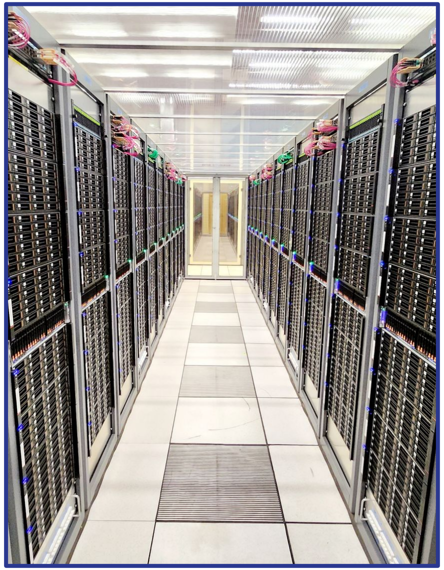
Physics Data Forge: Unveiling the Power of I/O Systems in CERN's Test Infrastructure

G. Amadio, A. Sciabà, A. Peters, D. Smith, L. Mascetti

CHEP 2024, Kraków, Poland

XRootD and the Path to HL–LHC

- Remote data access is critical in high energy physics (HEP)
  - XRootD and EOS are core components in HEP’s ecosystem
- Importance for the high luminosity LHC (HL–LHC)
  - Expected to generate **10x more data** than current LHC
  - XRootD and EOS need to be able to manage this data deluge
- Project between IT and EP–SFT on RNTuple Evaluation
  - Not only about RNTuple, but also verification of storage backend
  - Verify performance and scalability of large analysis workflows
- Network upgrades from 25/100G to 100/400G in the future
  - Different behavior than usual 1G to 10G networking
- Benchmark XRootD and HTTP clients in ideal setup
  - Ensure the software is not the bottleneck with new data rates



Are we ready for more than 50GB/s data rates?

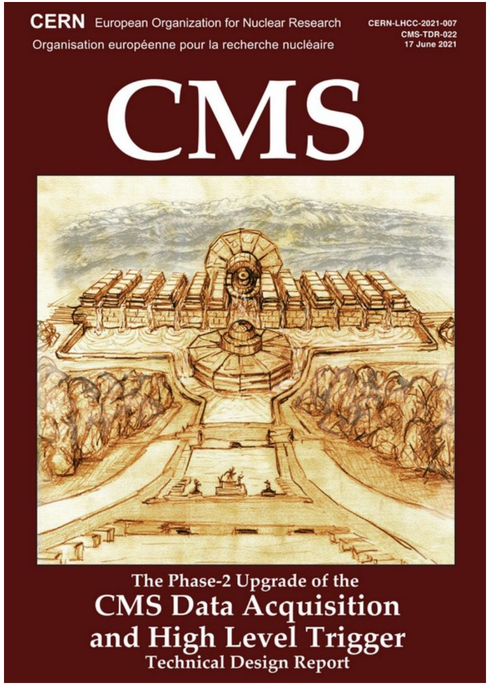


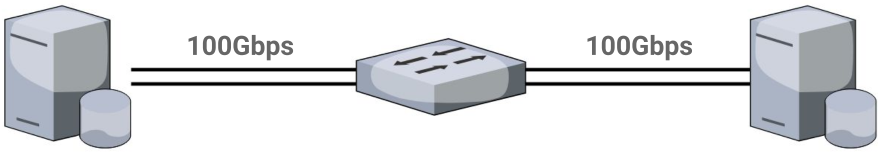
Table 1.2: CMS Phase-2 trigger and DAQ projected running parameters, compared to the design values of the current (Phase-1) system.

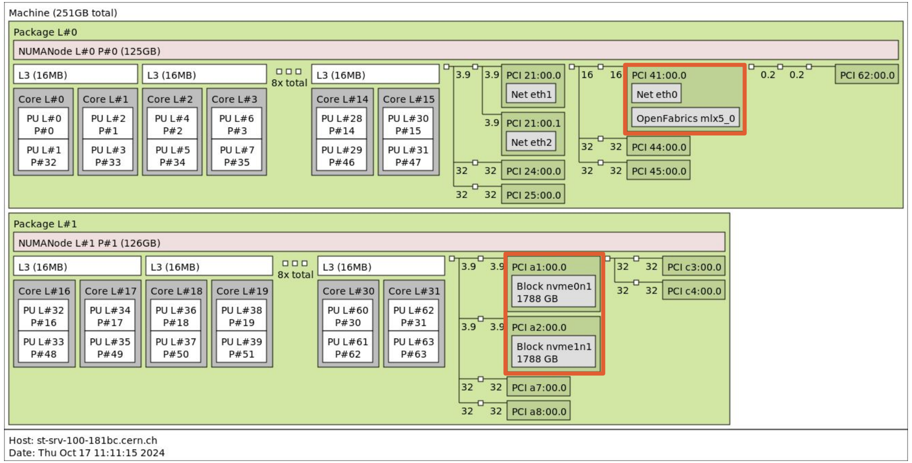
	LHC	HL-LHC	
CMS detector	Phase-1	Phase-2	
Peak ⟨PU⟩	60	140	200
L1 accept rate (maximum)	100 kHz	500 kHz	750 kHz
Event Size at HLT input	2.0 MB [a]	6.1 MB	8.4 MB
Event Network throughput	1.6 Tb/s	24 Tb/s	51 Tb/s
Event Network buffer (60 s)	12 TB	182 TB	379 TB
HLT accept rate	1 kHz	5 kHz	7.5 kHz
HLT computing power [b]	0.7 MHS06	17 MHS06	37 MHS06
Event Size at HLT output [c]	1.4 MB	4.3 MB	5.9 MB
Storage throughput [d]	2 GB/s	24 GB/s	51 GB/s
Storage throughput (Heavy-Ion)	12 GB/s	51 GB/s	51 GB/s
Storage capacity needed (1 day [e])	0.2 PB	1.6 PB	3.3 PB

[a] Design value.
[b] Does not include Data Quality Monitoring.
[c] Actual compression factor for Phase-1. For Phase-2 same factor is assumed, see Section 6.2.11.
[d] The storage throughput is defined as the effective throughput with concurrent recording and transfer. The throughput required is determined by the HLT output event size and the additional output streams, see Section 6.2.11.
[e] Assuming an LHC duty cycle, i.e. the fraction of time spent in stable colliding beams, of 75 %.

CERN Testing Setup

- Hardware Configuration
  - Two high-performance nodes
    - Dual AMD EPYC 7302 16-Core CPU
    - Mellanox ConnectX-5 NIC (100Gbps)
    - 256GB Memory, 2 x 2TB NVMe SSD
    - Alma Linux 8.10
    - Linux 4.18.0-553.22.1.el8_10
    - OpenSSL 1.1.1k
- Node 1: XRootD 5.7.1 Server
  - 128GB tmpfs mount point for data
- Node 2: XRootD / HTTP Clients
  - XRootD 5.7.1
  - Davix 0.8.7, curl 7.61.1, wget 1.19.5
  - OpenSSH 8.0p1 (scp)





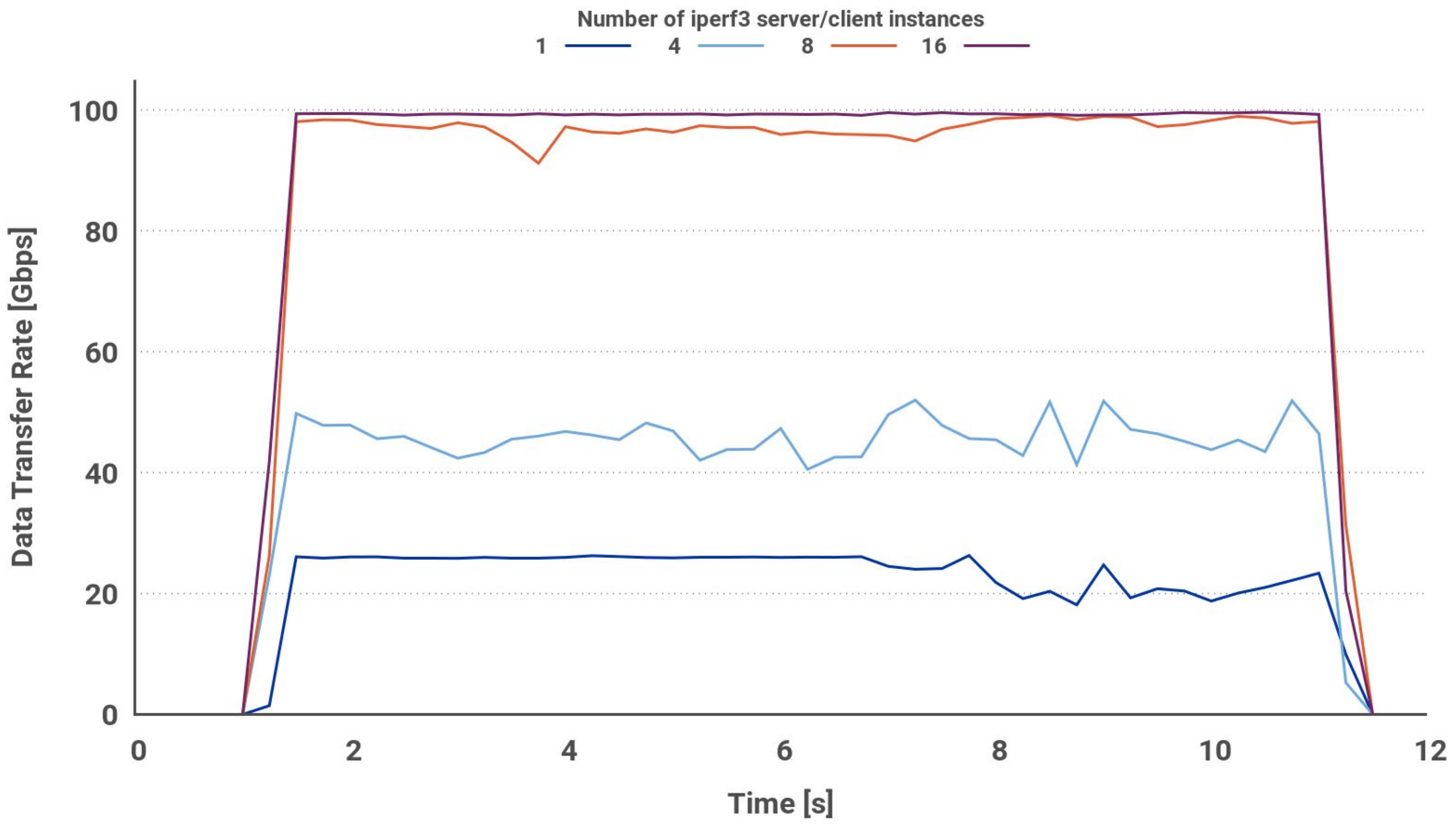
Network Tuning for 100GbE NICs

- Applied “standard” tunings for 100Gbps
- MTU (Maximum Transmission Unit)
  - Switch from 1500 to 9000
- TCP Congestion Control Algorithm
  - Using **bbr** algorithm
- TCP Optimizations
  - Increase window size
  - Increase read/write buffer size
- Increase NIC ring buffer size
  - `ethtool -gG eth0`

