XCache experience Virtual Placement (and ServiceX)

MANIAC LAB

Ilija Vukotic

US ATLAS Computing Facilities @SLAC F2F

2022-12-01

ATLAS EXPERIMENT

ATLAS Distributed Data Management

Almost all of our data in Rucio (exception are some unregistered datasets in CERN EOS).

Data placement/movement governed by Rucio rules and available disk space.

Jobs go where the data is (except - Panda can move the data by creating temporary replicas).

That all works, specially for large workloads - MC generation, reprocessing campaigns, etc.

ATLAS Distributed Data Management

But could be better...

- No natural way to support hot/cold storages
- Temporary replicas:
  - Add latency as it involves Rucio and FTS, jobs can't start until all files have been transferred
  - A lot of bandwidth is needed
  - Further reduce average number of accesses per file (coldness)
- Doesn't allow for "storage-less" sites
- Impractical for quick turnaround, low available bandwidth, preemptable queue use cases:
  - Analysis facilities (ServiceX, Coffea, Coffea-casa,..)
  - Cloud resources, HPCs

What is Virtual Placement?

VP is a mechanism that enables efficient and reliable data access over WAN.

Expected benefits:

- Enables storageless sites
- Less WAN bandwidth usage
- Less rescheduling
- Faster task turnaround
- Less replicas

Components - Origins

- Origins are all active ATLAS DDM endpoints.
- All origins should have a working xroot endpoint.
- If dual stacked it has to serve on both IPv4 and IPv6.
- If an endpoint is inaccessible it has to be set as being offline.
- This is a requirement for other reasons too (eg. ServiceX)

Components - XCache

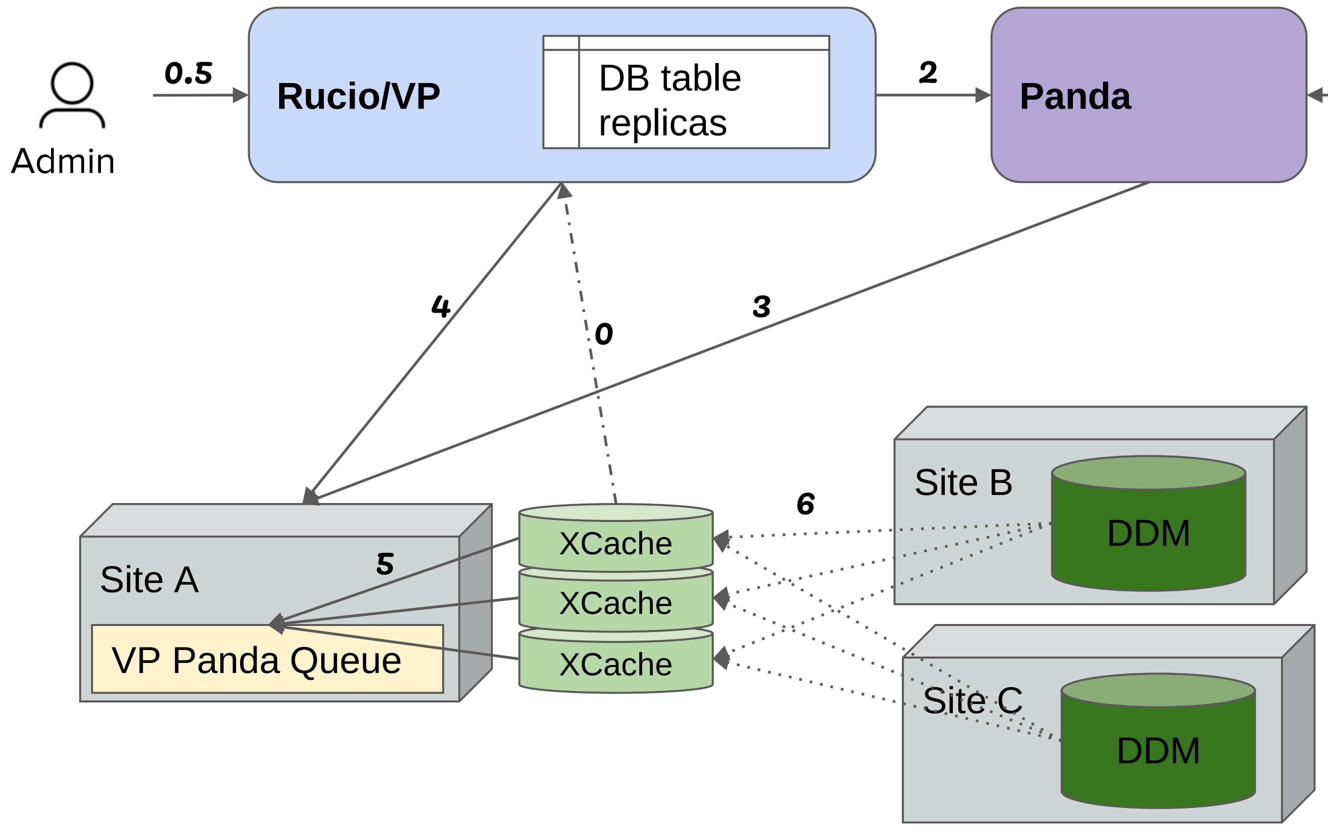
- Each site using VP needs one or preferably more XCache nodes.
- Nodes are equipped with JBODs (10-100TB) and a decent NIC.
- Runs xrootd but no cmsd, nodes are not federated
- Caches only blocks that are required.
- Each block is checksummed and retransfered if corrupted.
- Nodes send heartbeats every 10s.
- All nodes are centrally monitored, all details of all the accesses are logged.

