Operating the 200Gbps IRIS-HEP Demonstrator for ATLAS

Rob Gardner[1] on behalf of Doug Benjamin[2], Lincoln Bryant[1], Matthew Feickert[4], Farnaz Golnaraghi[1], Alexander Held[4], Fengping Hu[1], David Jordan[1], Ofer Rind[2], Judith Lorraine Stephen[1], Ilija Vukotic[1], Gordon Watts[3], Wei Yang[5]

[1]University of Chicago [2]Brookhaven National Laboratory [3]University of Washington [4]University of Wisconsin-Madison [5]SLAC

CHEP 2024, Oct 19-24, 2024

Motivation

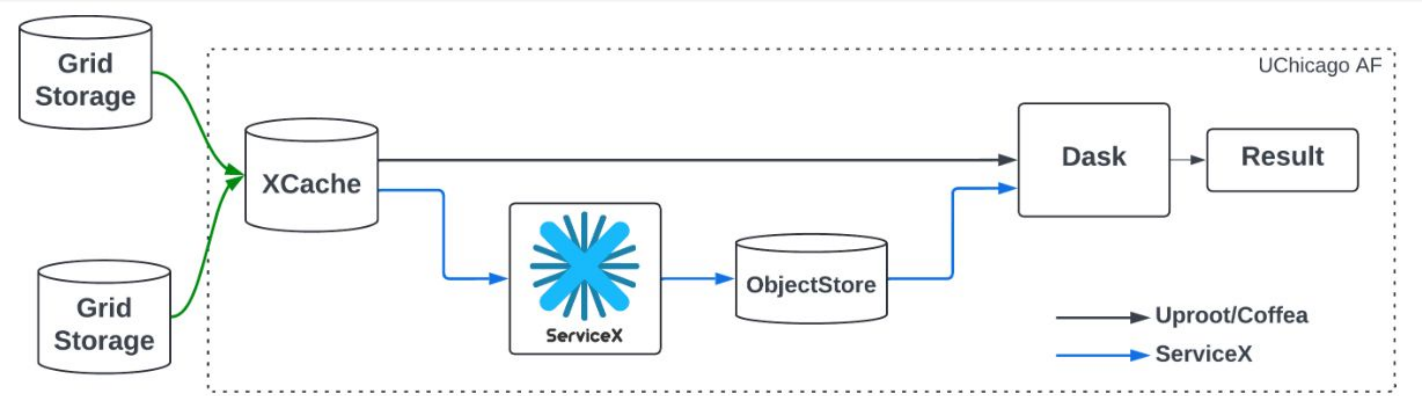
- High level goal: Make progress toward showing a realistic HL-LHC analysis (see Alex Held's talk from Monday)
- A full-scale model assumed 200TB read in 30 minutes
- Correspondingly, at 25% scale this ends up being around **200Gbps sustained data rate**
- We used the UChicago Analysis Facility as a testing ground for the ATLAS part of the 200Gbps Challenge

UChicago Analysis Facility

- The UChicago team operates a production Analysis Facility (AF) for ATLAS
  - Mix of traditional batch (HTCondor), interactive Jupyter notebooks (including GPU) and other interesting technologies (e.g. REANA, BinderHub, Triton inference, ...)
  - co-located with a large ATLAS Tier2 center (Midwest Tier2 – MWT2)
- Built on a flexible Kubernetes infrastructure, ideal for dynamic reconfiguration to meet the needs of this challenge and any future challenges
- Hardware specs:
  - About 35 hyperconverged (lots of disk, memory, CPU) nodes suitable for serving up storage, job slots, etc
  - 4 login nodes, 6 GPU nodes, a few other machines dedicated to running Jupyter notebooks
  - Added 75 additional servers from the Tier2 to meet CPU demand