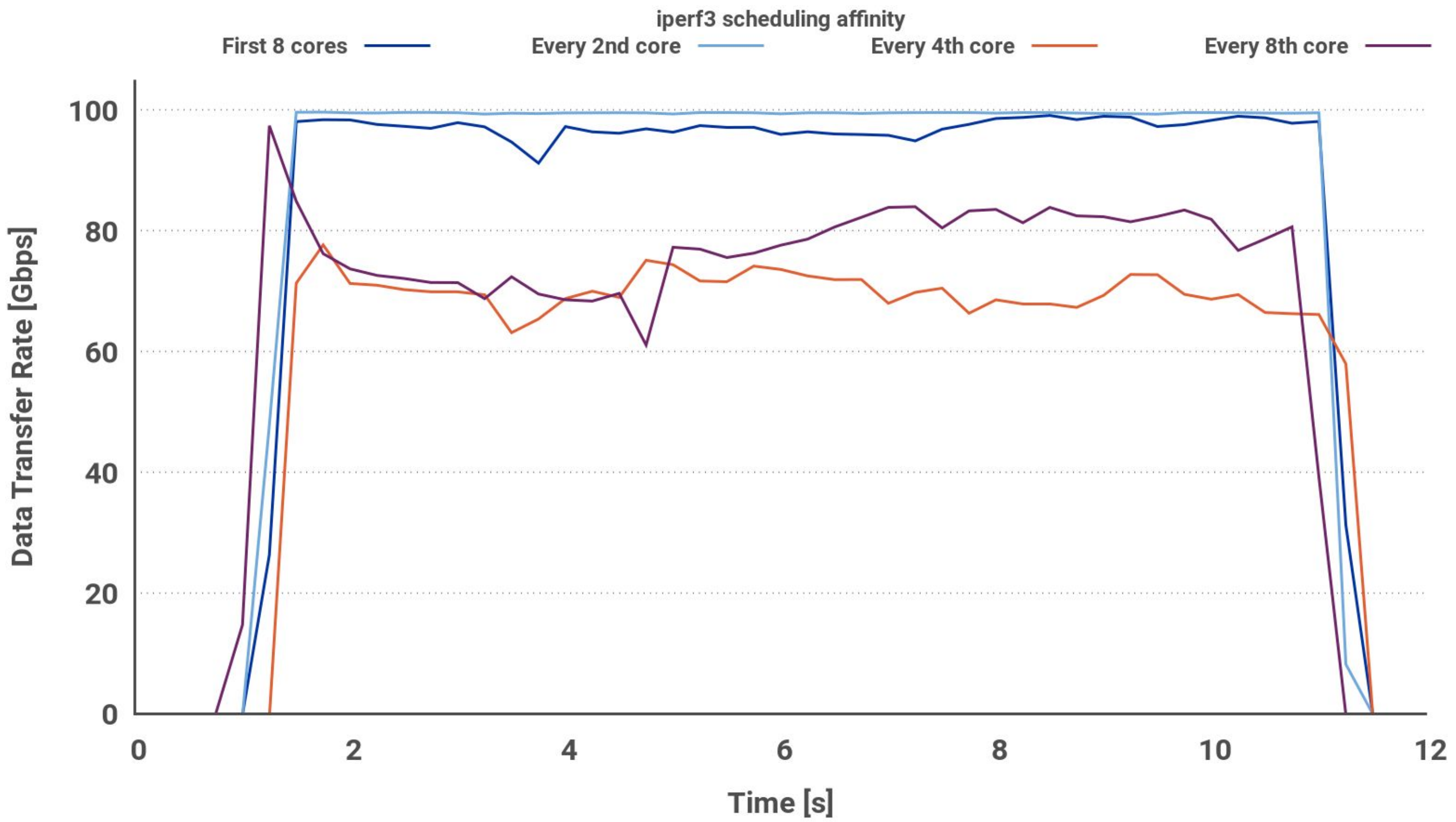
```
# sysctl -p
net.ipv4.tcp_wmem = 4096 65536 2147483647
net.ipv4.tcp_rmem = 4096 87380 2147483647
net.core.rmem_max = 2147483647
net.core.wmem_max = 2147483647
net.core.default_qdisc = fq
net.ipv4.tcp_congestion_control = bbr
net.ipv4.tcp_mtu_probing = 1
net.core.optmem_max = 1048576

# ethtool -g eth0
Ring parameters for eth0:
Pre-set maximums:
RX:             8192
RX Mini:  n/a
RX Jumbo:       n/a
TX:             8192
Current hardware settings:
RX:             8192
RX Mini:  n/a
RX Jumbo:       n/a
TX:             8192
```

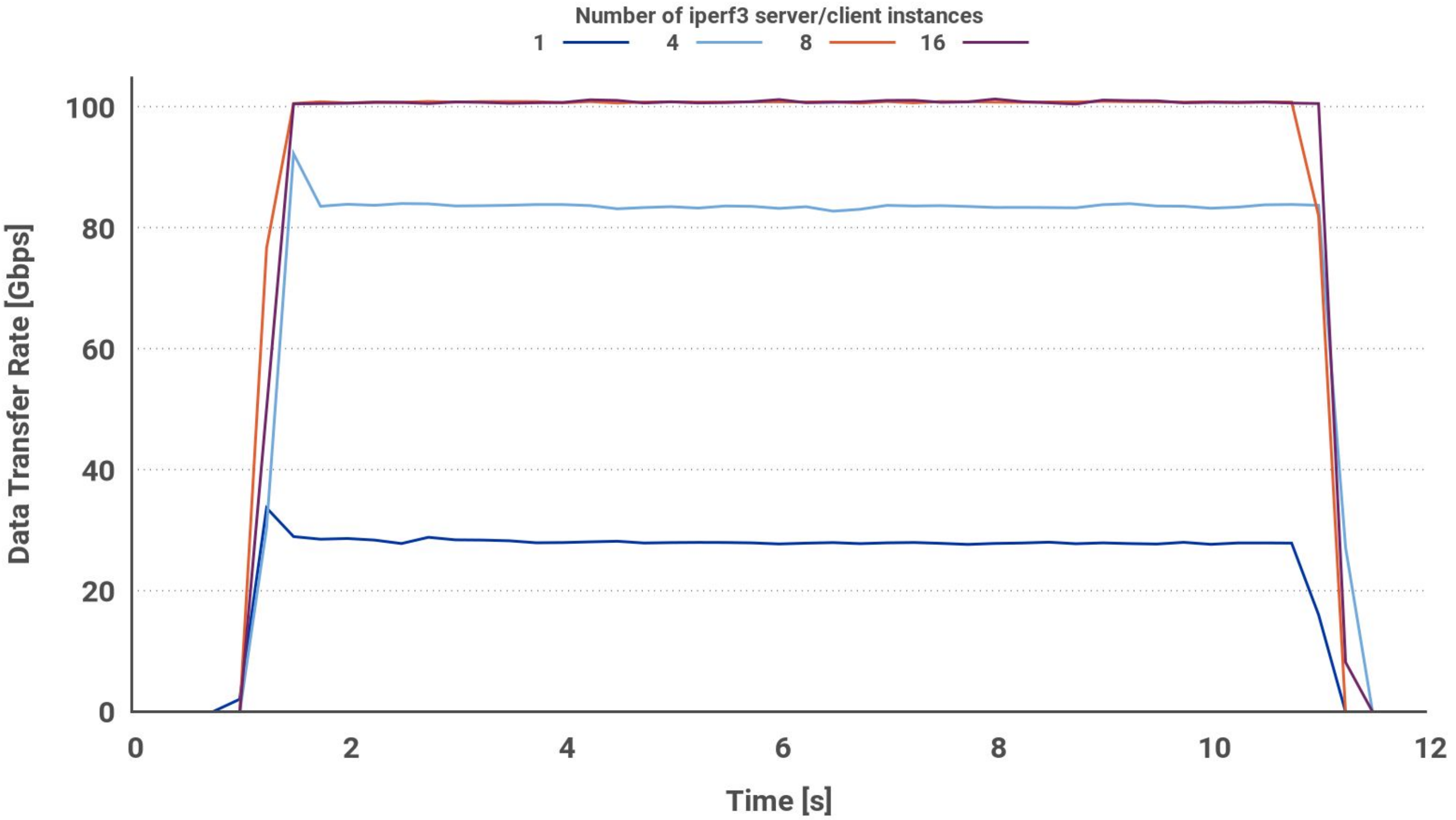
Network Speed Verification with iperf3 (MTU=1500)



Effect of Scheduling Affinity (8x iperf3 server/client)



