400Gbps benchmark of XRootD HTTP-TPC

Aashay Arora[1,*], *Jonathan* Guiang[1,**], *Diego* Davila[1,***], *Frank* Würthwein[1,****], *Justas* Balcas[2,†], and *Harvey* Newman[2,‡]

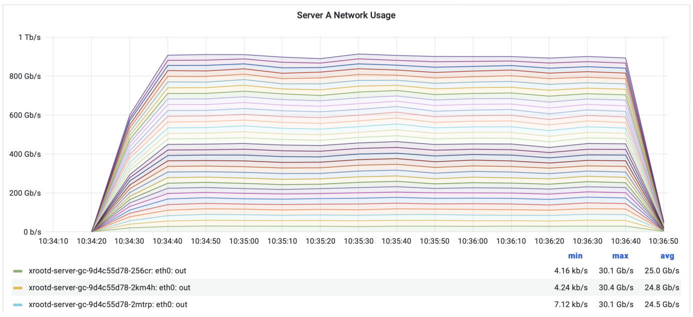
[1]University of California, San Diego, La Jolla, CA, USA
[2]California Institute of Technology, Pasadena, CA, USA

Abstract. Due to the increased demand of network traffic expected during the HL-LHC era, the T2 sites in the USA will be required to have 400Gbps of available bandwidth to their storage solution. With the above in mind we are pursuing a scale test of XRootD software when used to perform Third Party Copy transfers using the HTTP protocol. Our main objective is to understand the possible limitations in the software stack to achieve the target transfer rate; to that end we have set up a testbed of multiple XRootD servers in both UCSD and Caltech which are connected through a dedicated link capable of 400 Gbps end-to-end. Building upon our experience deploying containerized XRootD servers, we use Kubernetes to easily deploy and test different configurations of our testbed. In this work, we will present our experience doing these tests and the lessons learned.

1 Introduction

The High Luminosity LHC data collection era, planned to start in 2027, poses hard network challenges for the Tier 2 Sites in the US. All sites will require 400 Gbps burst capabilities for hours and the steady state network bandwidth consumption is expected to be approximately 100 Gbps, depending on the operational details and use of the various event-formats used in high energy physics [1]. Given these requirements, in addition to making sure that the hardware is up to par, it becomes imperative that we verify the robustness of our software stack to make sure it can support this throughput. The use of the HTTPS protocol has been steadily increasing on the internet to encrypt and secure the communication between servers which makes it a strong contender to ensure fast encryption and high security. Based on the success of HTTPS as a protocol for third-party-copy (TPC) transfers showcased in a past study [2], the adoption to use HTTPS-TPC for high energy physics data transfers has already been completed. In conjunction with XRootD [3, 4] as the storage backend, which provides high-performance, scalable fault-tolerant access to data repositories of many kinds, XRootD HTTPS-TPC transfers are presently the projected setup for exabyte scale transfers. In this paper, we test the feasibility of using this setup for the aforementioned high throughput

*e-mail: aaarora@ucsd.edu
**e-mail: jguiang@ucsd.edu
***e-mail: didavila@ucsd.edu
****e-mail: fkw@ucsd.edu
†e-mail: jbalcas@caltech.edu
‡e-mail: newman@hep.caltech.edu



	min	max	avg
🟩 xrootd-server-gc-9d4c55d78-256cr: eth0: out	4.16 kb/s	30.1 Gb/s	25.0 Gb/s
🟨 xrootd-server-gc-9d4c55d78-2km4h: eth0: out	4.24 kb/s	30.4 Gb/s	24.8 Gb/s
🟦 xrootd-server-gc-9d4c55d78-2mtrp: eth0: out	7.12 kb/s	30.1 Gb/s	24.5 Gb/s

Figure 1. Monitoring Plot showing 1 Tbps flowing out of an XRootD cluster deployed in Microsoft Azure

requirements by deploying XRootD storage servers at the University of California, San Diego (UCSD) and the California Institute of Technology (Caltech) respectively and running TPCs between them. The specifics of the deployment are highlighted in Section 3. The results are presented in Section 4, additionally a study to parameterize the throughput using the number of concurrent single stream 1 GB transfers (streams hereafter) and latency between the endpoints is presented in Section 4.2.

2 Previous Studies

In order to compare the performance of XRootD HTTPS-TPC with the GridFTP protocol [5] which was the standard for LHC transfers in the past, we deployed XRootD servers on several nodes in the National Research Platform [6] (formerly Pacific Research Platform) Kubernetes [7] Cluster connected to the wide area network (WAN) at 100 Gbps and performed TPCs among them [2]. The results revealed that XRootD HTTPS-TPC performs slightly better than GridFTP on average over high throughput links.

