Benchmarking and Optimisation of High-Performance Data Access and Storage Systems

CERN openlab Summer Student Programme

Author: Gustavo Mattos Lopes

Supervisor: Guilherme Amadio

August 2026

Project Specification

Objectives

- Benchmark storage solutions for CERN's computing infrastructure
- Evaluate alternatives to AFS for lxplus home directories
- Develop a reproducible methodology to evaluate future storage technologies, particularly in preparation for the HL-LHC era

Project Specification

Deliverables

FIO Benchmark Suite

JULY

Reproducible Flexible I/O (FIO) job files designed to replicate representative HEP analysis and user I/O workloads.

Storage Insights

AUGUST

Evaluation of benchmark results to characterise and compare storage systems under the representative workloads.

Methodology

Workloads definition

User Activities

Workload Tasks

- Git Clone
- Compilation
- Pip Install
- ROOT Tutorials

HEP Applications

HEPScore23 Benchmarks

ATLAS: atlas-gen-sherpa-ma-bmk / atlas-reco-mt-ma-bmk

CMS: cms-gen-sim-run3-ma-bmk / cms-reco-run3-ma-bmk

LHCb: lhcb-sim-run3-ma-bmk

Belle II: belle2-gen-sim-reco-ma-bmk

ALICE: alice-digi-reco-core-run3-ma-bmk

Methodology

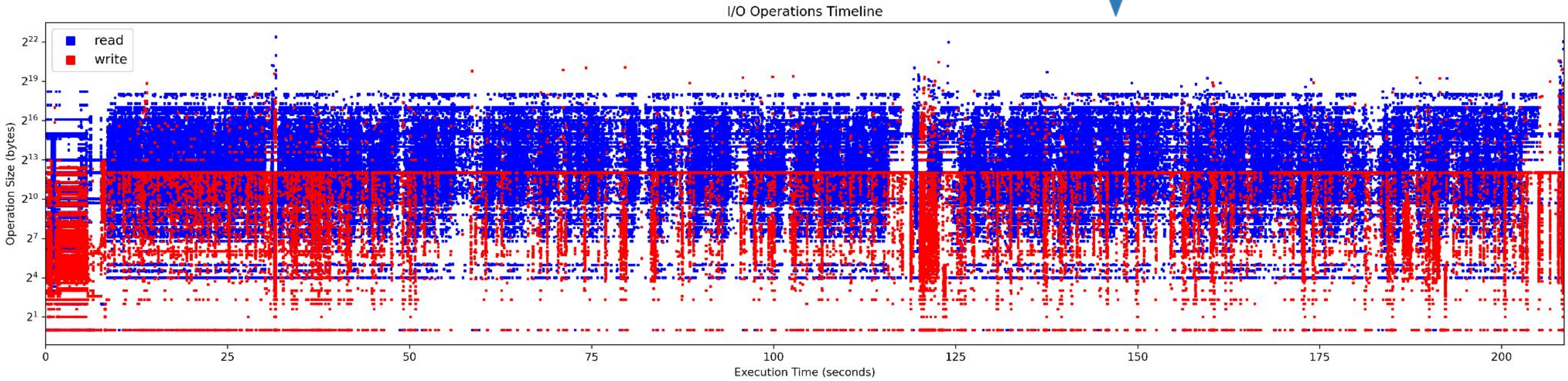
Trace capture: XRootD compilation

Execution Command

```
strace -f -ttt -T -o trace.log -e
trace=read,write ctest -S test.cmake
```

Trace Log

```
…
100208 1783674073.132947 write(6, "\n\t.align 8\n\t.type\t_ZTV8XrdOssDF,"..., 4096) = 4096
<0.000145>
100236 1783674073.134293 read(7, "//------------------------------"..., 1693) = 1693 <0.000260>
100236 1783674073.135277 read(7, "//------------------------------"..., 1514) = 1514 <0.000685>
100208 1783674073.136104 write(6, "de/stdio.h\"\n\t.file 61 \"/usr/incl"..., 4096) = 4096
<0.000157>
…
```

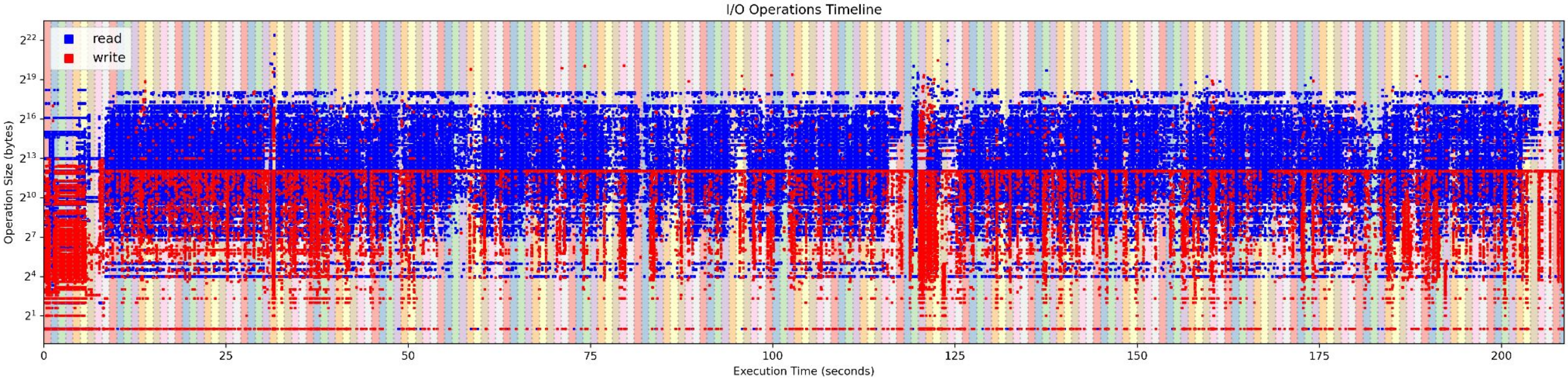


Methodology

Trace time segmentation

Split the trace into fixed-size time windows (typically one per second)