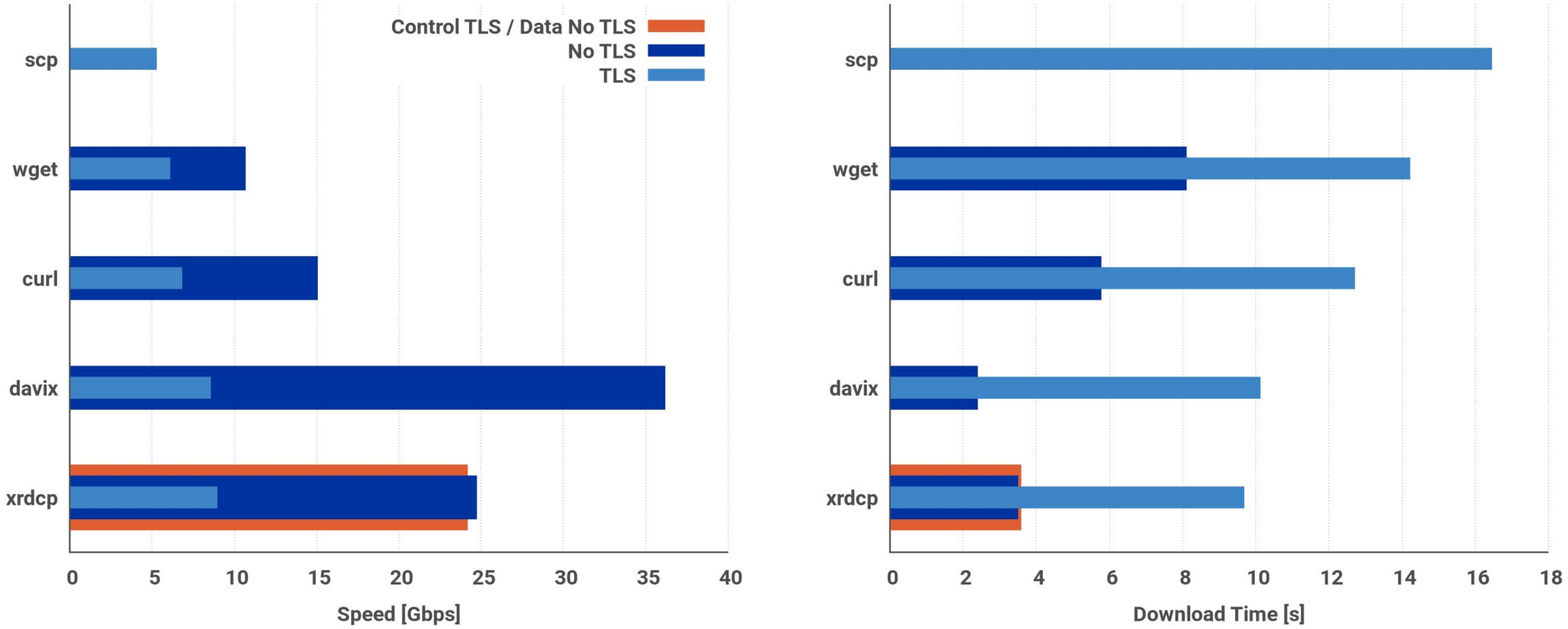
Effect of Jumbo Frames (MTU = 9000)



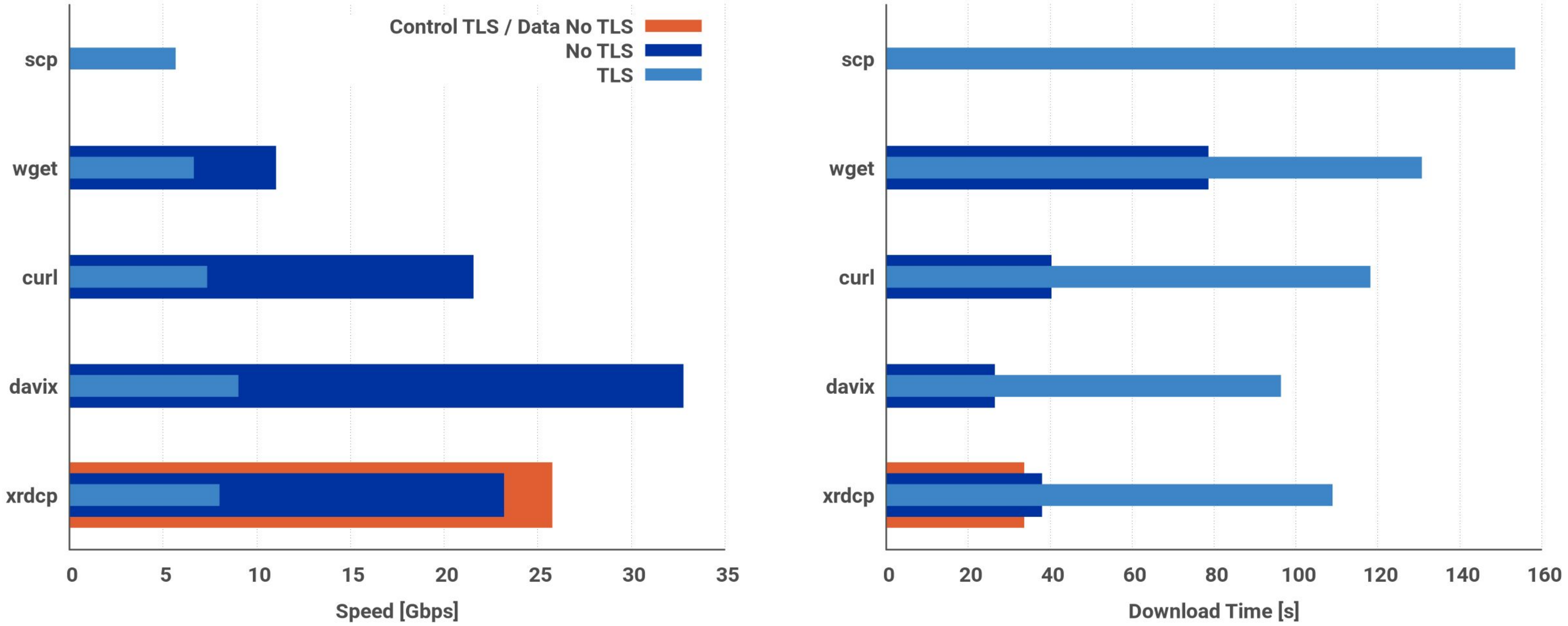
XRootD and HTTP Client Benchmarks

- Compare download speed of 10GB and 100GB files with random data
  - Files are in 128GB tmpfs mount, exposed via XRootD server
  - No authentication is used for these tests
  - However, some tests use TLS encryption
  - Link saturation achieved by running concurrent transfers
  - Downloaded file is “written” to `/dev/null` to avoid bottlenecks from storage devices
- Test multiple data stream support from XRootD client
  - Stream 0 is control stream, up to 15 additional data streams for up to 16 total streams
  - TLS encryption can be applied to all streams or control stream only, we test both cases
  - PgRead/PgWrite has an effect on performance, so we test with it enabled/disabled as well