Components - Panda

- To send a job to a queue, Panda requires an input data replica already present at the site.
- VP creates dataset Virtual Replicas, that are permanently fixed to 3 VP DDM endpoints.
- Panda will use Virtual Replicas only if regular replicas can’t be used (DDM endpoint or Site is down, regular queues are too busy).

Components - VP/Rucio

- Knows which xcache site(s) serve which ATLAS site(s)
- Keeps track of live XCaches
- Calculates probabilities to create VP replicas
- “remembers” virtual replicas
- Uses Rendezvous hashing to decide which XCache node of the XCache site will be used for each individual access.



6

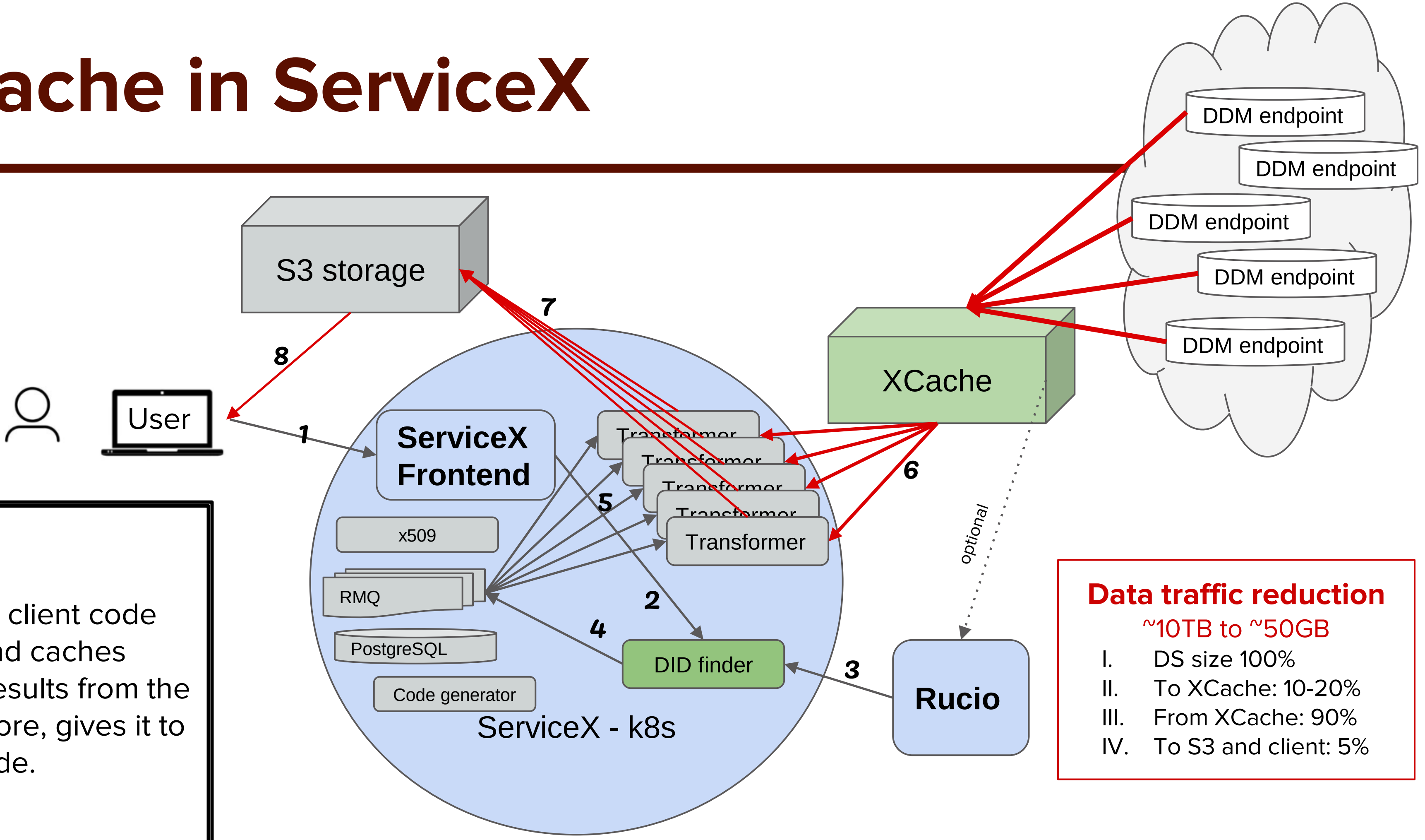
Job

If a file was not accessed before or user now has different selection, parts of the file missing in xcache gets transferred from real replica, delivered to the job, and stored in the xcache.

What is ServiceX?

ServiceX aims to provide nearly interactive filtering, enrichment, transformation of very large datasets and result delivery in multiple formats, with emphasis on pythonic style analysis.

XCache in ServiceX



8

ServiceX client code reads (and caches locally) results from the object store, gives it to users code.

Data traffic reduction

~10TB to ~50GB

I. DS size 100%
II. To XCache: 10-20%
III. From XCache: 90%
IV. To S3 and client: 5%

Requirements on XCache in ServiceX

- Must sustain thousands of concurrent writes/reads mostly in small files.
- Storage
    - performance - must be top of the line - all nVME storage.
    - Size - roughly one month of turnover time.
- At least 40Gbps NIC(s).
- Access to HTTP data too.
    - We added xrdcl-http plugin that allows access to CERN open data and data at certain Tier3s that don’t have xrootd doors.
    - Paths like: root://xcache.xxx.org//http://origin.xxx.org

Experience with XCache

- Image building
- Deployment
- Registration
- Monitoring
- Stability and performance of XCaches
- Stability and performance of the whole system
- Issues

Image building

- The original idea was for OSG to build and test images for everyone.
- Have a base image, and on top of that ATLAS, CMS, Stashcache could add things to address specific needs (code here).
- Despite several merge attempts, we (ATLAS) are constantly out of sync. Mainly due to me being constantly overcommitted. But also due to me doing frequent tweaks.
- We build in GitOps and push images both to DockerHub and OSG Harbor.

Image building - ATLAS specifics

- Configuration:
    - Limits (file descriptors, processes)
    - Monitoring configuration
    - Block size, prefetch
    - Xrdcl-http plugin
- gStream2tcp - for our gStream monitoring
- Extra monitoring
    - Reports CPU utilization, memory, all disks activity, network ingress/egress, etc.
- Heartbeats sending
    - To VP service
    - To Rucio, using Rucio API.
- Dark data cleaning
    - Done once before server starts. Cleans data without cinfo files, cinfo files without data, empty directories, etc.
- Docker compose deployment.

