EOSDIS
NASA'S EARTH OBSERVING SYSTEM
DATA AND INFORMATION SYSTEM

Hosting Hyrax in the Cloud While Preserving the User Experience

ESIP Summer 2018

James Gallagher, contractor
EED-2 Role: Technical lead – OPeNDAP
jgallagher@opendap.org

This work was supported by NASA/GSFC under Raytheon Co. 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.

Outline

- Hosting Hyrax Server in the Cloud
- Preserving User Experience
- Retaining Existing Data Organization
- Mitigating Object Store Latency
- Summary

The Hyrax Server in the Cloud

- Part of the EOSDIS Evolution and Development (EED-2) contract
- Serve data (mostly HDF5) from S3
- Keep the current User Experience (Ux)
- I'll describe two important aspects to this that apply to any software moving from traditional hardware to the cloud
  - Retaining existing data organization
  - Mitigating object store latency

Why Preserve User Experience?

- When there's no new functionality…
  - People expect the same Ux
  - … unless benefits are substantial
- But when new concepts are added…
  - There is an implied new Ux
  - … but preserve existing features

EOSDIS

Simple Web APIs too?

- Yes! Even simple web APIs present a UX
- Data organization
  - People expect data to ‘look the same’
  - Programs are even less flexible
  - Changes in organization disrupt established workflows
- Access latency
  - People expect quick responses
  - Machines require that response not time-out

Preserving hierarchies

- Cloud Object APIs* use Key-Value
- These are flat – no hierarchies
  - There are ways to fake hier using names…
- To provide a hierarchical view
  - Encode the hierarchy
  - Bind into that encoding the Keys
  - Present the view with links

EOSDIS

Example: THREDDS Catalogs

- Use XML to build a linked set of catalogs
- Store these on the object store or not
- Can be transformed to HTML
  - In cloud or in client's browser
- Not just THREDDS – generic XML too
- Supports browsers; more software for Web API
- Web architecture friendly

Example: NASA EED-2 CMR

- Use a service – query about:
  - Collections
  - Specific levels
  - Individual files/granules
- Support browsers and Web API
- Central store of information – can be used by many different applications

Why Response Latency Matters

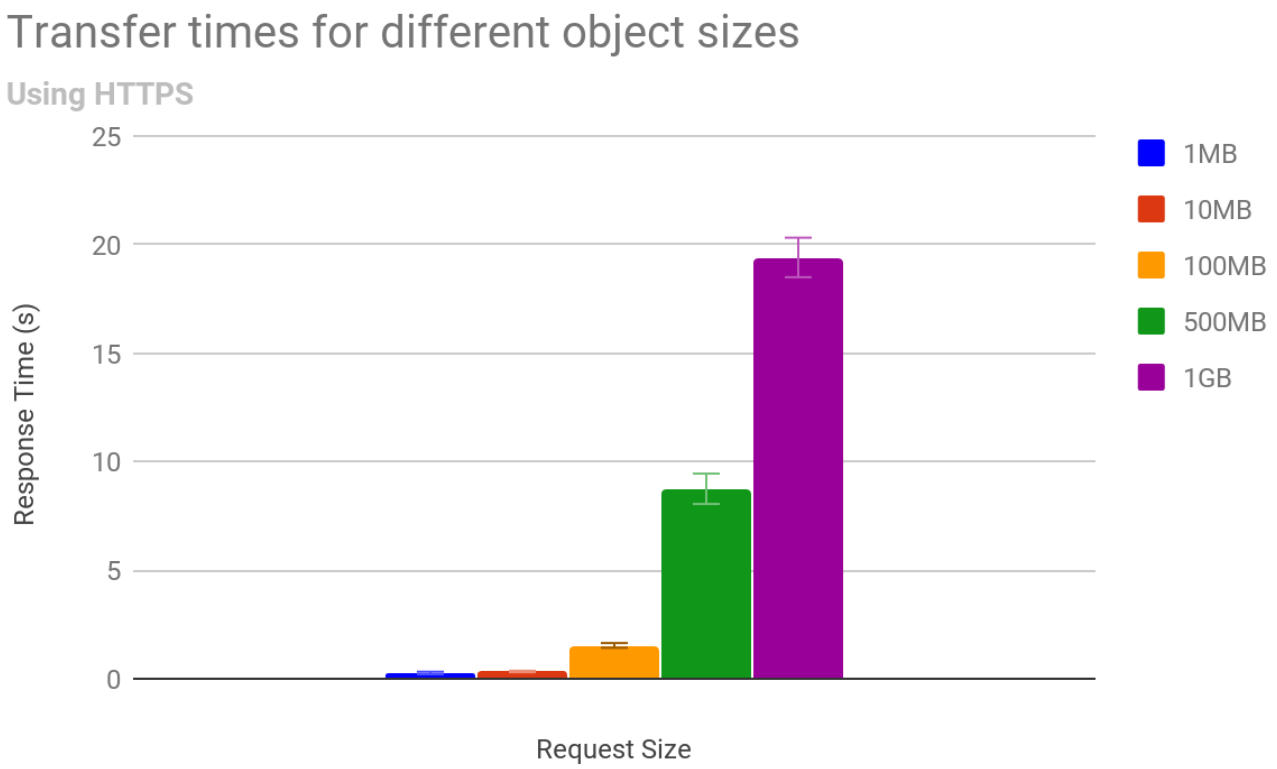
- Existing software, low-latency storage (reuse; *caveat emptor*)
- Users expect similar responsiveness
  - … unless new capabilities are added
- Worst case: Latencies compound and lead to outright failure