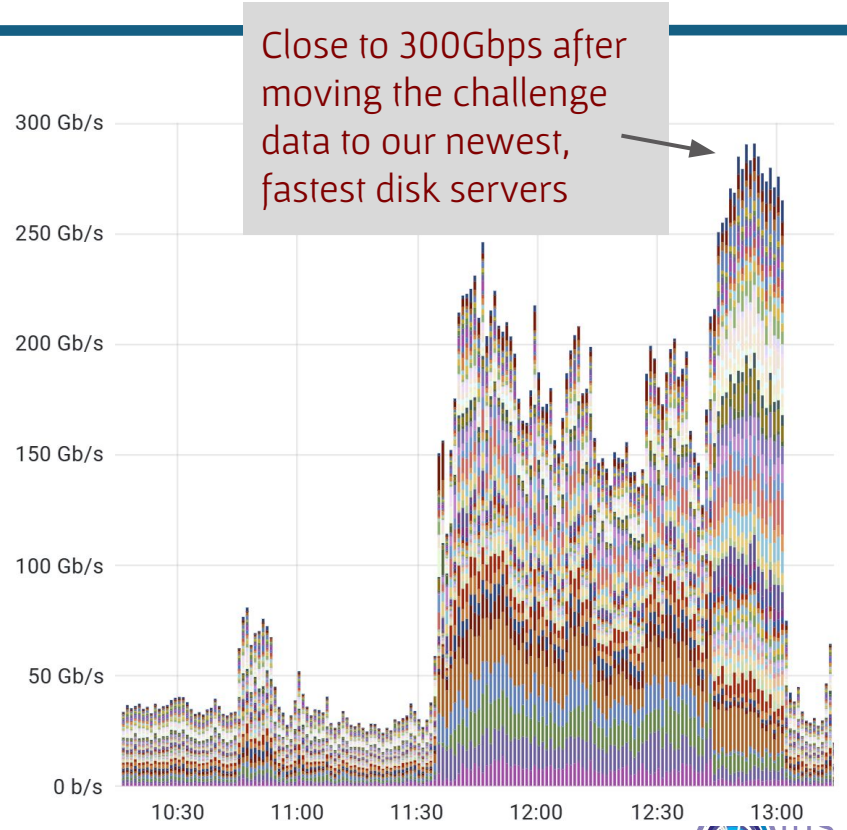
The Shape of the Challenge

- Two data paths
  - Data flowing from XCache directly to Dask workers using Uproot/Coffea
  - Data flowing from XCache to ServiceX, transformed into columnar format and stored on an S3-like object store, and then read into Dask workers



Starting simply

- We replicated the entire 200G Challenge dataset to MWT2 LOCALGROUPDISK for the challenge
- Before getting into ServiceX, Pythonic data analysis tools, etc, we first wanted to see how fast we could directly read data out of MWT2 dCache
- Up to ~300Gbps with 250 XRootD clients (xrdcp to /dev/null) **while serving production MWT2 workloads**