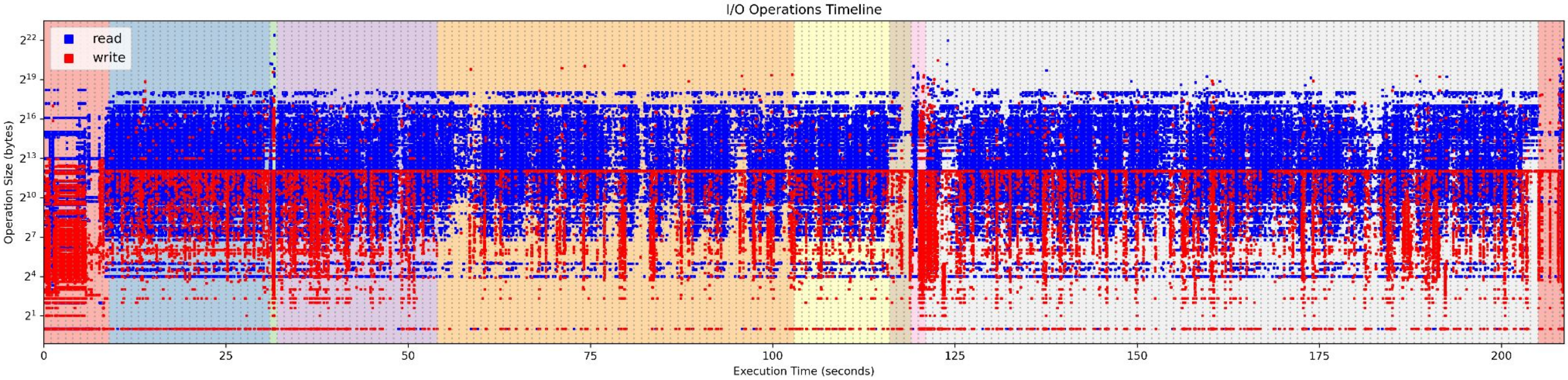
to capture and characterise workload variations over time.



Methodology

Trace workload clustering

Group trace windows with similar I/O behaviour to identify a smaller set of representative patterns reproducible through FIO jobs.



Methodology

FIO Workload Generation

Each cluster is translated into an FIO job reproducing its I/O behaviour.

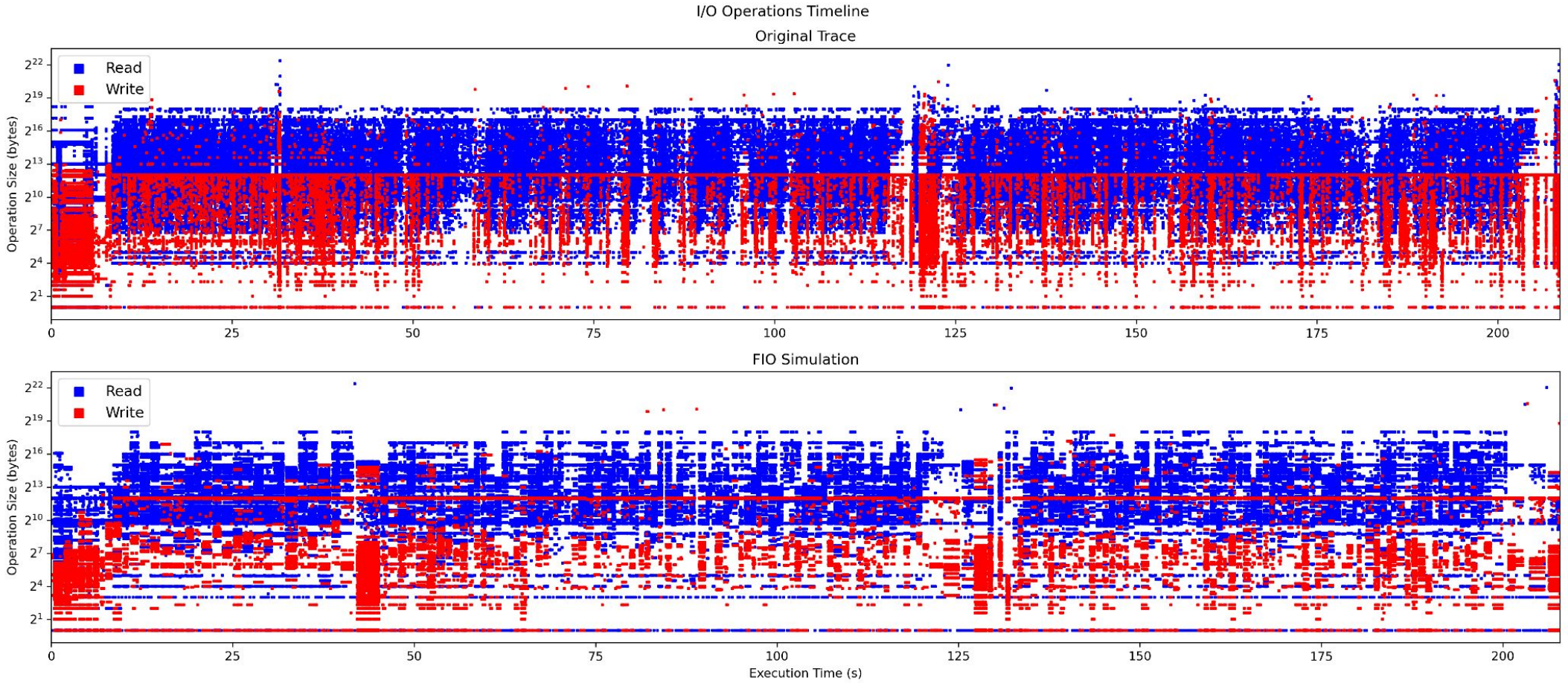
script.fio

```
[global]
ioengine=sync
directory=data
nrfiles=9622
filesize=550k
filename_format=file.$filenum
stonewall

[cluster0]
rw=randrw
rwmixread=83
number_ios=41944
bssplit=4096/35:832/22:1024/11:8192/9:8191/9:4/5:1/2:16/2:64/2:192
0/1:442/1:704/1:294/1:13735/1:242/1:164/1:4032/1:1088/1:96/1:512/1:
75/1:114/1:1472/1:1063/1:126/1:4286/1:32768/1:289/1:768/1:108/1:246
2/1:1109/1:128/1:970/1:28672/1:255/1:69632/1:31/1:232/1:353/1:384/1
:2176/1:18527/1:1409/1:1283/1:1810/1:273/1:185/1:143/1:481/1:1661/1
:3737/1:7332/1:5713/1:1905/1:1340/1:1266/1:206/1:867/1:5039/1:1037/
1:159/1:3156/1,4096/11:726/2:832/2:128/4:2029/1:48/80
...
```