Benchmark: Download 10GB File

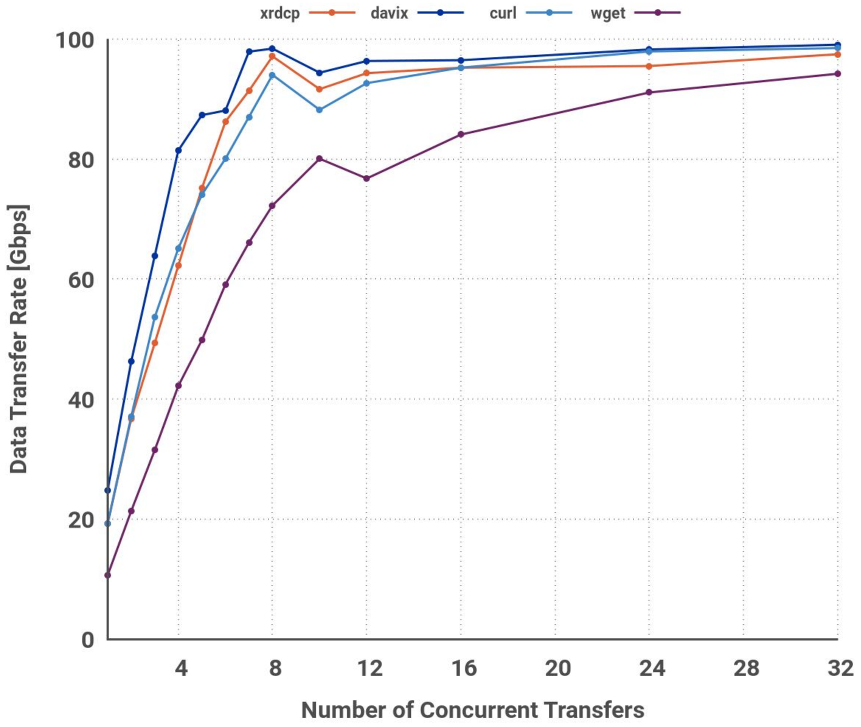


Benchmark: Download 100GB File

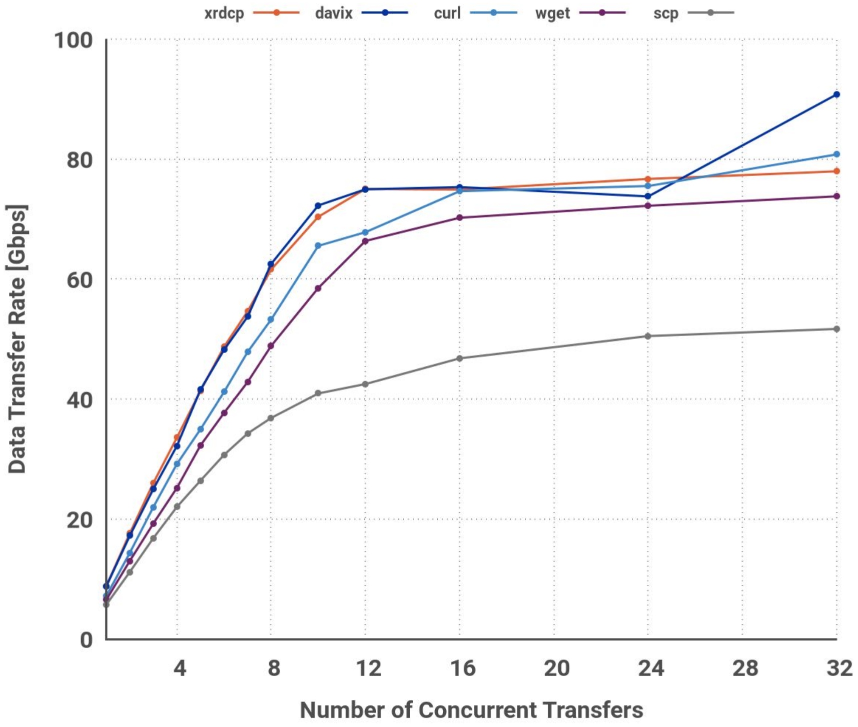


Link Saturation with Concurrent 10GB File Transfers

No Encryption

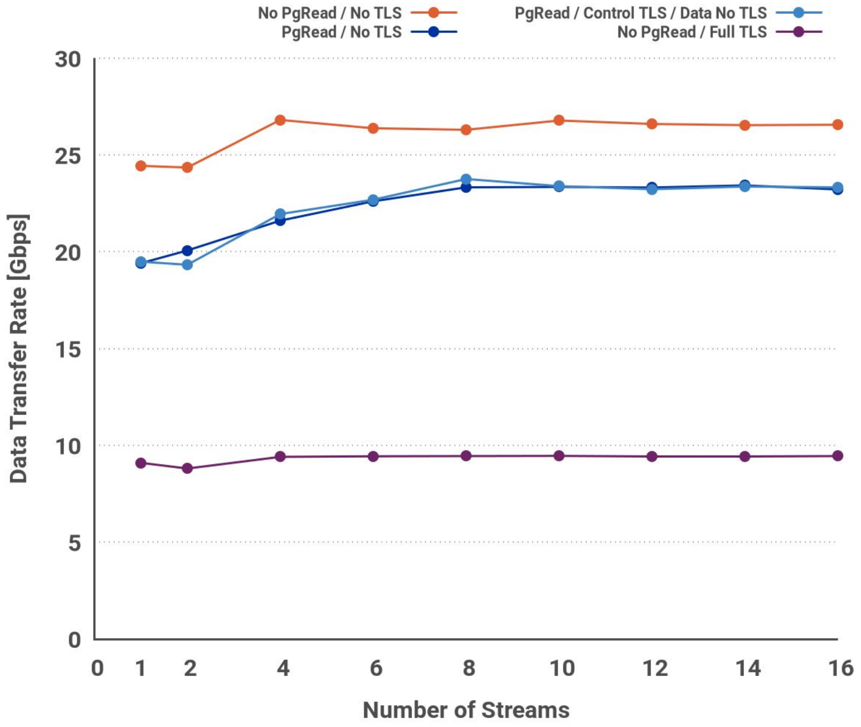


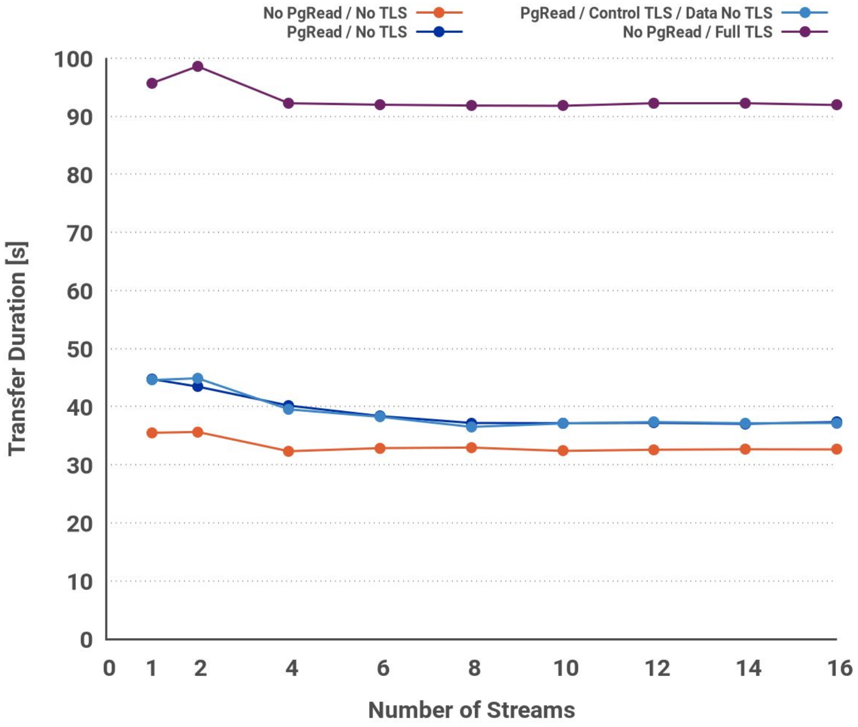
TLS Encryption



XRootD Client Performance with Multiple Streams

Download speed with xrdcp did not improve very much when adding streams.





Bug Report on GitHub Hints at the Solution

xrdcp with single socket vs parallel sockets vs extreme copy mode #1938

Edit | New issue

Open · **apeters1971** opened this issue on Mar 1, 2023 · 8 comments

apeters1971 commented on Mar 1, 2023 — Member

I did some **100GE** benchmarks and I notice the following when comparing parallel sockets vs extreme copy.

The single stream copy :

```
[root@node]# time xrdcp -y 1 e.meta4 /dev/null -f
[9.766GB/9.766GB][100%][==================================================][2.441GB/s]

real    0m4.887s
user    0m1.523s
sys     0m4.945s
```

The parallel socket implementation using 10 sockets:

```
[root@node]# time xrdcp -S 10 e.meta4 /dev/null -f
[9.766GB/9.766GB][100%][==================================================][1.953GB/s]

real    0m4.964s
user    0m1.688s
sys     0m4.998s
```

An extreme copy by using 10 named connections to the same xrootd server:

```
[root@node]# time xrdcp -y 10 e.meta4 /dev/null -f
[9.766GB/9.766GB][100%][==================================================][9.766GB/s]

real    0m0.947s
user    0m2.685s
sys     0m7.357s
```

It is fantastic, that I can run a single copy with 10 GB/s but I have to do some gymnastic to get this, while the easy defaults with single or parallel sockets have a much lower limit.

Maybe one could use the same switch for implicit extreme copy mode to specify the number of connections if the source is not a meta link file?

Assignees
amadio

Labels
enhancement

Projects
None yet

Milestone
No milestone

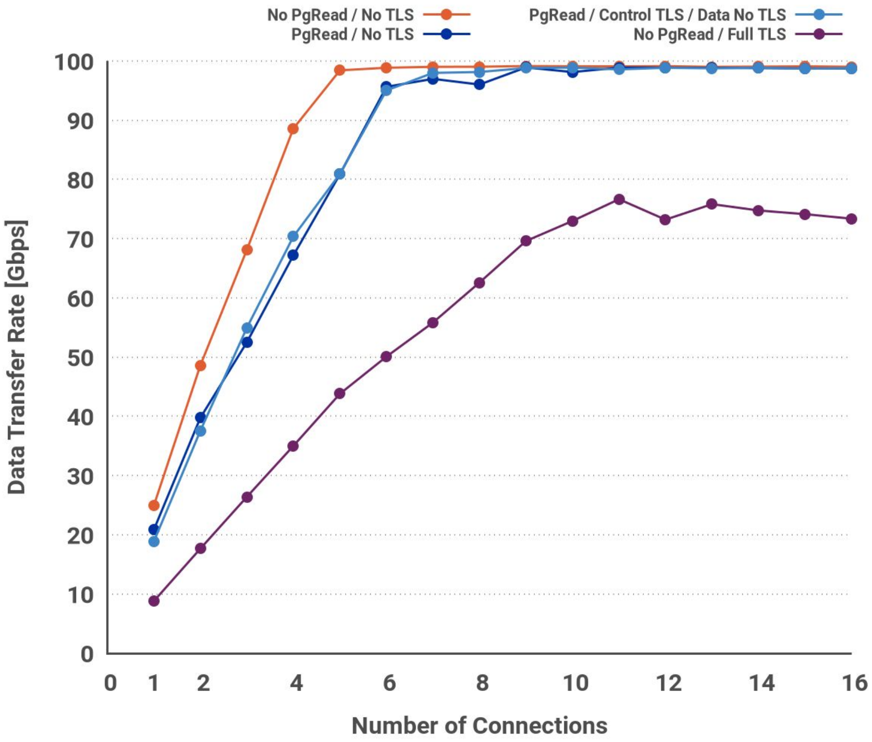
Development
Create a branch for this issue or link a pull request.

Notifications — Customize
Unsubscribe
You're receiving notifications because you're watching this repository.