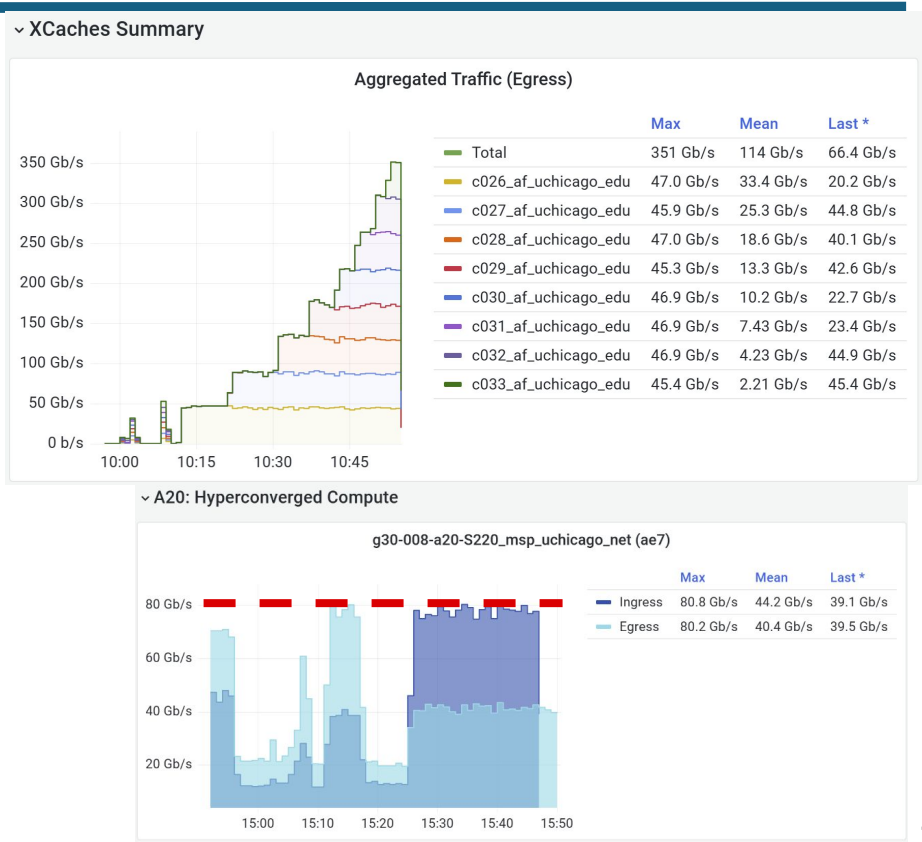


XCache Configuration

- Before the challenge, the AF was served by a single XCache server at 2x25Gbps connectivity
- Decided on 8 nodes at 25Gbps each
  - **200Gbps** bandwidth in total
    - doubled to **400Gbps** via 2x 25Gbps interface bonding
  - Each node with 10x3.2TB NVMe (256TB in total)
    - Enough to contain the entire 200G challenge dataset (191TB) once the caches are warmed up
  - Disks configured in JBOD mode with XFS and mounted e.g.
    - `/xcache/{1..N}`
    - `/xcache/meta` for metadata
  - Nodes were not clustered. Rucio assigns an xcache node to each file (filename hashing).

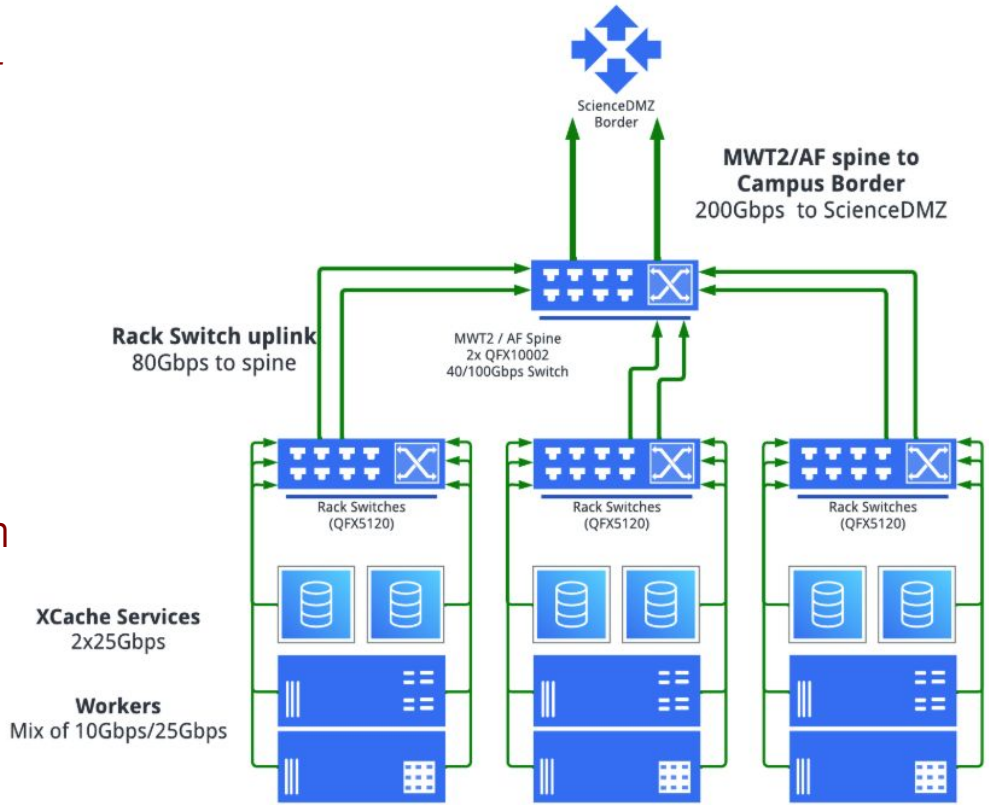
XCache Performance

- From a software perspective, XCache performed well
  - Clients were able to saturate the network on each XCache (~50Gbps) easily
- However, a problem with the network topology became apparent when we tried to scale
  - **Limited to 80Gbps** between switches!
- All 8 XCache nodes were on the same switch, but all of the workers were not!
  - Partially mitigated by moving half of the XCaches to another switch, such that half of the workers got half of the XCaches



AF Network Topology

- Hyperconverged nodes split across 2 racks, each with 80Gbps top-of-rack to spine
- 2 additional racks with workers reclaimed from MWT2 retirements
- XCaches split across racks, to try to have a large portion of the cache accesses happen within the rack switch



