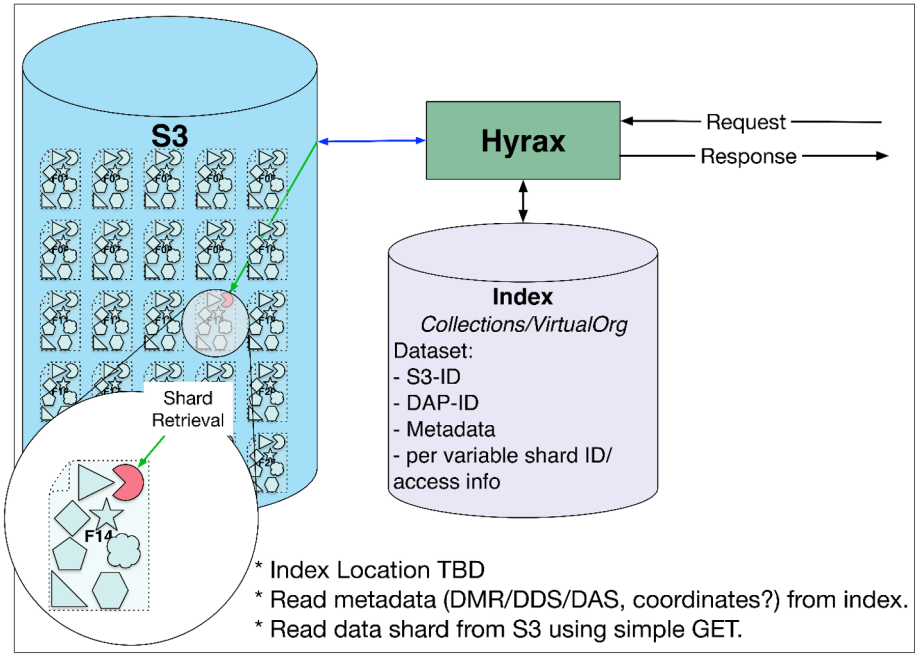
HTTP Range-Get access to whole files in S3



HTTP: Hypertext Transfer Protocol
DDS: Data Access Protocol (DAP) Dataset Descriptor Structure
DAS: DAP Dataset Attribute Structure

Arch 3: Shredded HDF5 Files

HTTP Get access to parts of HDF5 files as objects in S3



Compare these designs

- Transferring the whole file can be the fastest way to get data – even small amounts of data. Why?
  - Because files with many small parts require many accesses, each with high latency
- Virtual sharding preserves the original data file (can be used by Arch #1) and allows subset-in-place access.
  - To be performant, this virtual sharding must be optimized
  - NB: this can only be used if metadata are computed beforehand.
- True sharding is the solution chosen by the designers of zarr
  - For files with many variables/shards the cost of access to the whole file is very high (think 1,000 variables * 100 chunks/var * *X* latency)*
  - Reduce size – eliminate duplicate objects

Hyrax/DMR++ Use: Set up

- First, build the DMR++ metadata file
  - Choose options that alter the appearance of the file
  - DMR++ documents are (now) XML
- Store the Data and DMR++ metadata in S3, or another web object store (WOS)
  - We use CMR to find these S3 objects

EOSDIS

XML: eXtensible Markup Language
WOS: Web Object Store
CMR: Common Metadata Repository

Hyrax/DMR++ Use: Data access

- The server reads the byte-offset and size (i.e., number of bytes) of a variable from the DMR++
- The server uses HTTP Range GET operations to read just those bytes
- The server typically must issue many reads in parallel to get data for one variable
- Discrete ‘chunks’ are then assembled into the variable’s value

Hyrax/DMR++ Use: Optimization

- Issue parallel reads
- Decompress ‘chunks’ in parallel
- Combine adjacent ‘chunks’ to maximize transfer volume per request

Other tricks

- The DMR++ can be used to aggregate data across many objects
- It is generally much faster than aggregations formed using files
  - There is no need to open each file and read its metadata
- It can also be used on spinning disks
- Data are accesses without the format API library – maybe we could make a very thin server?

Questions?

James Gallagher
EED-2 Software Engineer
jgallagher@opendap.org

This work was supported by NASA/GSFC under Raytheon Technologies contract number NNG15HZ39C.