4 participants

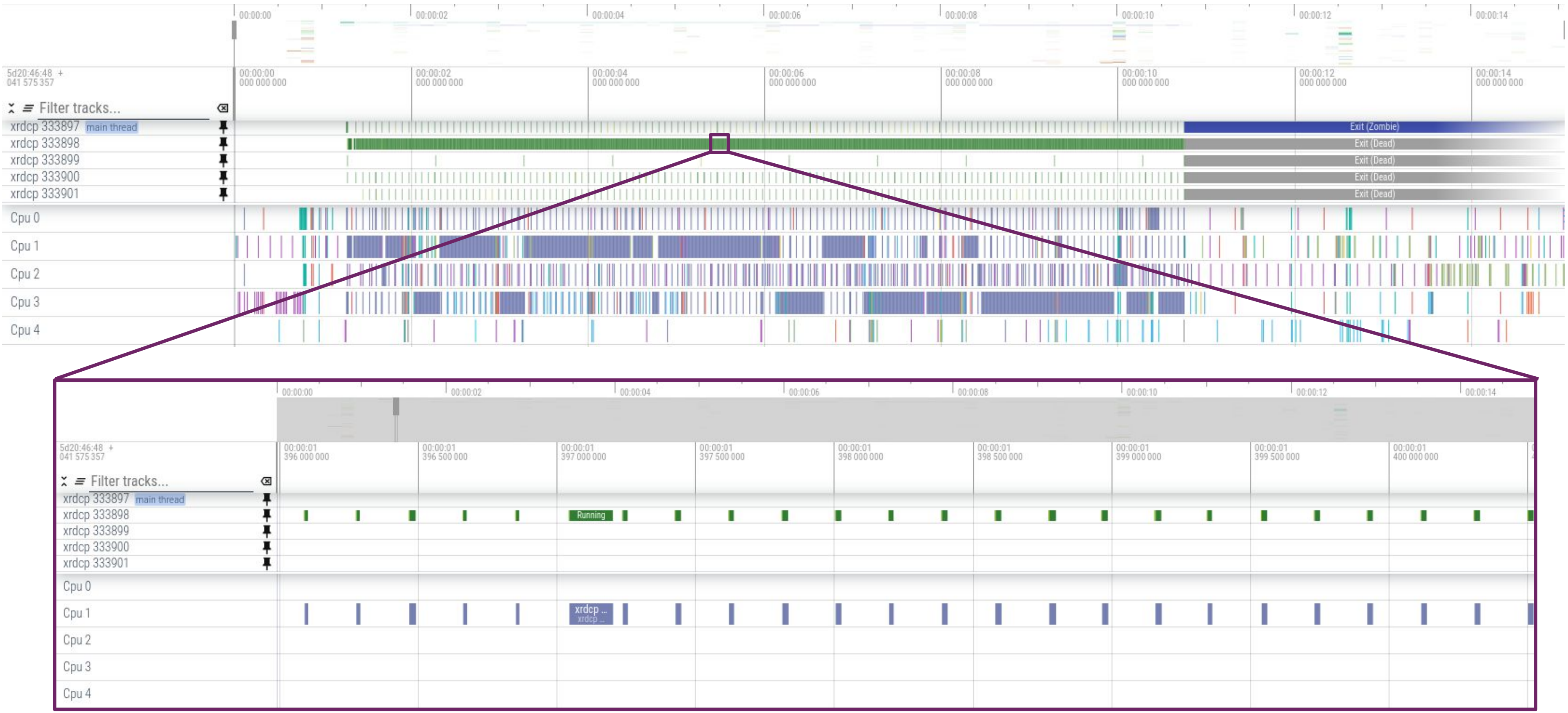
Lock conversation
Pin issue
Transfer issue
Convert to discussion
Delete issue

XRootD Client Extreme Copy

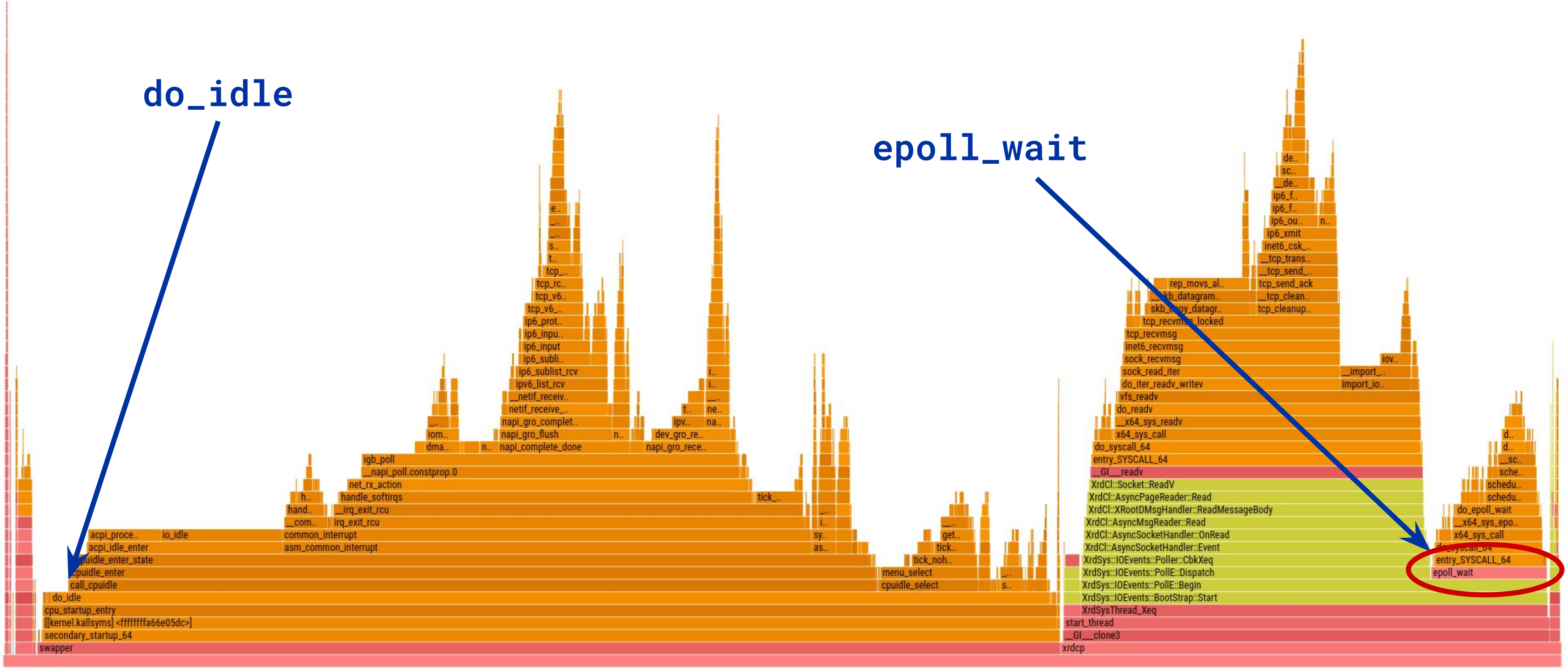



```
<?xml version="1.0" encoding="UTF-8"?>
<metalink version="3.0" xmlns="http://www.metalinker.org/">
 <files>
  <file name="file100G.raw">
   <resources>
    <url type="file" location="ch" preference="1">root://a@xrootd-server.cern.ch//file100G.raw</url>
    <url type="file" location="ch" preference="1">root://b@xrootd-server.cern.ch//file100G.raw</url>
    <url type="file" location="ch" preference="1">root://c@xrootd-server.cern.ch//file100G.raw</url>
    <url type="file" location="ch" preference="1">root://d@xrootd-server.cern.ch//file100G.raw</url>
    <url type="file" location="ch" preference="1">root://e@xrootd-server.cern.ch//file100G.raw</url>
    <url type="file" location="ch" preference="1">root://f@xrootd-server.cern.ch//file100G.raw</url>
    <url type="file" location="ch" preference="1">root://g@xrootd-server.cern.ch//file100G.raw</url>
    <url type="file" location="ch" preference="1">root://h@xrootd-server.cern.ch//file100G.raw</url>
    <url type="file" location="ch" preference="1">root://i@xrootd-server.cern.ch//file100G.raw</url>
    <url type="file" location="ch" preference="1">root://j@xrootd-server.cern.ch//file100G.raw</url>
    <url type="file" location="ch" preference="1">root://k@xrootd-server.cern.ch//file100G.raw</url>
    <url type="file" location="ch" preference="1">root://l@xrootd-server.cern.ch//file100G.raw</url>
    <url type="file" location="ch" preference="1">root://m@xrootd-server.cern.ch//file100G.raw</url>
    <url type="file" location="ch" preference="1">root://n@xrootd-server.cern.ch//file100G.raw</url>
    <url type="file" location="ch" preference="1">root://o@xrootd-server.cern.ch//file100G.raw</url>
    <url type="file" location="ch" preference="1">root://p@xrootd-server.cern.ch//file100G.raw</url>
   </resources>
  </file>
 </files>
</metalink>
```

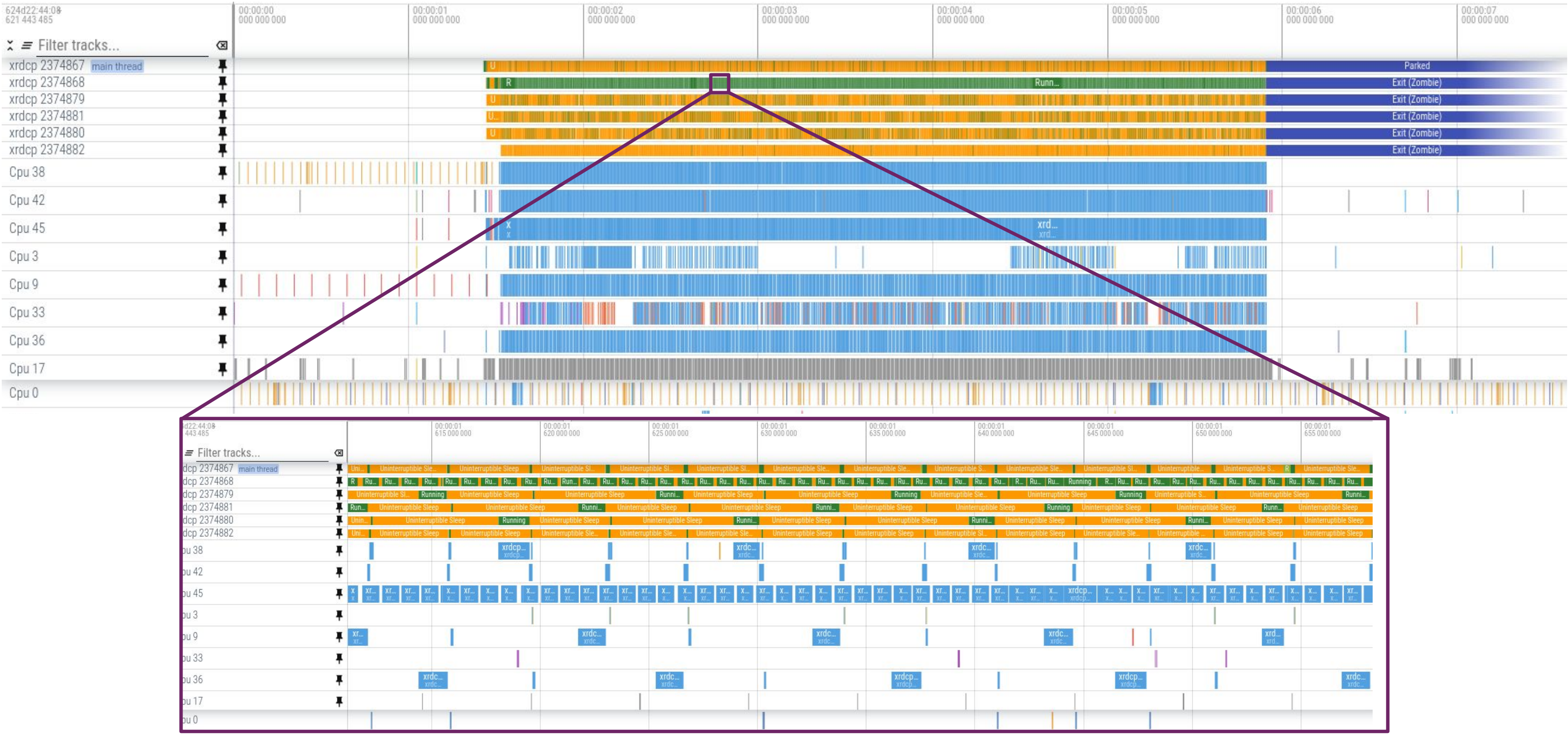
Trace: File Download with xrdcp (1GbE)