Methodology

FIO Workload Comparison



Methodology

Block-size Distribution Optimisation

To satisfy FIO's constraints, we pre-select a restricted set of request sizes based on frequency and data volume, then apply an optimisation model to generate new integer percentages that minimise deviation from the expected mean.

1. Heuristic Pre-selection

Addressing FIO bssplit Limits

The simulation of tracked request size distributions is constrained by FIO's limit of **64 entries**.

To reduce the search space for large clusters, a candidate set is pre-selected using two criteria:

- **32 (or more) Most Frequent Sizes:** Selected based on high occurrence rate.
- **32 (or more) Largest Data Volumes:** Selected based on total bytes transferred.

2. Optimisation Formulation

Minimises expected mean deviation

$$\min e$$

$$\text{s.t.} \sum_{i=1}^{n} p_i = 100,$$

$$100\mu - \sum_{i=1}^{n} p_i s_i \leq e,$$

$$\sum_{i=1}^{n} p_i s_i - 100\mu \leq e,$$

$$p_i \in \mathbb{Z}_{\geq 0}, \quad i = 1, \ldots, n,$$

$$e \in \mathbb{R}_{\geq 0}$$

3. FIO Workload

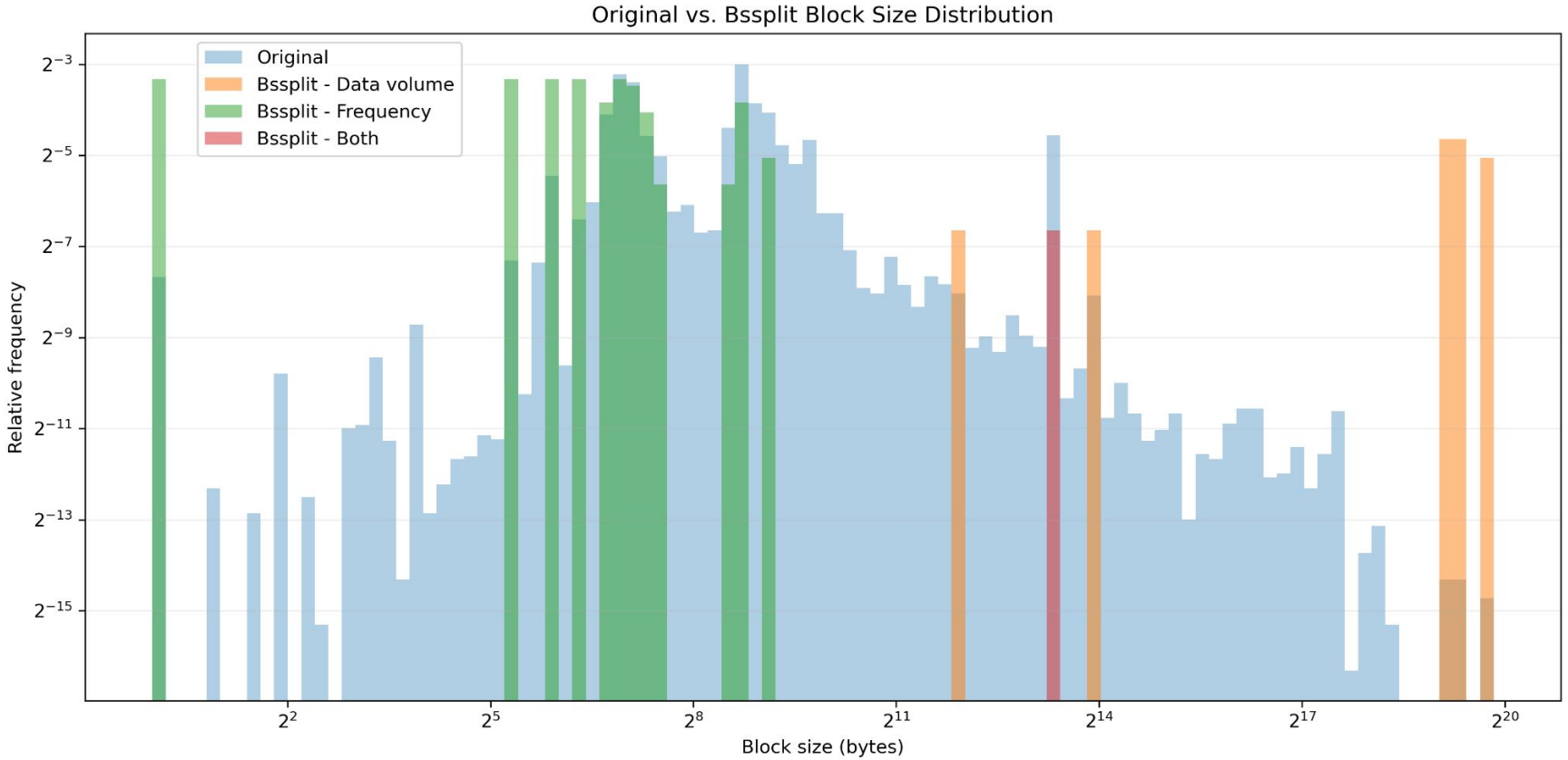
*Results evaluated using the **atlas-reco-mt-ma-bmk** workload.*