This result evoked the urge to test the limits of XRootD HTTPS over high-bandwidth links. A natural choice was the cloud, namely Microsoft Azure [8] in our case, which gave us access to dedicated ingresses and industry-standard processors and memory thus providing us with a high-performance testbed without disturbing production workflows. Repeating the throughput benchmarking exercise in this environment, we were able to get 1 Tbps [9] on low latency links between endpoints in the same cloud region (US-West) as showcased in Figure 1.

Building upon the insights gained from our previous tests, the next thing to do was to test this over WAN with 400 Gbps connectivity. We deployed XRootD servers on endpoints with appreciable latency between them and performed TPCs between them as detailed in the rest of this paper.

3 Setup

3.1 Deployment Details

Our testbed is comprised of two sets of servers, one at each of the participating institutions — UCSD and Caltech. The two sets of servers are geolocated 120 miles apart on the west coast of the US, with a dedicated link of 400 Gbps and a network latency of roughly 3ms connecting both sites. This link is mostly dedicated as it is only shared with other R&D projects. Each group of servers is configured as an XRootD cluster i.e. all data servers connect to a redirector that acts as a load balancer.

At Caltech we have 13 servers, the specifications of which can be seen in Table 1. At UCSD we have 2 servers with their specifications described in Table 2. In Figure 2 we can see the diagram that depicts the network connectivity between both sites. It is evident that a minimum bandwidth of 400 Gbps is available between the two sites.

Table 1. Caltech servers specifications

Name	CPU model	# of real cores	RAM GB	Total bandwidth capacity Gbps
sandie-1	2x E5-2667 v3 @ 3.20GHz	16	256	100
sandie-3	2x Silver 4110 CPU @ 2.10GHz	16	96	40
sandie-5	1x AMD 7551P @ 2Ghz	32	256	100
sandie-6	1x AMD 7551P @ 2Ghz	32	256	100
sdn-dtn-1-7	2x E5-2687W v3 @ 3.10GHz	20	128	100
sdn-dtn-2-09	2x E5-2690 v2 @ 3.00GHz	20	128	40
sdn-dtn-2-11	2x E5-2670 v3 @ 2.30GHz	24	128	100
neu-sc-01	2x E5-2667 v4 @ 3.20GHz	16	128	100
sdn-sc-03	2x E5-2667 v4 @ 3.20GHz	16	128	100
sdn-sc-04	2x E5-2667 v4 @ 3.20GHz	16	128	100
sdn-sc-05	2x E5-2667 v4 @ 3.20GHz	16	128	100
sdn-sc-06	2x E5-2667 v4 @ 3.20GHz	16	128	100
sandie-9	2x E5-2667 v3 @ 3.20GHz	16	128	100

Table 2. UCSD servers specifications

Name	CPU model	# of real cores	RAM TB	Total bandwidth capacity Gbps
k8s-gen4-01	2x AMD EPYC 7763 @ 2.4GHz	128	2	500
k8s-gen4-02	2x AMD EPYC 7763 @ 2.4GHz	128	2	700