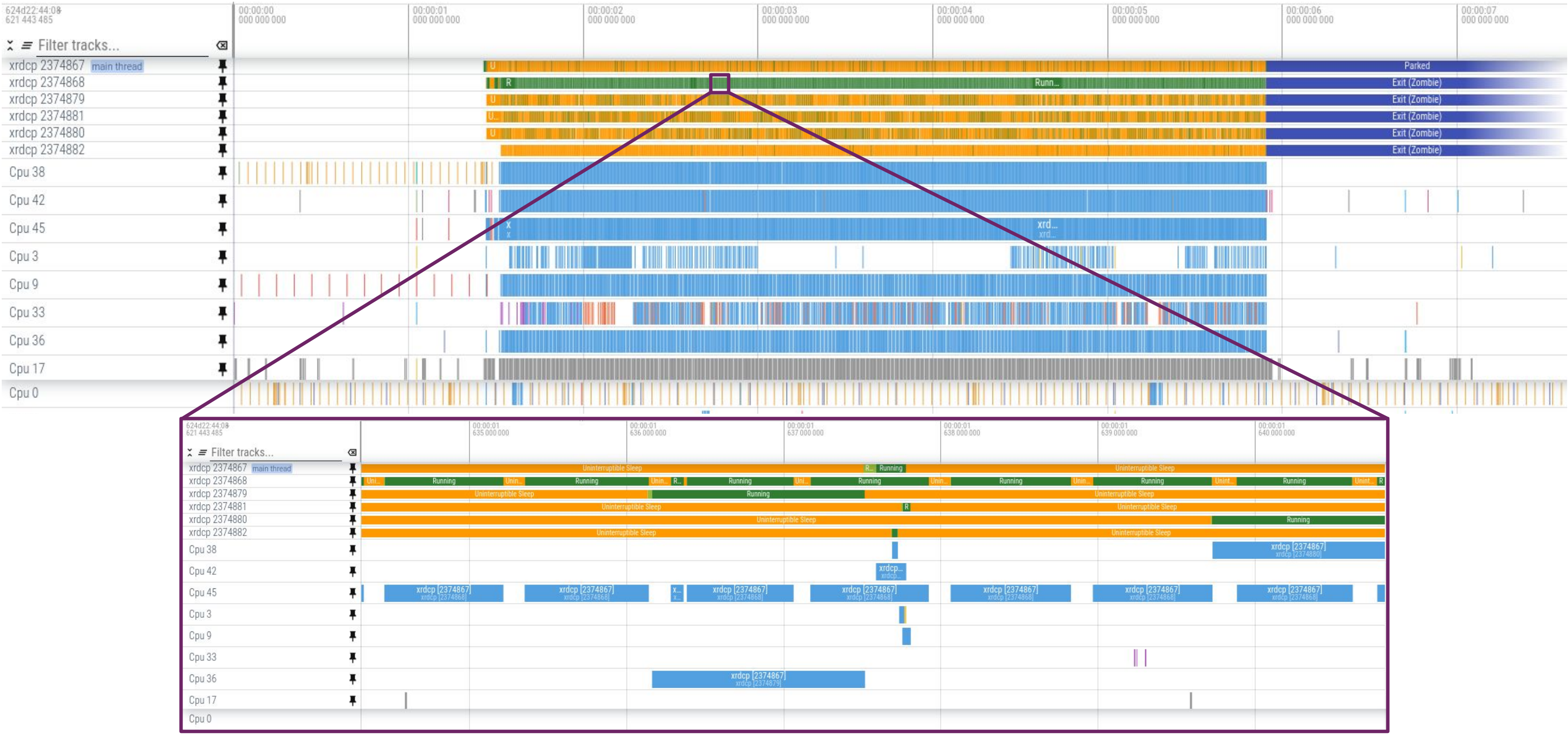
Flamegraph: File Download with xrdcp (1GbE)



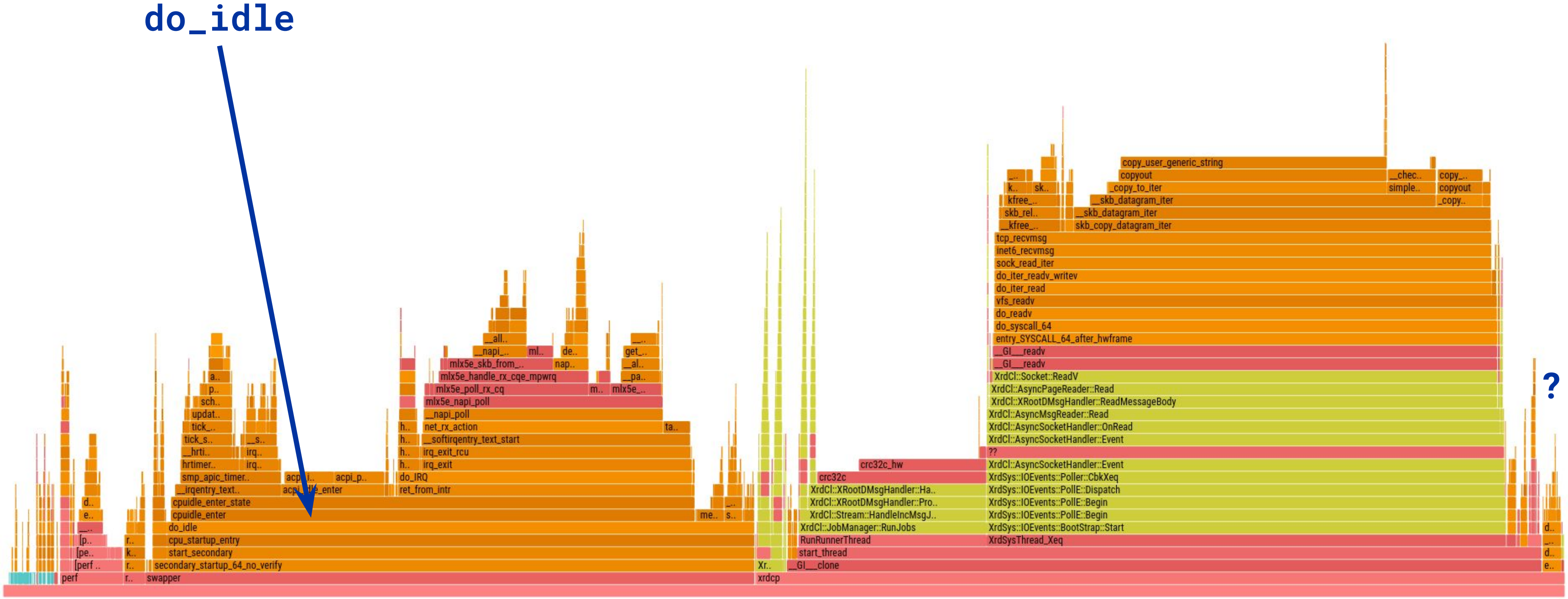
Trace: File Download with xrdcp (100GbE)



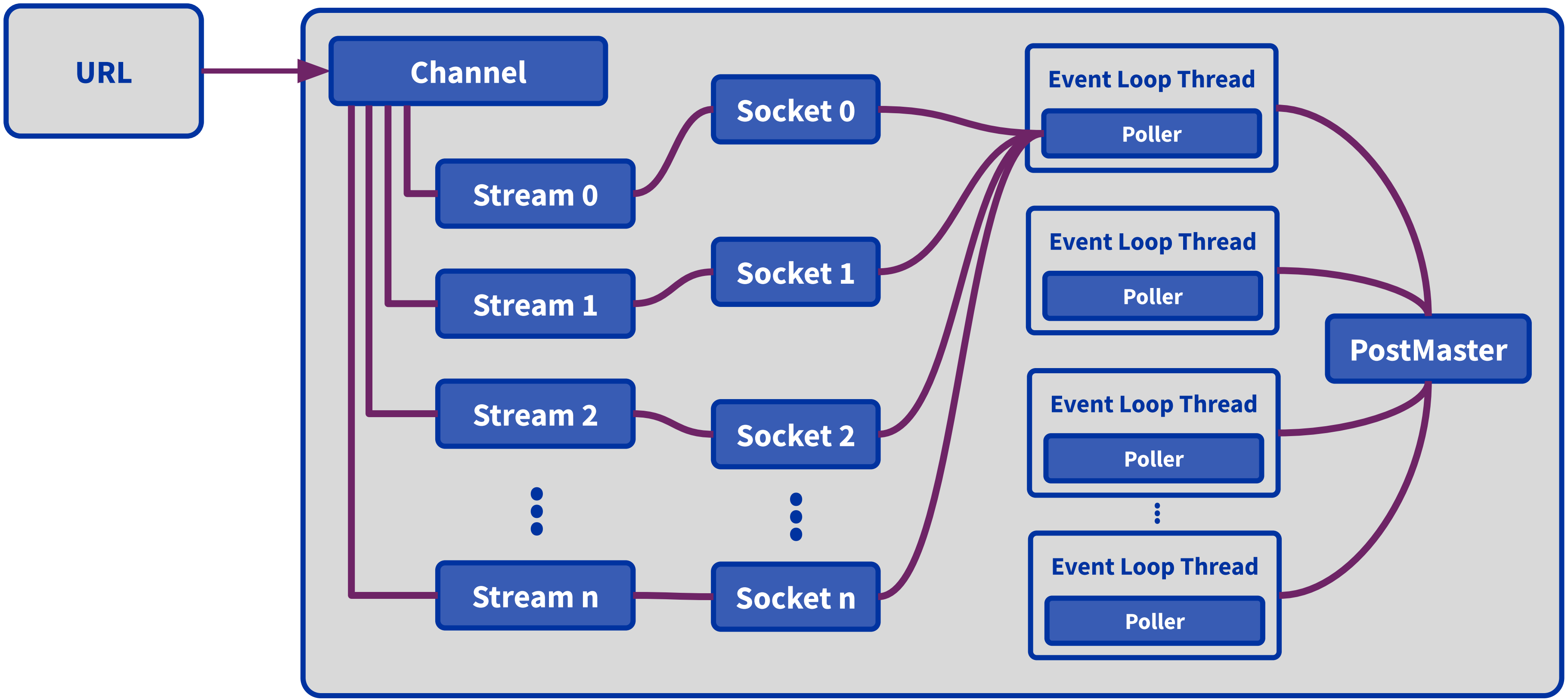
Trace: File Download with xrdcp (100GbE)