EOSDIS

Response Latency (S3-centric)

- HTTP access involves inherent latencies
  - Amazon S3: 70ms HTTP and 270ms HTTPS
- HTTP Keep-Alive can mitigate cost of many accesses
- Parallel transfers can boost transfer speeds

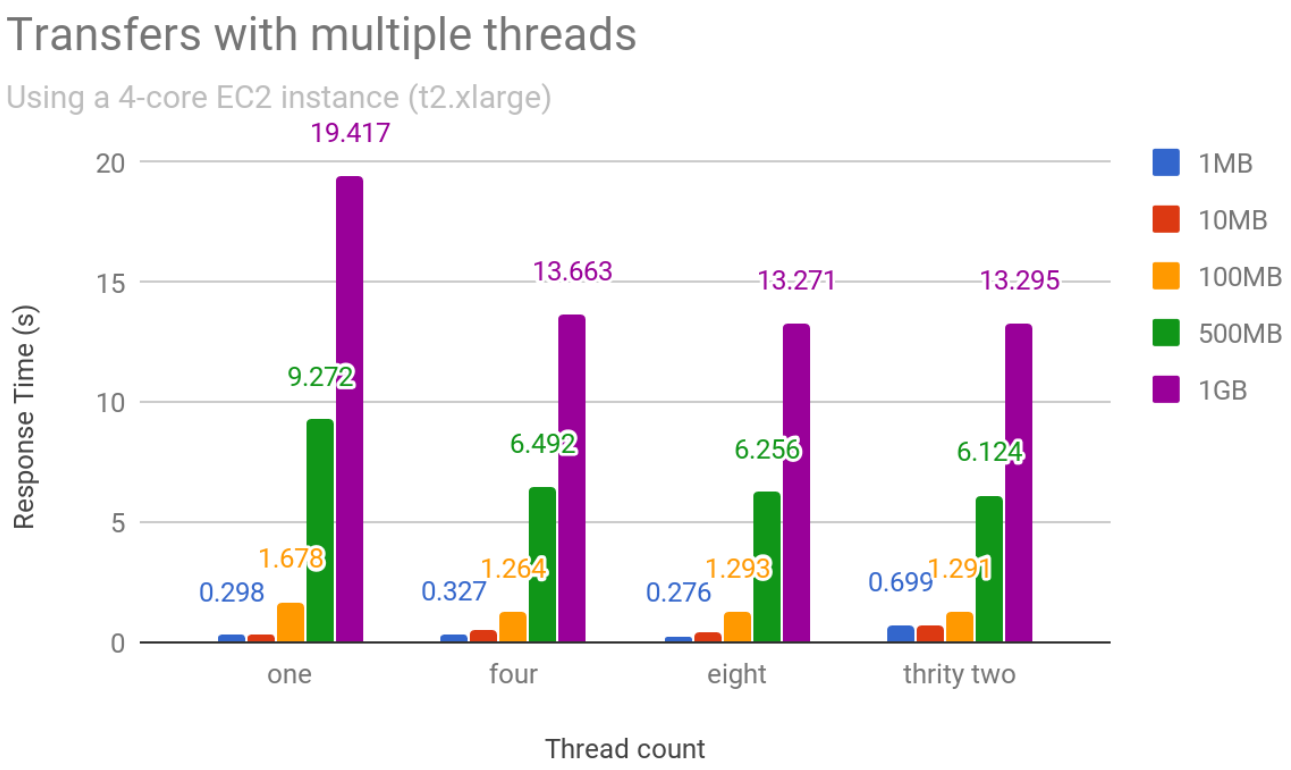
Accessing Data From the S3 Object Store: Baseline Data