Challenges running a production AF in parallel

- Users started to feel the limited available job slots (due to Dask workers taking priority over HTCondor in K8S)
- We decided to permanently move 75 of the oldest Midwest Tier2 workers at UChicago (Dell 13th Gen) to the Analysis Facility to help alleviate the load
  - 3,000 additional hyperthreaded cores
- Configured a K8S Horizontal Pod Autoscaler such that:
  - There is a static configuration of 3000 (HT) cores for HTCondor
  - Additional HTCondor pods start up if there is demand in the queue and availability on the rest of the nodes in the K8S cluster

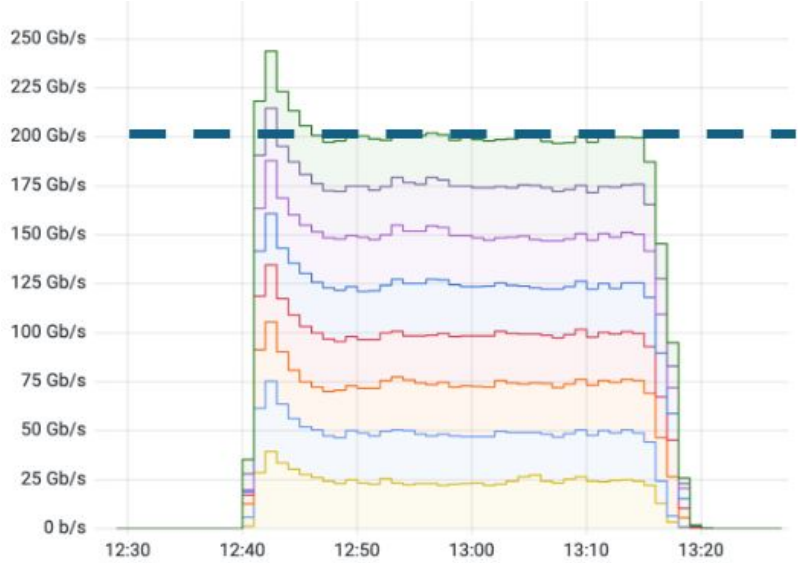
Other changes/updates

- Mass parallel Dask worker launches effectively caused a denial-of-service against etcd (the Kubernetes database)
  - Fixed by tuning the maximum database size
- Calico (K8S networking plugin) internally uses a MTU of 1450 bytes by default
  - Shouldn't cause issues in this configuration, but could be a source of unnecessary overheads if the kernel is (presumably) repacking 9K MTU packets into <1450 byte packets for the Calico interfaces
- Curious DNS resolver timeouts when many ServiceX workers were launched
  - Mitigated by setting up NSCD (name server caching daemon) everywhere, but not fully understood

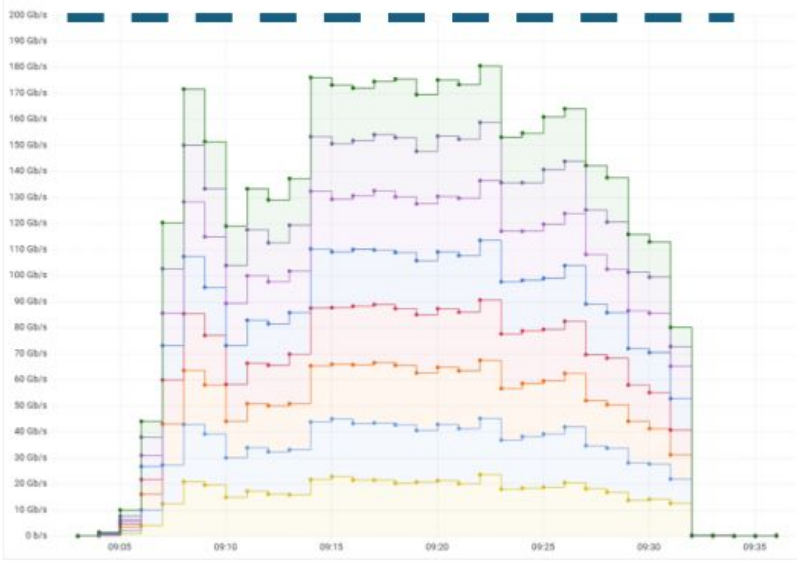
General observations about bottlenecks

- Considering this is all in a Kubernetes cluster, at any given time there is a lot of traffic going through many layers of indirection
  - Ingresses, LoadBalancers, CNIs, …
- Two ways we can go:
  - Make the services much smarter and rack/switch-aware
    - May be possible in the long term
    - Increased maintenance burden
  - Eliminate the bottlenecks in the network
    - Probably best in the short term

Results – Success!