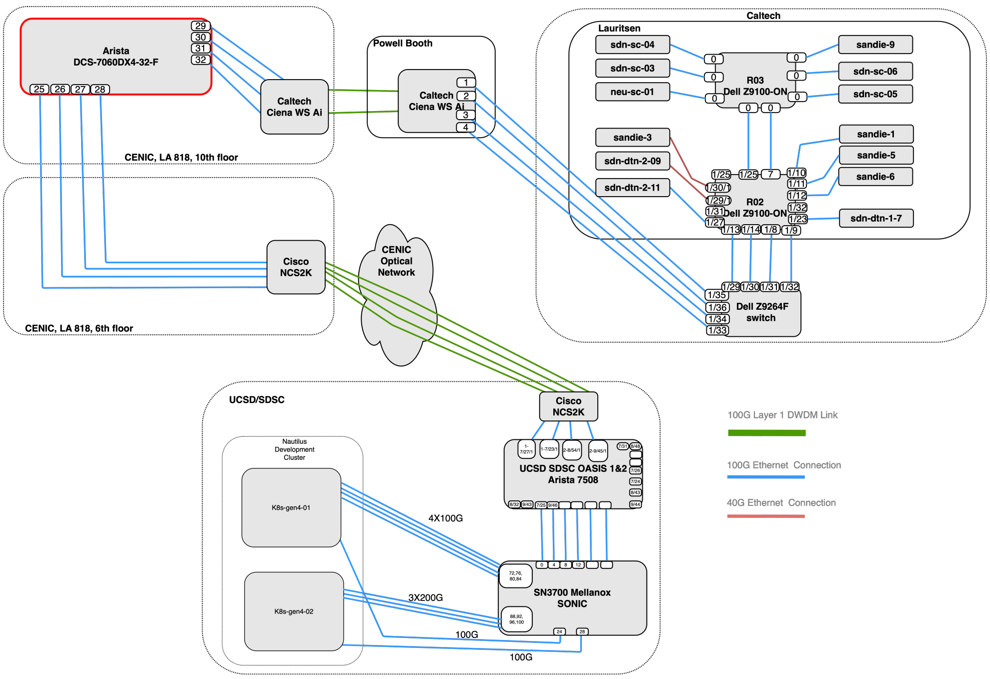


Figure 2. Network connectivity between UCSD and Caltech

All servers are deployed and managed using Kubernetes and have a non-shared, tmpfs file system mounted using the Kubernetes emptyDir volume directive with the medium defined to be Memory [10]. This helps avoid a likely bottleneck imposed by a slow file system. The test files are 1 GB each and are created using the Unix *dd* command prior to starting the XRootD service.

3.2 XRootD Configuration

The XRootD servers are configured to mimic the CMS production systems at UCSD and Caltech. The servers are configured with the XRootD http directive and use a combination of X509 and Macaroon tokens for authentication/authorization. X509 is used initially by gfal-copy [11] to obtain tokens from both source and destination sites, then the TPC is performed using only Macaroon tokens. This follows the current model used by CMS production systems.

3.3 Transfer Orchestrator

The transfers are orchestrated using a bash script running third-party-copies using gfal-copy. This script, although simplistic in design, drains a significant amount of CPU and thus has to be run on a separate server. The transfers run concurrently, with each gfal-copy client running on a separate thread launching single stream transfers. The transfers are run between the XRootD clusters using the Redirector as the primary endpoint.

4 Results

4.1 Achieving 400 Gbps

By virtue of trial and error, we determined that 40 streams per destination server (i.e. $40 \times 13 = 520$ streams coming out of UCSD) were required to saturate the bandwidth of the network link between UCSD and Caltech. Configuring our TPC orchestrator to produce this number of active transfers continuously we were able to generate and maintain an aggregated throughput of 400 Gpbs flowing through our testbed. Figure 3 shows the network traffic going out of the PortChannel towards Caltech on the Arista switch (marked in red in Figure 2). The throughput was sustained for around 1 hour and overall looks stable with little fluctuations.

4.2 Extended Study

We try to go a step further and parameterize the throughput we achieve between two endpoints by the latency between the endpoints, number of streams, number of CPU cores used by each server, and the number of XRootD origins in use. This will, in a perfect world, give us an equation for the minimum values of each of these parameters required to attain a specific throughput.

In order to do this, we systematically vary these parameters and run transfers between two identical hosts at UCSD (k8s-gen4-01 and k8s-gen4-02 as listed in Table 2). The transfers are run until the throughput stabilizes and then are killed. We use the raw latency between the hosts (which is 0.1ms) as the baseline, and simulate different latencies using Linux traffic control (tc) to introduce artificial delay by holding packets in a buffer for a specified amount of time before allowing them to be transmitted. The number of CPU cores in the cluster are controlled using Kubernetes resource allocation.