Metric	Original Trace	FIO Workload
Total operations	1,190,515	1,181,909
Read operations	880,576	879,223
Write operations	309,939	302,686
Total bytes	12,777,335,624	12,647,969,773
Read bytes	11,433,496,594	11,346,493,189
Write bytes	1,343,839,030	1,301,476,584
Average read size (bytes)	12,984.1	12,905.1
Average write size (bytes)	4,335.8	4,299.8

Methodology

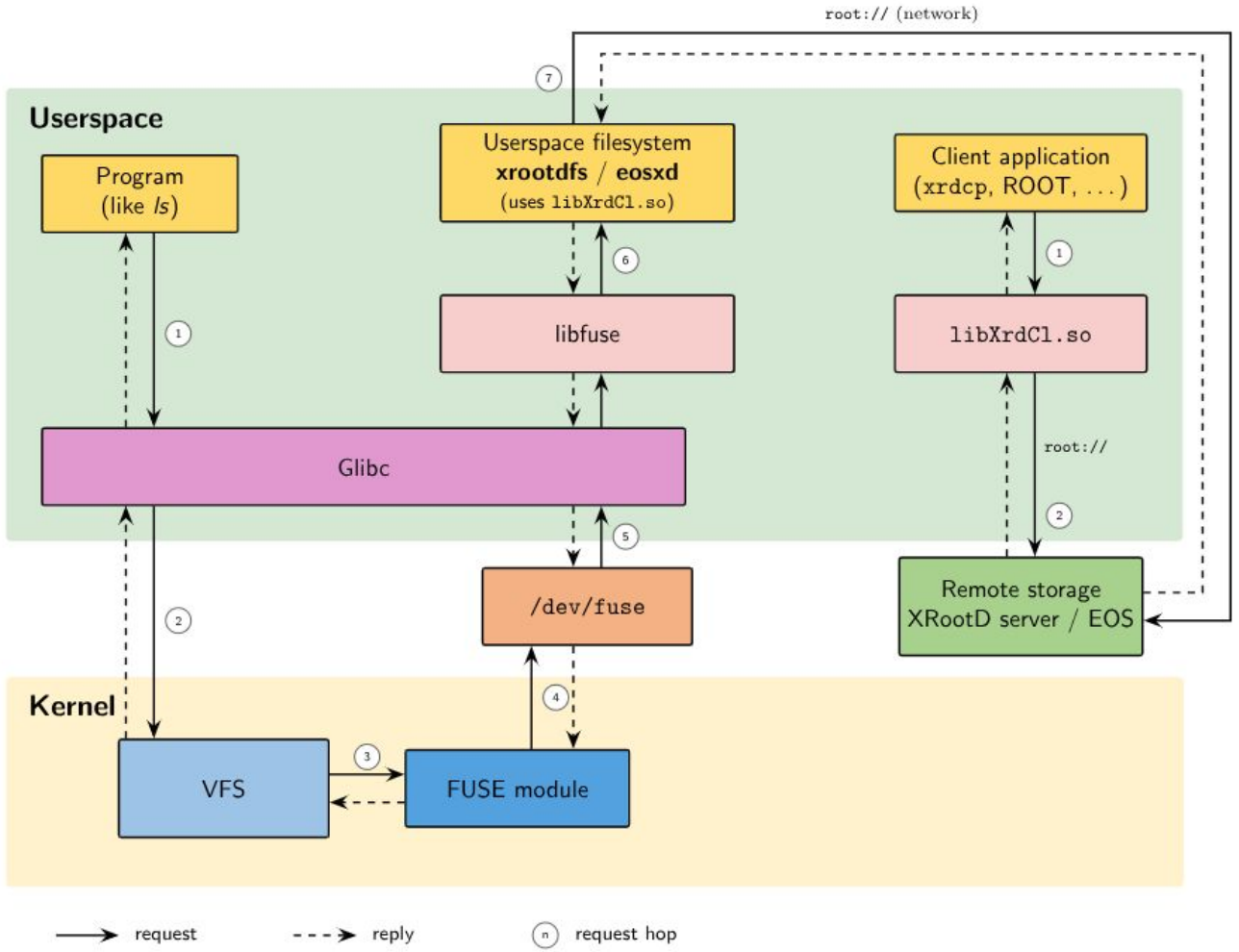
Block-size Distribution Optimisation

Example of the resulting simplified block-size distribution for an **atlas-reco-mt-ma-bmk** workload cluster