Read Method: uproot



Read Method: ServiceX

Planning future extensions to the AF

- We are thinking about what kind of infrastructure we'll need to support further HL-LHC work
- Purely flash storage and 100Gbps networking is becoming reasonably priced these days
  - Servers with 100TB+ of NVMe and 100Gbps connectivity are less than $20K USD
  - White-box switch vendors like Edgecore are selling reasonably performant and comparatively inexpensive (<$40K USD) 32 port, 400Gbps switches
- Perhaps an add-on to the Analysis Facility with ~1PB of all NVMe storage, 100Gbps connectivity throughout, and a few thousand cores
  - Would be nice to have good connectivity to dCache (LOCALGROUPDISK, DATADISK) at MWT2 as well !

Summary

- After a considerable amount of facility reconfiguration and tuning, we were successful in meeting the IRIS-HEP 200Gbps challenge
- The demonstrator exposed bottlenecks in our network and places to improve the infrastructure
- Expect to conduct future challenges and "mini"-challenges as more realistic analysis tasks and situations (e.g. multi-user) are designed
- The results of these efforts go a long way to inform Facility R&D efforts for HL-LHC → and future infrastructure investments both at our Analysis Facility and our Tier 2 center

Thank you!

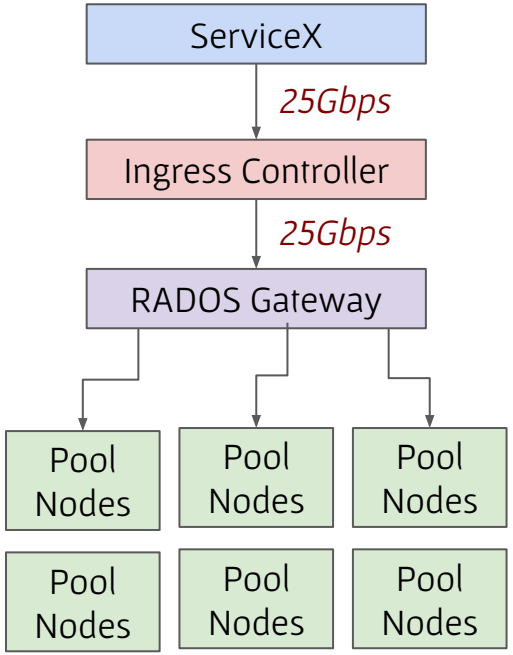
This work was supported in part by these NSF grants:

- OAC-2115148 CICI:UCSS:Securing an Open and Trustworthy Ecosystem for Research Infrastructure and Applications (SOTERIA)
- OAC-2029176 Collaborative Research: IRNC: Testbed: FAB: FABRIC Across Borders
- OAC-1836650 Institute for Research and Innovation in Software for High Energy Physics (IRIS-HEP)
- PHY-2120747, U.S. ATLAS Operations: Discovery and Measurement at the Energy Frontier

extra

Rook Configuration and challenges

- The UChicago AF has a Rook (Ceph) storage system in K8S for various storage needs (POSIX FS, Block Devices, and S3)
- RADOS Gateway (S3) uses an all-flash NVMe disk pool with 3x replication
  - (this also means that any write is amplified 3x!)
- At the beginning of the challenge we had a single RADOSGW serving the AF, behind an Ingress controller which handles all of the TLS termination
  - It was clear this would be a bottleneck for ServiceX



RADOSGW Performance

- Significant number of `503 "Slow Down"` messages from the RADOS gateway indicating that something was being overloaded
- Scaling up the number of gateways was a little challenging for two reasons:
  - There was still only one ingress, which proxied traffic through 1 server
  - Our MetalLB configuration (L2 mode) also proxies all traffic through 1 server
- To mitigate, we put a RADOSGW on every node with at least 25Gbps connectivity and switched the networking interface to NodePort
- We also discovered that the number of Ceph Placement Groups was dangerously low and badly skewing data distribution across the pool
  - Resolved by upping the number of PGs from 32 → 512