- Smaller objects' transfer times are dominated by S3 overhead
- Objects 100MB and greater show linear relationship between size and time

EOSDIS

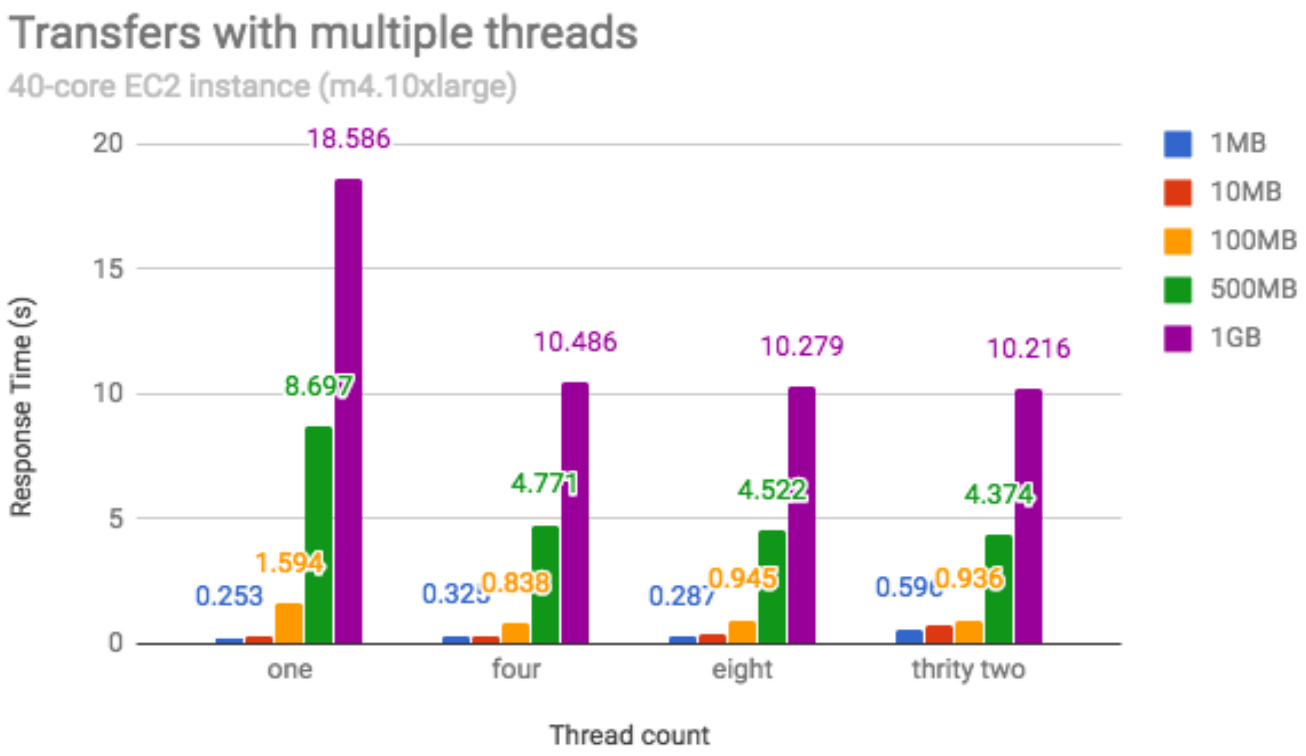
Parallel access to shards decreases response time