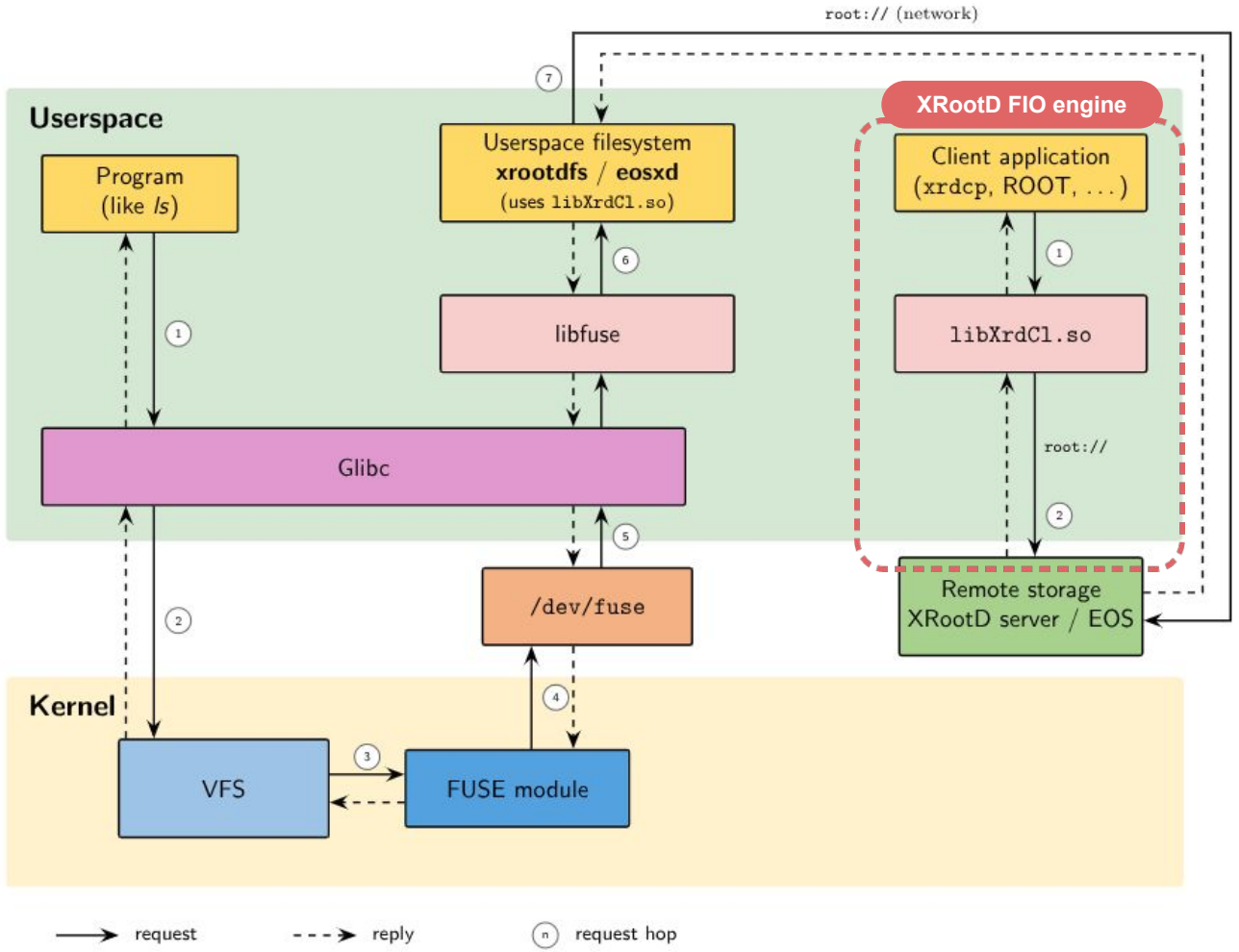
I/O Request Optimisation

FUSE mount vs. direct XRootD access via FIO engine



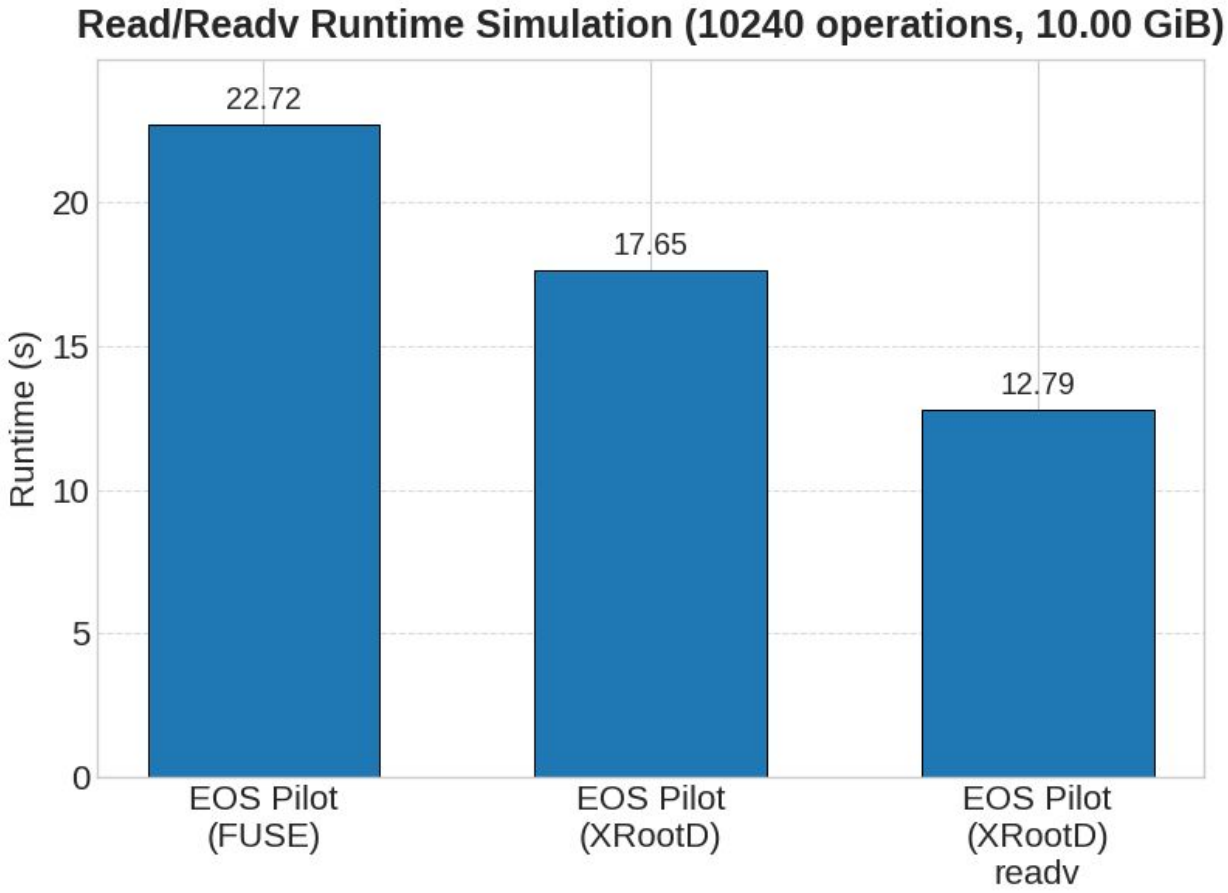