Deployment options

Three different ways to setup XCache:

- Very easy - install k8s, install SLATE, fedOps team installs and manages xcache.
- Easy - install docker-compose, use provided template to configure and start service. When informed that an upgrade is needed, simply restart it.
- Not hard - install xcache, setup two cron jobs. When asked update things.

We update certificate on all xcache nodes once per year. These use special service certificate.

Deployments on SLATE

XCache application in SLATE is a regular HELM chart with just a few additional fields.

Deployment/update of the application on SLATE site takes ~1 minute and is completely transparent for the site admins.

Restart of node or even cluster update are transparent to me as XCache network administrator.

SLATE XCache instance have their helm configuration values in a github repository, and I can update settings by a simple push as FluxCD will redeploy the edited instance.

Deployments - XCache hardware

- It can (and often does) use older hardware.
    - eg. Prague - node with 89, 1TB disks.
- While NVMe is prefered, HDDs work OK (the more spindles the better).
- Optimally three independent nodes, but even a single instance setup works well as XCache is now much more stable than before.

Registration

I need a low latency between xcache going up/down and rucio knowing about it.

All XCache instances have been removed from OSG Topology and ATLAS-Crick (latency at least 30 min).

XCaches send heartbeats every 10 seconds to VP informing it about: instance name, xcache site, total disk size, IP address and port. Three missed heartbeats and the instance is unregistered.

Rucio keeps a map of which ATLAS site is served by which XCache site.