- Parallel access using shards improves response time for large objects
- But, it does not improve response time for small objects

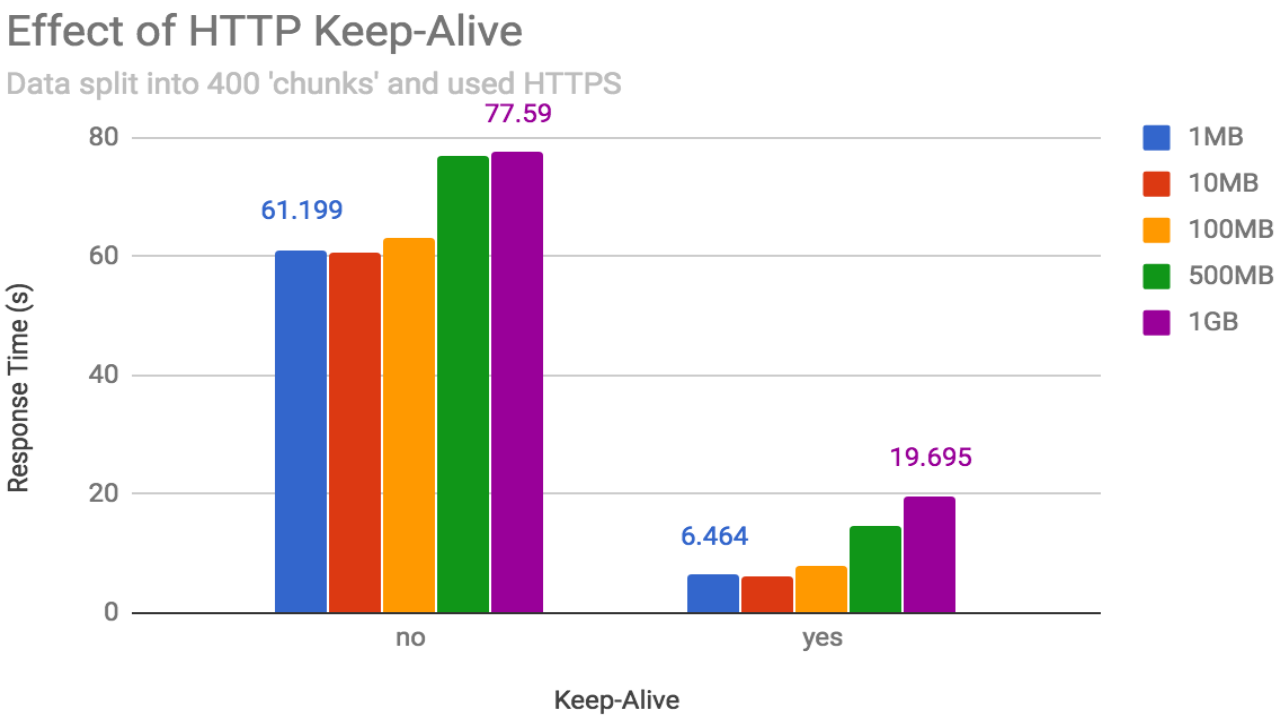
EOSDIS

Do bigger EC2 instances alter this?