I/O Request Optimisation

FUSE mount vs. direct XRootD access via FIO engine



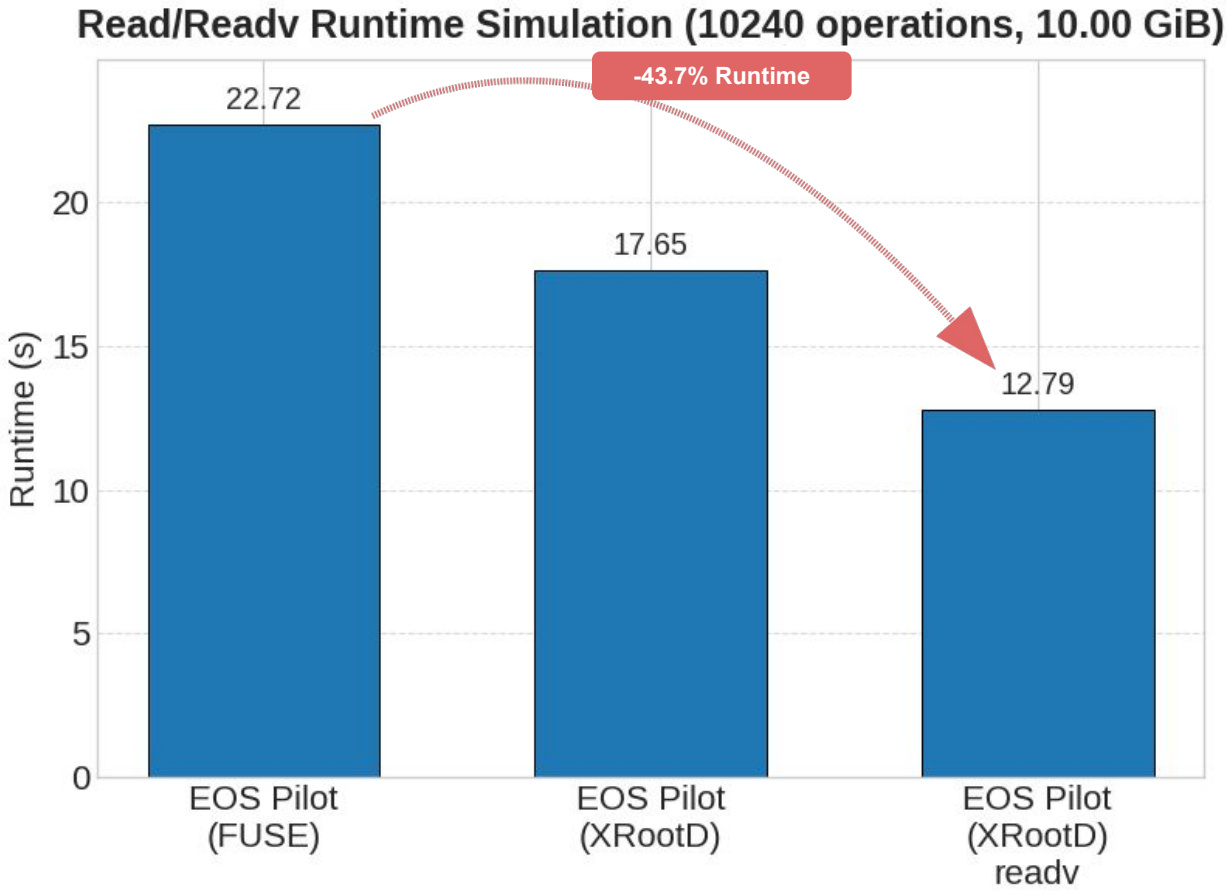
I/O Request Optimisation

FUSE mount vs. direct XRootD access via FIO engine