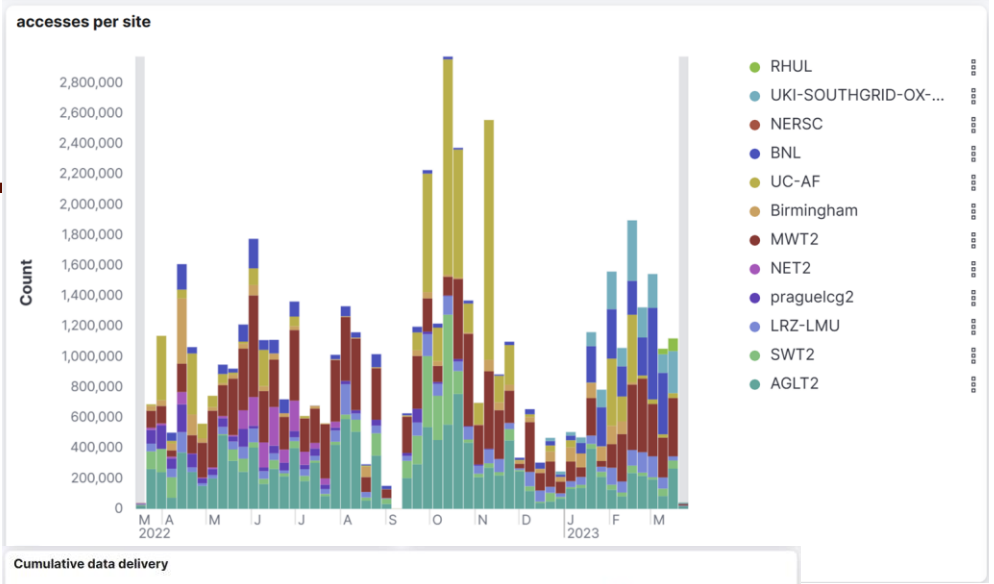
Monitoring I

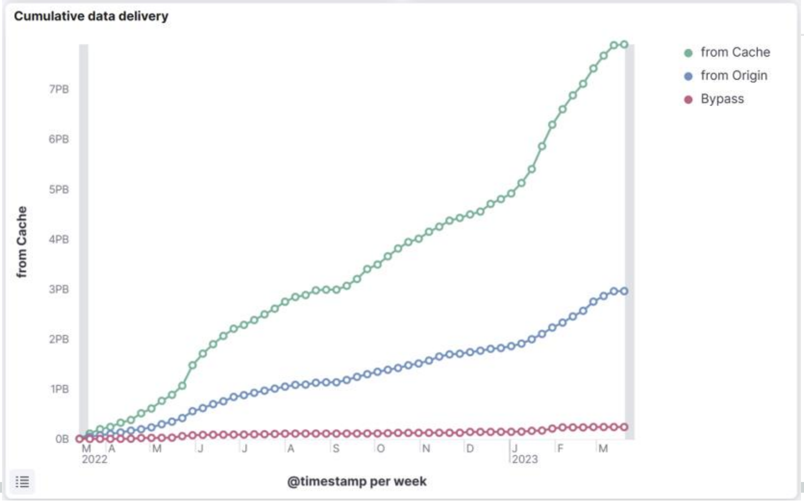
Main monitoring is gStream based.

Each XCache instance is a k8s pod with one of the containers responsible for monitoring.

Runs gStream2tcp, simple python code that receives UDP packets, decodes them, repacks info into JSON documents, sends them to a logstash instance running at UC River cluster, which indexes it in UC Elasticsearch.

All details of each access to each file are preserved.