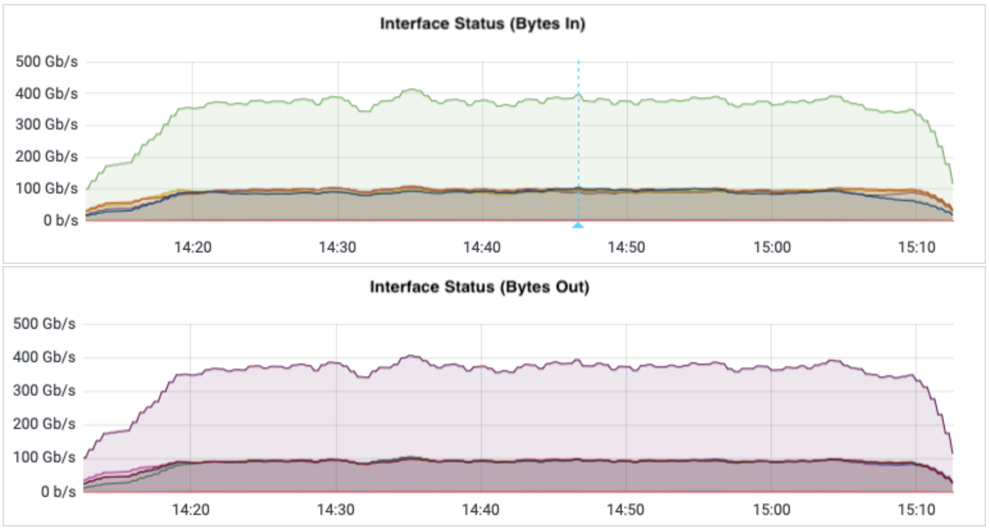


Figure 3. Monitoring plots showing 400 Gbps of throughput through the Arista switch. The plot on top shows traffic flowing into the Arista and the bottom plot shows throughput out of the port channel towards Caltech. The throughput through each separate port is showcased in the lines peaking at 100 Gbps and the line at 400 Gbps shows the aggregate.

We observed that a machine reboot was required in order to completely reverse the network delays introduced through the use of the tc command.

Each server uses its own network interface card (NIC) which has a bandwidth capacity of 100 Gbps, therefore increasing the number of XRootD origins increases the total bandwidth capacity accordingly. The number of CPU cores are aggregated over all origins in the cluster. The total number of streams are distributed across the origins, i.e. 100 streams with 4 servers would mean each server transferring using 25 streams. The results are showcased in the plots in Figure 4

4.3 More Lessons Learned

4.3.1 Overhead due to using Redirector

Using the Redirector for load balancing as opposed to point to point transfers between the origins, even for large number of concurrent transfers causes no major overhead. Performance difference between the scenario where the Redirector manages which origin the request goes to, as opposed to when the source and destination server pairs are chosen by hand is negligible thus making it evident that using the Redirector is feasible.

4.3.2 Choice of client transfer tool

The choice of client transfer tool does make a slight difference. When testing the TPCs using gfal-copy and curl, we see 15% better performance using gfal-copy. Thus other production tools like FTS [12] that employ the gfal client for TPCs should give performance similar to what we see.