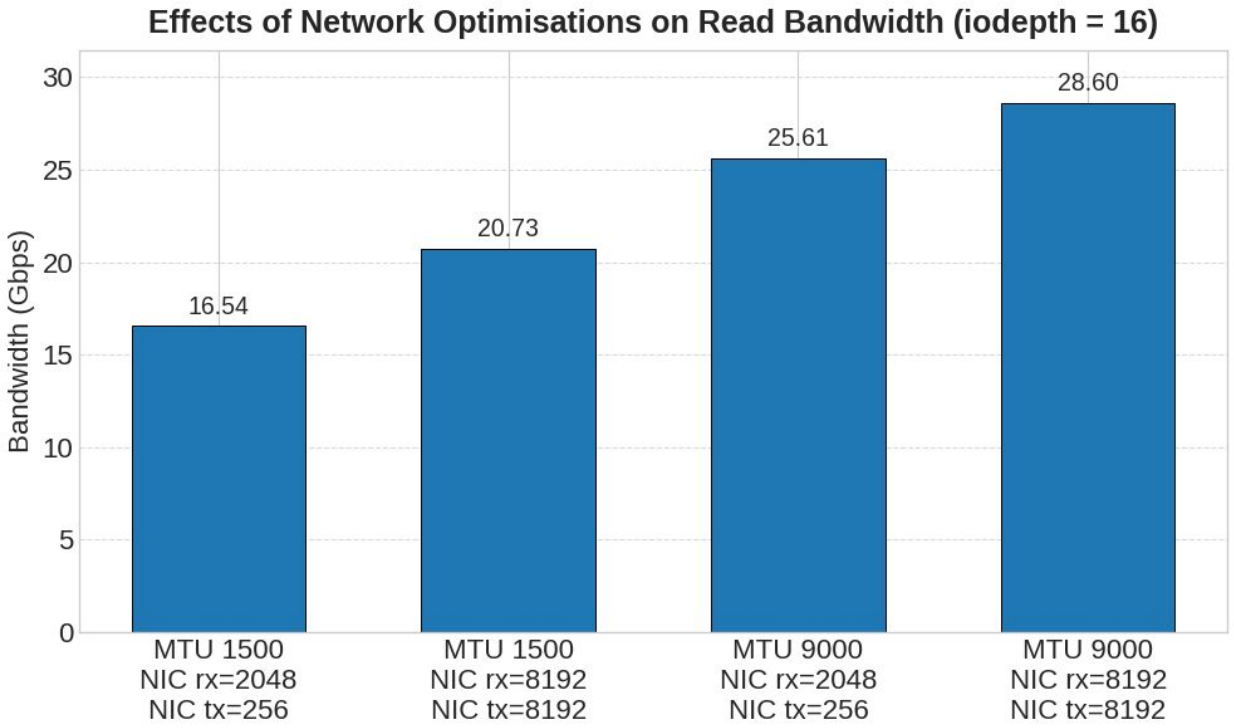
I/O Request Optimisation

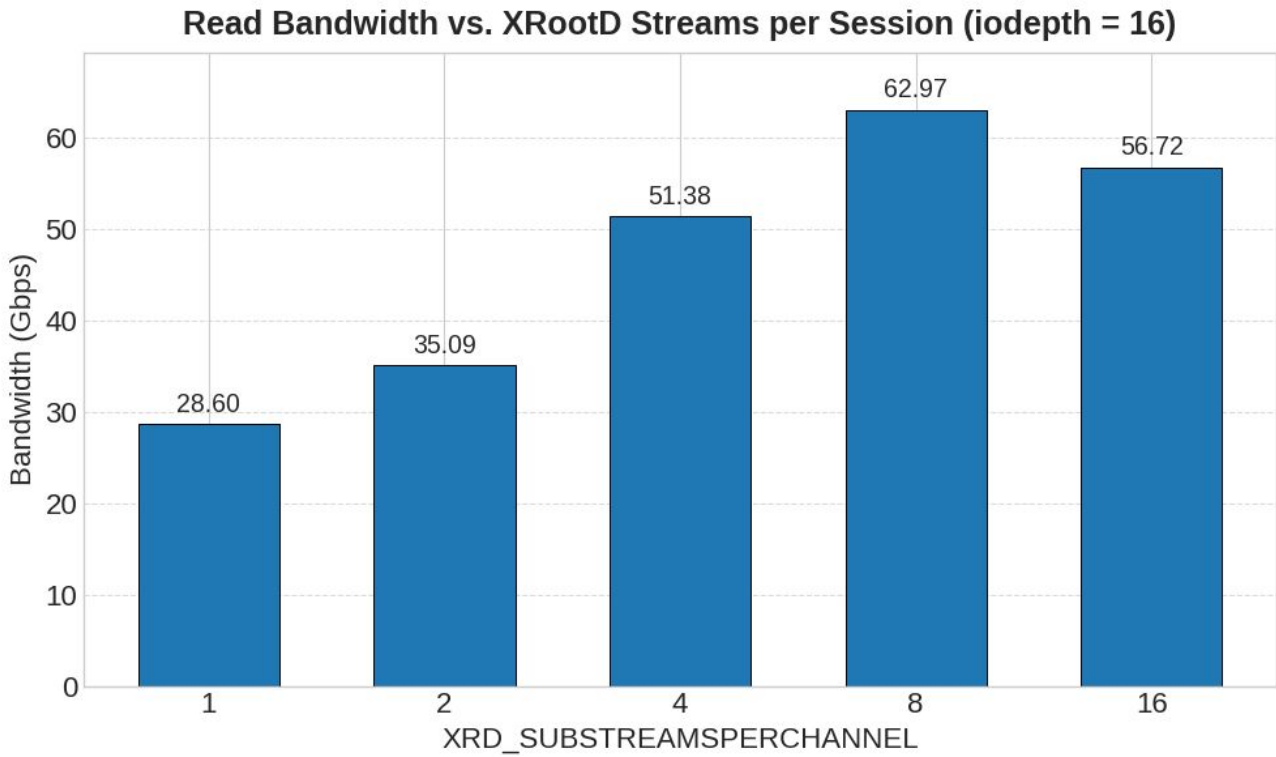
FUSE mount vs. direct XRootD access via FIO engine