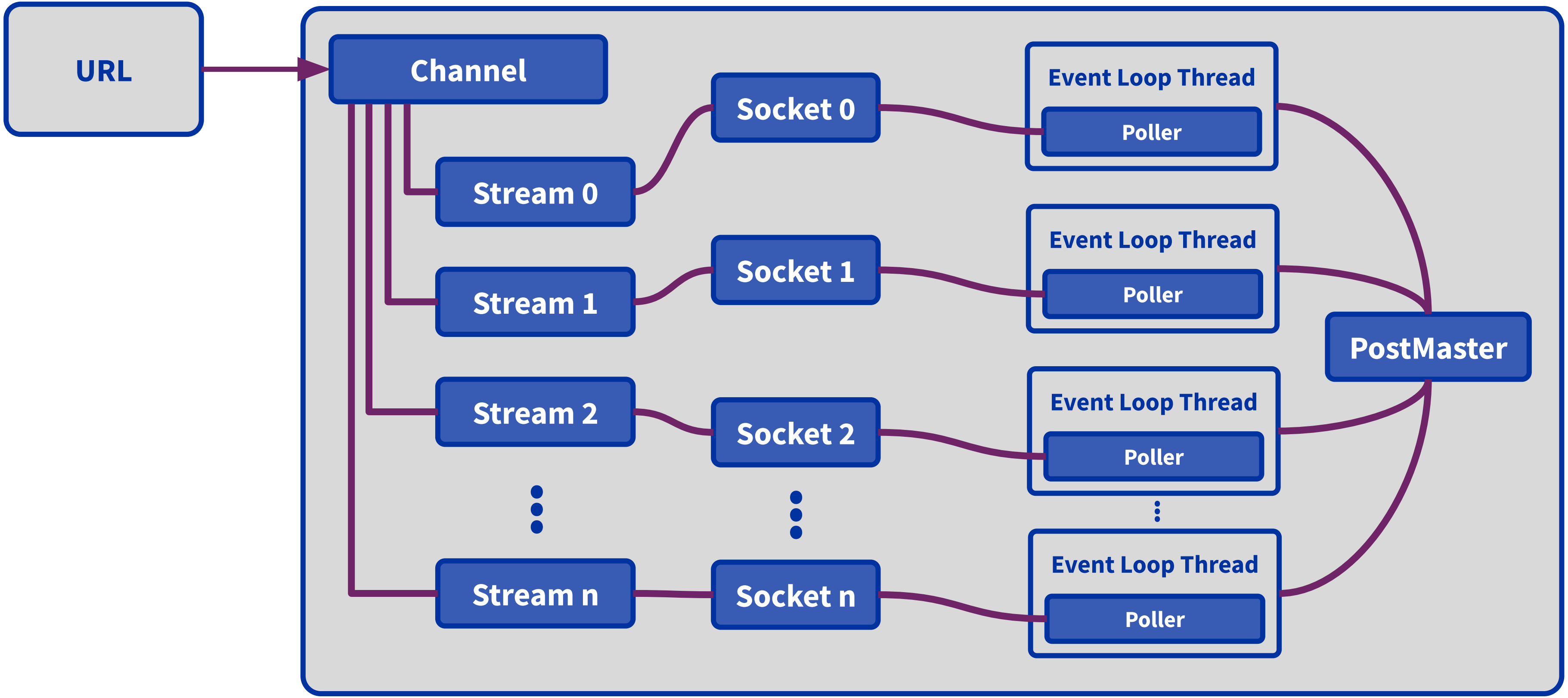
Flamegraph: File Download with xrdcp (100GbE)



XRootD Client Current Socket to Poller Mapping

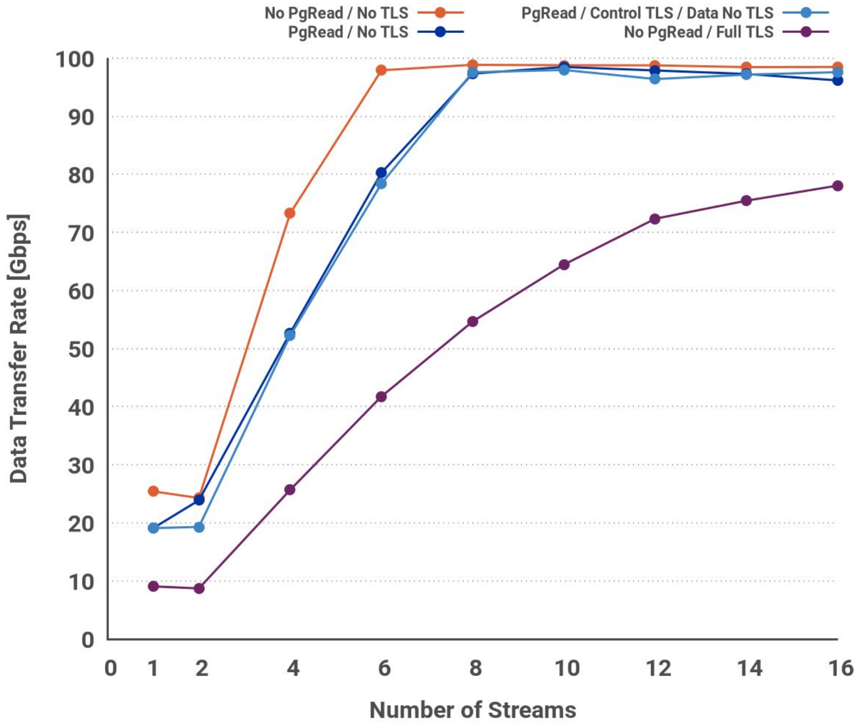


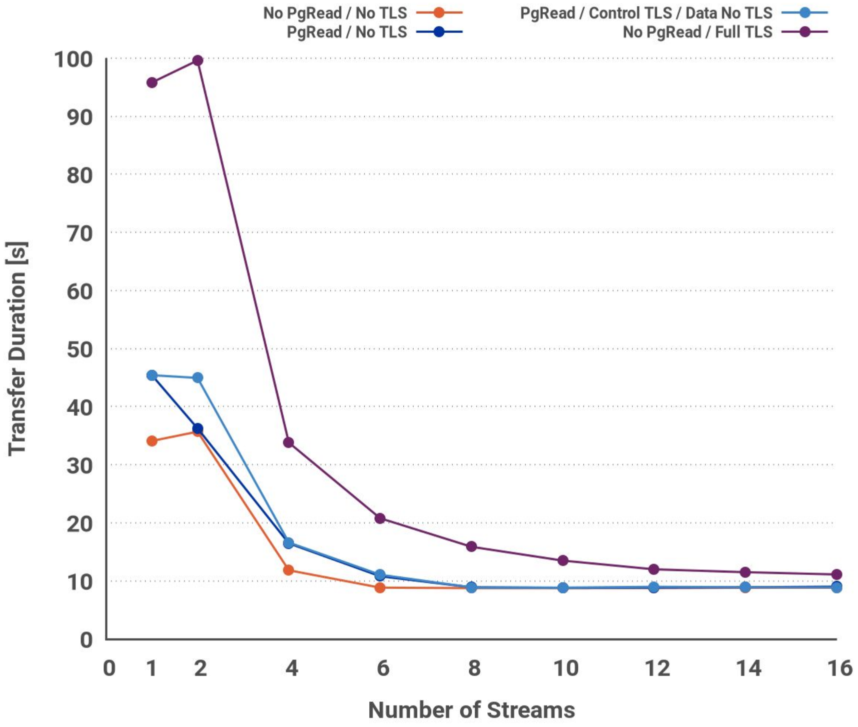
Solution is to Map Sockets to Different Pollers!



XRootD Client Performance with New Architecture

Mapping sockets from same channel to different pollers improves performance.