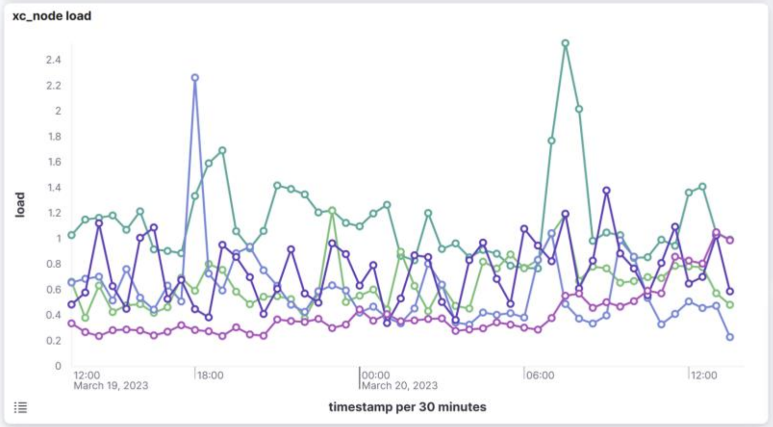


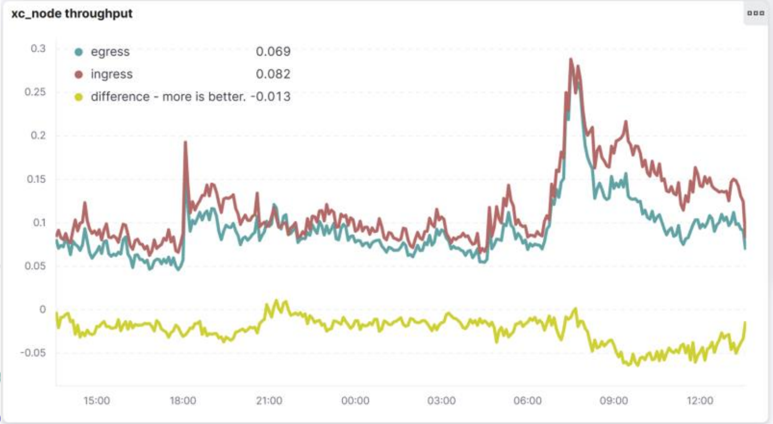


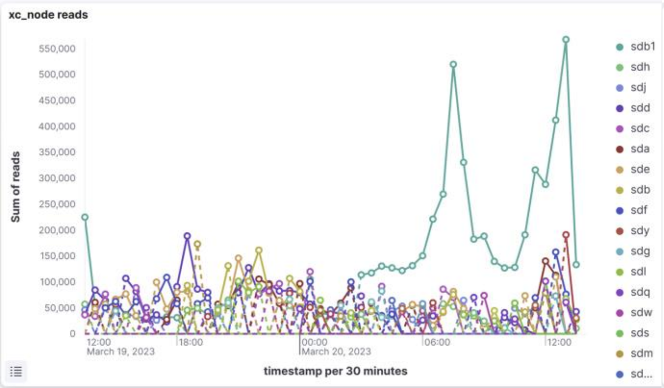
Monitoring II

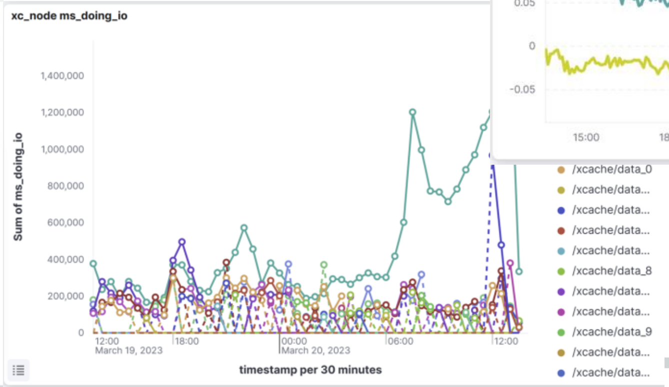
SLATE deployed hosts run a code that periodically report CPU usage, utilization, network Ingress/Egress, etc.

While interesting, it isn’t really actionable.









Monitoring III

- There are two kinds of ATLAS jobs: direct access and copy2scratch
    - Direct access - pilot tests that it can access each input file before letting the job start.
    - copy2scratch - pilot uses rucio mover to pre-fetch data to scratch.
- Both methods report full information about each file access to Rucio traces.
- Rucio traces are also indexed at UC Elasticsearch.
- Useful to find issues with origins.
- A lot of issues are temporary (origin busy, slow, temporarily inaccessible). I run continuous “recheck code” that retries failed transfers.

Monitoring IV

It happens from time to time that xcache is sending heartbeats, delivering cached data, part of the jobs is succeeding, but xcache can’t access any outside data.

External tester:

- Creates and registers a new Rucio dataset every night with 288 unique 1kb files.
- Every 5 min accesses a new file through each of the supposedly live xcaches, raises alarm if needed.
- Doesn’t work with xcaches exposed only on a site local network.

Stability and performance of XCache

XCache stability is now very high.

I don’t see any crashes or unexplained restarts.

The biggest issue is unreliable hardware. With nodes of 30+ spinning disks that are all out of warranty, it too often happens that the disk dies, then nodes is left hanging. This requires admin intervention to identify failed disk, make a github PR to remove the disk.