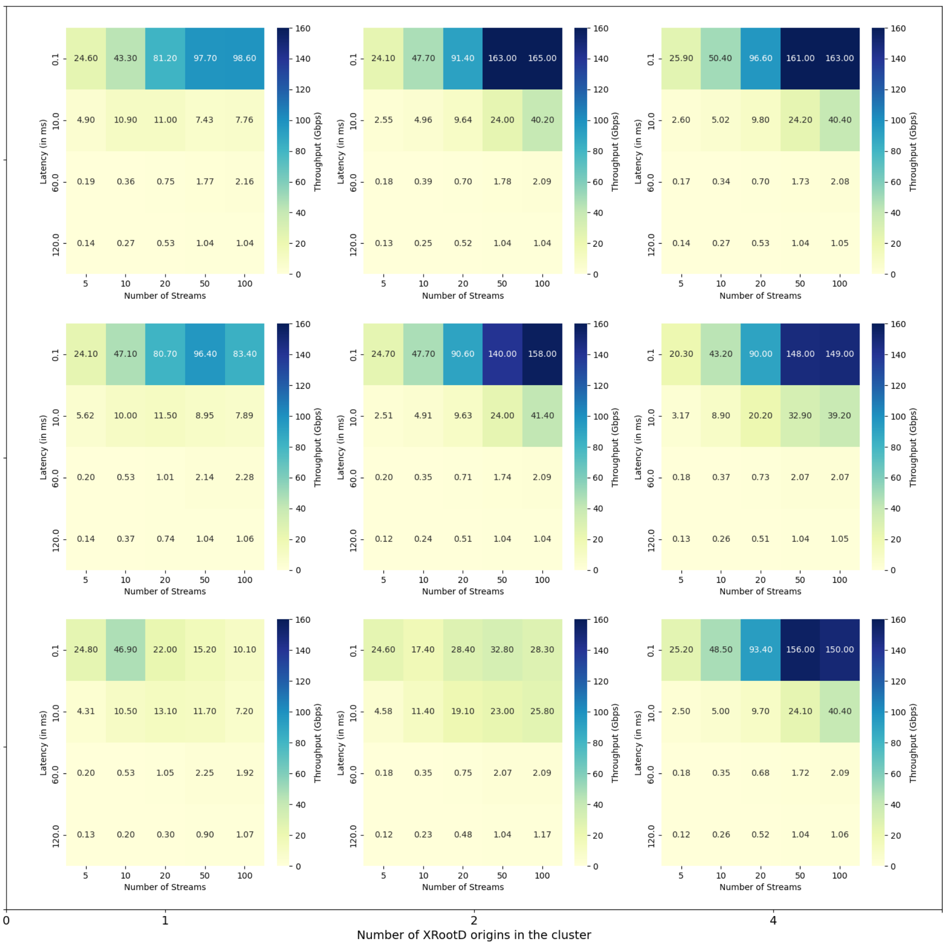


Figure 4. Plots showing Throughput (in Gbps) as a function of Number of streams and Latency (in ms) for different combinations of CPU cores and XRootD origins

5 Conclusion

5.1 Extended Study Results Interpretation

As was observed in our previous study [2], the relationship between throughput and number of streams holds. As can be seen in all the plots in Figure 4, for a low number of CPU cores, the throughput initially increases with the number of streams and then falls when the CPU usage is very high. This is likely due to resource contention or slow packet processing. Therefore, there is a maximum throughput that can be achieved using a given number of CPU cores.

Next we see that, adding more origins to the cluster even while keeping the number of cores the same (i.e. traversing in the rightward direction in any row in the plot) provides better performance which might owe to better kernel load balancing due to a higher number of processes.

We also conclusively see that throughput decreases severely with increasing latency. Going from 0.1 to 10 ms causes a five fold decrease in the amount of data transferred. It is therefore evident, as we would expect, that in order to achieve high throughput at very high latency, a very large number of streams would be required. We must note, however, that we do not see the theoretical linear scaling of throughput with latency whilst keeping the number of streams constant. This might point to the fact that tc is not ideal for delay emulation in high bandwidth tests. This suspicion is further deepened by the fact that repeating the transfer test with two origins, each having access to all 128 cores on the host and pushing up to 500 streams, the throughput at 120ms of latency is still restricted to around 1 Gbps.

5.2 Overall Conclusion

The results show that XRootD HTTPS-TPC is capable of sustaining the high throughput required in the High Luminosity LHC era at low latencies. The trends in our study seem to indicate that with the right combination of streams, number of cpu cores and number of origins in the cluster, the desired throughput can be achieved for higher latencies, but more work is required to be completely certain.

6 Future Work

More work is required to understand the deviation from the expected trend when we increase the latency from 10ms to 60ms and also why throughput scaling with the number of streams stops at higher latencies.

7 Acknowledgement

The authors would like to thank CENIC for providing us with the network bandwidth that made this work possible, ARISTA for donating to us the switch in CENIC LA 818, 10th floor, and the funding agencies to support this effort, in particular the National Science Foundation through the following grants: OAC-1836650, PHY-2323298, OAC-2030508 and OAC-2112167

References

[1] D. Carder, E. Dart, M. Graf, C. Hawk, A. Holder, D. Jacob, et al., *Basic energy sciences network requirements review (final report)* (2022), https://escholarship.org/uc/item/3jj0h54n

[2] E. Fajardo, A. Arora, D. Davila, R. Gao, F. Würthwein, B. Bockelman, *Systematic benchmarking of https third party copy on 100gbps links using xrootd* (2021), https://doi.org/10.1051/epjconf/202125102001

[3] *Xrootd*, https://xrootd.slac.stanford.edu

[4] A. Dorigo, P. Elmer, F. Furano, A. Hanushevsky, *Xrootd-a highly scalable architecture for data access* (2005)

[5] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, *The Globus Striped GridFTP Framework and Server*, in *SC '05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing* (2005), pp. 54–54

[6] *National research platform*, https://nationalresearchplatform.org

[7] *Kubernetes documentation*, https://kubernetes.io/docs/home/

[8] *Microsoft azure*, https://azure.microsoft.com/en-us

[9] A. Arora, I. Sfiligoi, D. Davila, F. Würthwein, *Using Microsoft Azure for XRootD network benchmarking* (2021), https://doi.org/10.5281/zenodo.8361532

[10] Kubernetes, *Kubernetes - volumes*, accessed: Sept. 18, 2023, `https://kubernetes.io/docs/concepts/storage/volumes/`

[11] *gfal*, `https://dmc-docs.web.cern.ch/dmc-docs/gfal2/gfal2.html`

[12] A. Ayllon, M. Salichos, M. Simon, O. Keeble, *FTS3: new data movement service for WLCG*, in *Journal of Physics: Conference Series* (IOP Publishing, 2014), Vol. 513, p. 032081