Summary and Conclusion

- Benchmarked various clients on 100GbE network
  - Comparable performance for curl, davix, and xrdcp in single-stream copies
- Identified reason for performance bottleneck in XRootD client with multiple streams
  - Plan to include the fix for this into a future release of XRootD
  - Significant impact for XCache, since it relies on the client to access original data
- Networking with 100GbE NIC behavior is different than with 1GbE NIC
  - Single CPU core not enough to process high request rates (even after tuning)
  - Need to resort to concurrent transfers for now, or multiple streams once the fix is released
- XRootD PgRead/PgWrite is not free, but good compromise in terms of performance
  - Can still easily reach 100Gbps speeds with 8 streams or more
  - For speeds beyond 200Gbps, may need to use network io_uring + pgread/pgwrite
- TLS has bigger performance impact, much higher CPU cost than PgRead/PgWrite

CERN

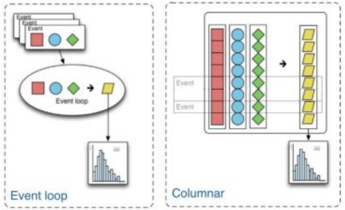
XRootD/XCache in the context of Analysis Facilities

I/O performance studies of analysis workloads on production and dedicated resources at CERN

A. Sciabà, J. Blomer, P. Canal, D. Duellmann, E. Guiraud, A. Naumann, V.E. Padulano, B. Panzer-Steindel, A.J. Peters, M. Schulz, D. Smith

Analysis Grand Challenge ttbar analysis

- **Simplified analysis from CMS used as technical demonstrator in IRIS-HEP**
  - Input dataset 3.6 TB, 2300 ROOT files, 1.5 GB/file consisting of CMS 2015 Open Data
- **Columnar analysis paradigm**
  - Distributed using a map-reduce concept
- **Original Coffea implementation**
  - ROOT-less, parallelism via Python futures or Dask
- **RDataFrame port (talk)**
  - ROOT-based, parallelism via implicit multithreading, Dask and other



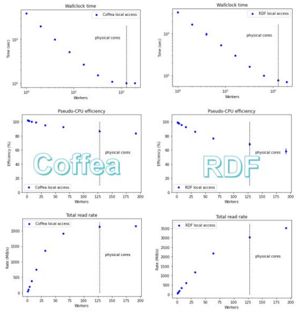
- **Measure performance and scalability**
  - Local parallelism on client node
  - Data read from local node vs. directly from EOS via xrootd vs. via an XCache instance
  - NOT a comparison between Coffea and RDF
    - Simply, different workloads with different behaviors

6

XRootD/XCache in the context of Analysis Facilities

Local access

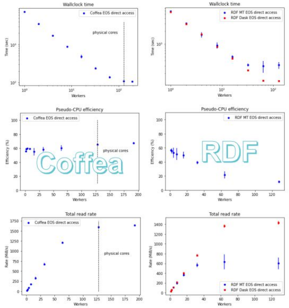
- **Scalability is excellent**
  - Some bottleneck appears for high numbers of workers
  - Overcommitting does not help for Coffea, but it increases throughput for RDF
    - Indication that Coffea is more CPU-constrained
- **The CPU efficiency comparably high**
  - I/O not a strong bottleneck
- **Local, fast SSD storage is always going to work well**
  - Aggregate read rates up to 3 GB/s



7

Direct access to EOS

- **Scalability still good when parallelism is via multiprocess**
  - RDF implicit multithreading does not perform well with xrootd and many threads
  - RDF via Dask is multiprocess, scales better
- **CPU efficiency practically constant around 60% with multiprocess parallelism**
  - I/O time is not negligible anymore but no bottlenecks
- **Two EOS instances tested**
  - EOSCMS by default, but CERNBOX produces similar (slightly worse) results



8

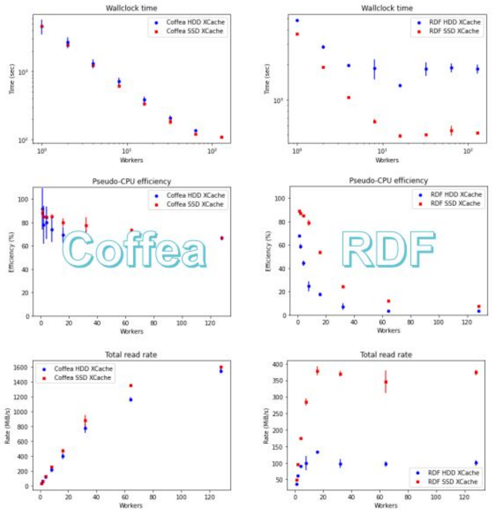
HDD/SDD-based XCache

- **Compared performance of direct access to Nebraska and CERN, cold cache and warm cache**

Coffea + HDD XCache: wallclock time (s)			
Site	Direct	Cold	Warm
Nebraska	442 ± 16	608	133 ± 6
EOSCMS	139 ± 8	325	137 ± 3

RDF MT + HDD XCache: wallclock time (s)			
Site	Direct	Cold	Warm
Nebraska	5470 ± 910	18841	1531 ± 78
EOSCMS	323 ± 95	8149	1558 ± 400

- **Performance results**
  - A cold cache is slower than direct access!
    - Due to sparse file access and network latency
  - Multiprocess scales very well
    - HDD XCache almost as good as SSD XCache
  - RDF multithreaded scales very poorly "out of the box"
    - All connections multiplexed into one ⇒ bottleneck!
    - SDD XCache helps a lot, but scalability is still broken



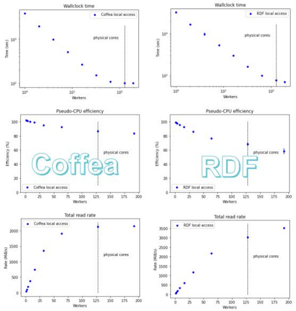
9

XCache == single server == single channel ⇒ XRootD client is CPU bound!

Better XCache Performance by Forcing more Channels

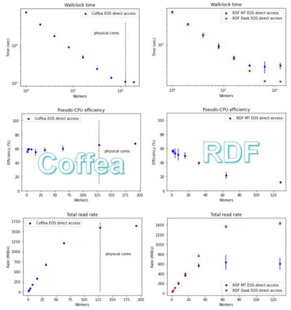
Local access

- **Scalability is excellent**
  - Some bottleneck appears for high numbers of workers
  - Overcommitting does not help for Coffea, but it increases throughput for RDF
    - Indication that Coffea is more CPU-constrained
- **The CPU efficiency comparably high**
  - I/O not a strong bottleneck
- **Local, fast SSD storage is always going to work well**
  - Aggregate read rates up to 3 GB/s



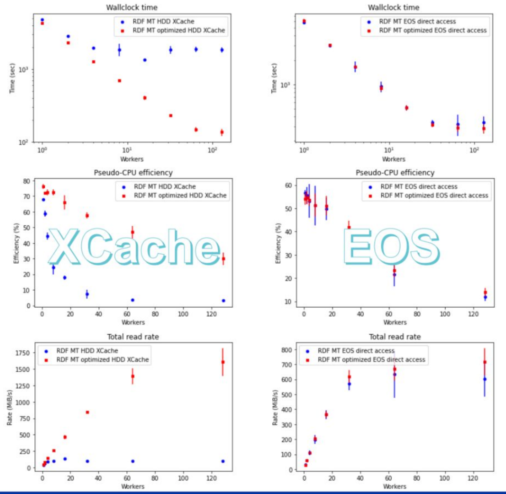
Direct access to EOS

- **Scalability still good when parallelism is via multiprocess**
  - RDF implicit multithreading does not perform well with xrootd and many threads
  - RDF via Dask is multiprocess, scales better
- **CPU efficiency practically constant around 60% with multiprocess parallelism**
  - I/O time is not negligible anymore but no bottlenecks
- **Two EOS instances tested**
  - EOSCMS by default, but CERNBOX produces similar (slightly worse) results

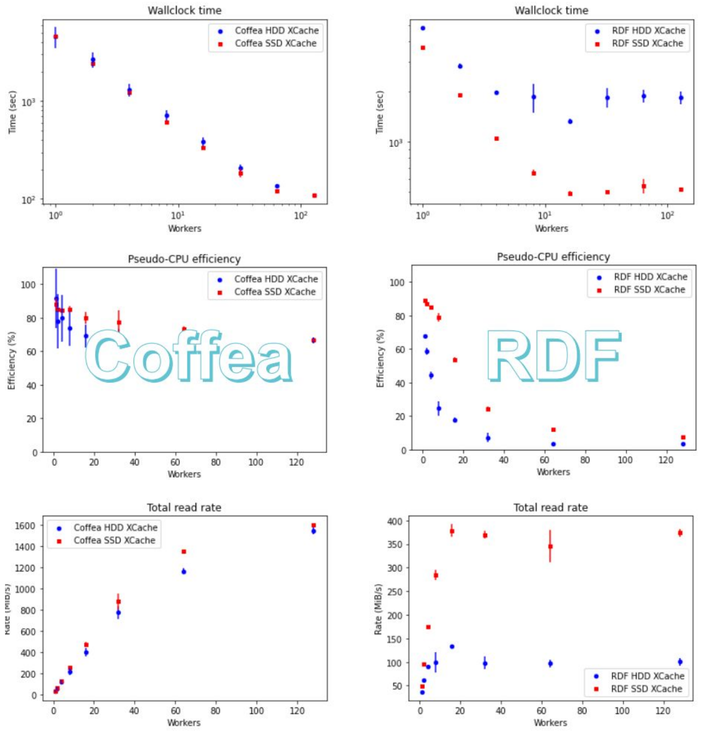


RDF + Xrootd performance optimization

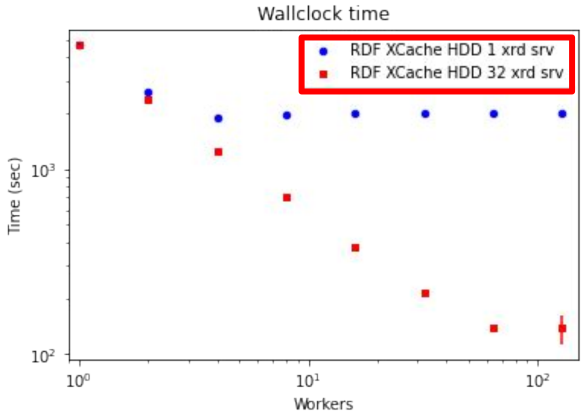
- **Scalability with ROOT multithreading and XCache can be improved**
  - XRD_PARALLELEVTLOOP=10 on the client largely improves Xrootd performance
  - Prevent the connection multiplexing by adding different client names to the file names `root://client1@eoscms.cern.ch//eos/myfile.root`
- **Enormous impact when reading from XCache**
  - Obvious as it is a single server and multiplexing a big bottleneck
- **Effect negligible when reading from EOS**
  - Files already spread over hundreds of disk servers, multiplexing irrelevant



Running Multiple XCache Daemons is Other “Solution”



After reconfiguring XCache server with internal clustering, ROOT performs much better for multi-threaded analysis.



No “fake clients” trick needed here.