Performance of systems with SSDs is always great. One node can easily serve at least 3k worker nodes. A single node xcache with HDDs can easily be pushed to show data bypass.

Performance of cached HTTP accesses is still not tested at sufficient scale

Stability of Virtual Placement system

Designed to gracefully handle failures so all the fixes can wait for Monday morning.

- If a disk fails, only some small number of jobs will failover to origin.
- If an xcache node goes down it gets removed from rotation in 30s. Minimally affects number of cache hits.
- If all xcache nodes at the site go down, jobs get data directly from origin.
- If whole of VP goes down, current jobs will finish and new ones won’t be submitted.

Issues

- Caches that partially work (eg. bad disk, incapable of getting origin data but capable of serving cached data, etc.) I would prefer them self-repairing or even shutting down
- Not everyone wants a SLATE deployment. It is easier to handle 10 remotely managed sites than a single “proxy” managed one.
- Bad origins (eg. site has xrootd door registered and set active when it is not so, too slow transfers)
- Issues with our WFMS
  - allowing users to demand copy2scratch
  - allowing users to avoid VP brokering
  - bad error reporting from Pilot.

Plans

- Test single site scaling. Now being done at BNL at 2500 cores, but can probably go significantly higher.
- Test at HPC site. Currently done on NERSC PerlMutter. A whole other set of configuration issues to solve.
- Compare performance of xrootd and http origins.
- Compare performance of xcache http and other http caching options (NGINX, Apache Traffic Service)
- Finish VP Rucio integration.