- Increasing the number of cores does not radically alter the shape of the response curve
- (note that the somewhat faster m4 machine achieves better performance over all)

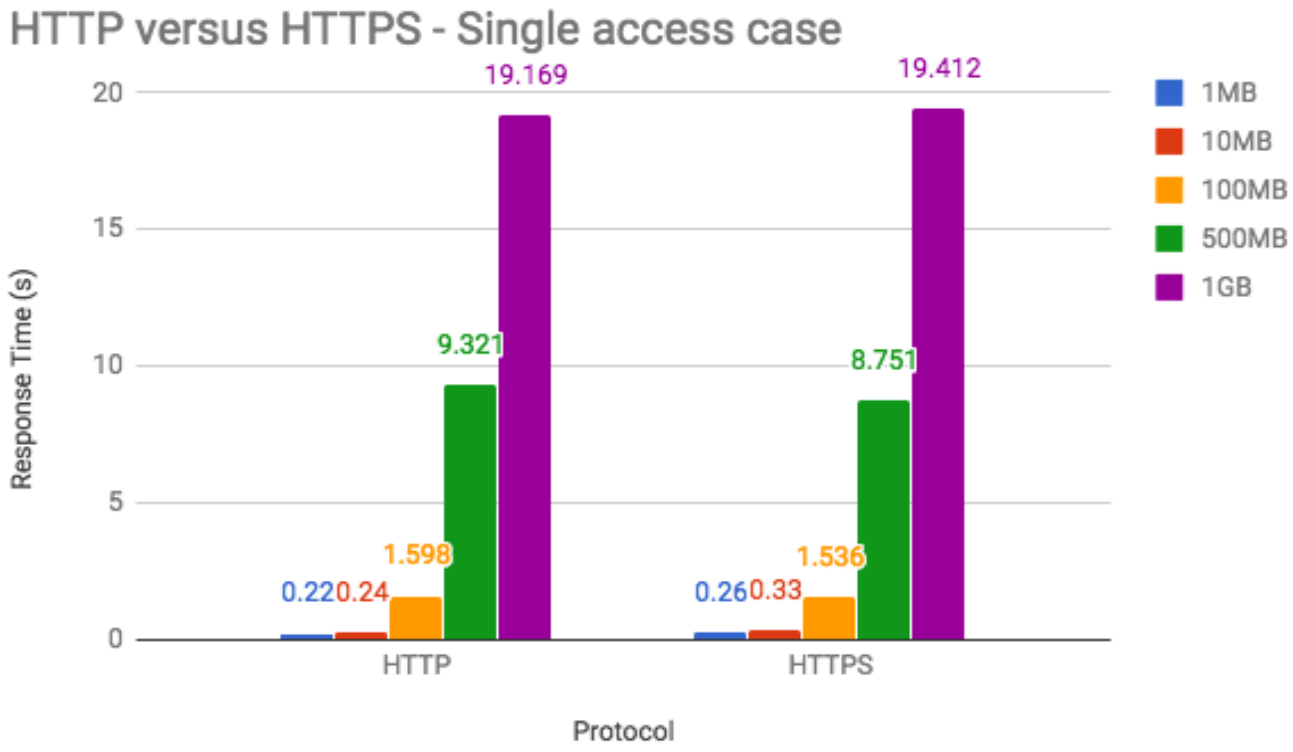
HTTP connection reuse is important when using sharding



- 400 shards and accessed them with and without connection
- Note that for the small responses there is a significant penalty in performance even with connection reuse while for the large responses there is negligible cost

EOSDIS

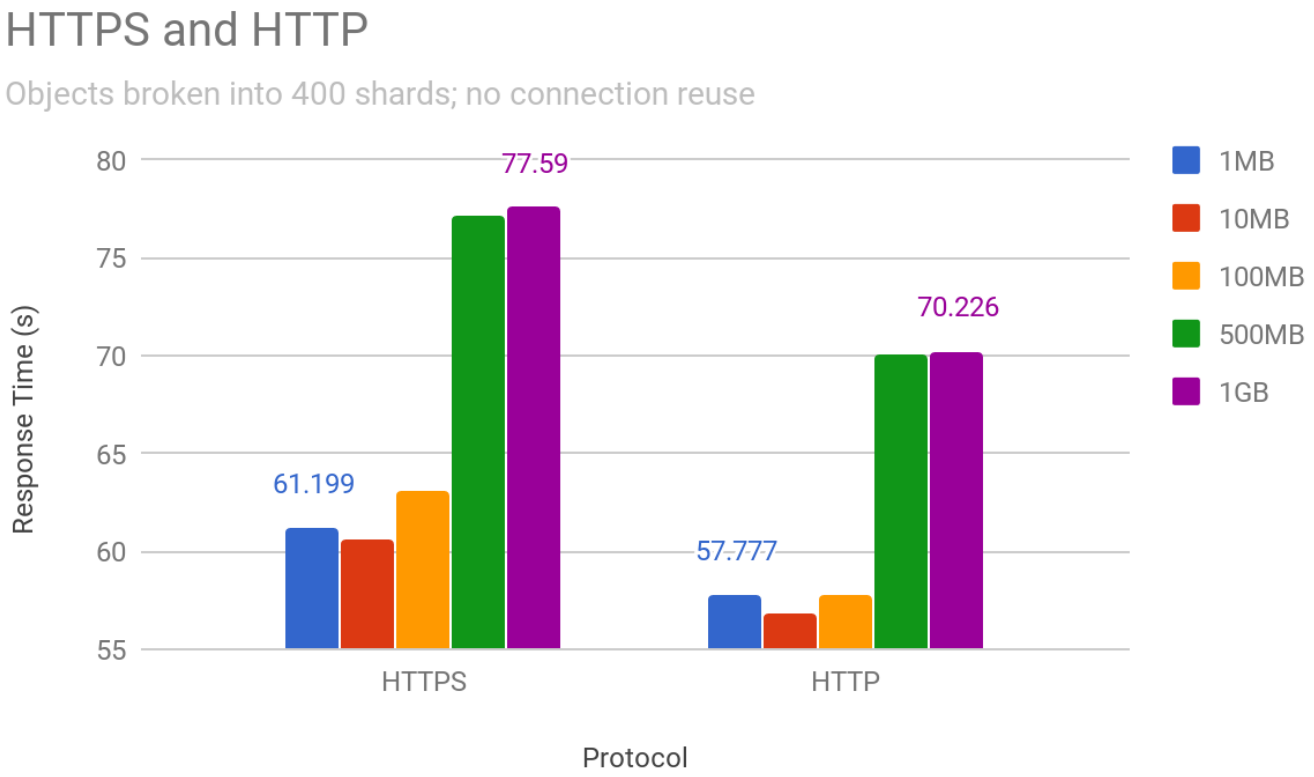
HTTP versus HTTPS?
The Single access case



- Using HTTP instead of HTTPS makes no difference for large objects and negligible difference for small ones

EOSDIS

HTTP versus HTTPS
But with shards...



- With shards (e.g., 400), the cost of the HTTPS over HTTP does become noticeable.
- *Where is the cost*? It is primarily the cost of establishing the secure connection.

Summary

- Preserving hierarchies is important
  - Can be done using static ‘catalogs’
  - Or, using a dynamic service
  - Model the hierarchy, Encode the keys, Present the view
- Minimizing response latency
  - Parallel access increases transfer rate
  - Connection reuse reduces Tx startup time
  - HTTP is faster than HTTPS for small Txs

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

Raytheon

in partnership with

ati

Bellfore Consulting Group

Element 84

Genex Systems

The HDF Group

Innovim The Power of Innovation

Maryland Hawk Corporation

OPeNDAP

REAN Cloud

Red Canyon Engineering & Software

Stinger Ghaffarian Technologies

Valador

EOSDIS