Network Optimisation

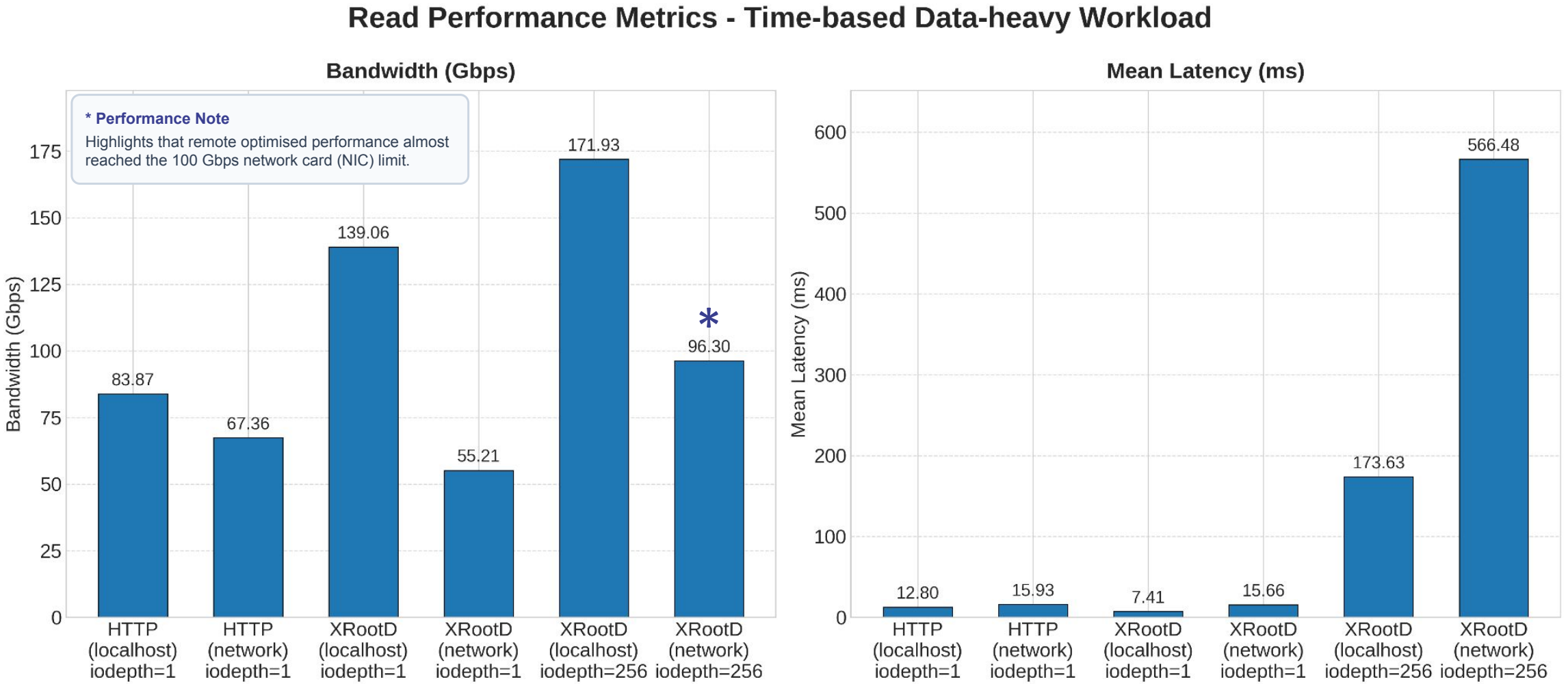
Maximum Transfer Unit (MTU), NIC buffer sizes and XRootD streams





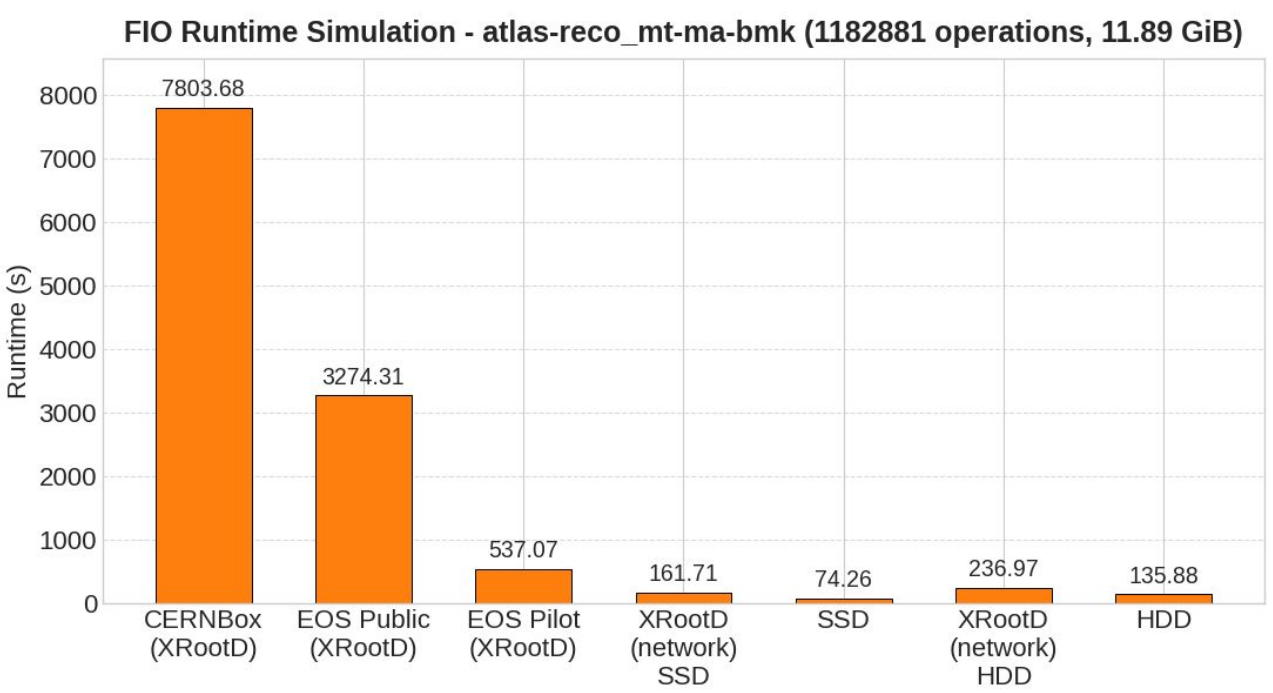
Network Benchmarking

Local vs. Remote (optimised) storage access on XRootD server

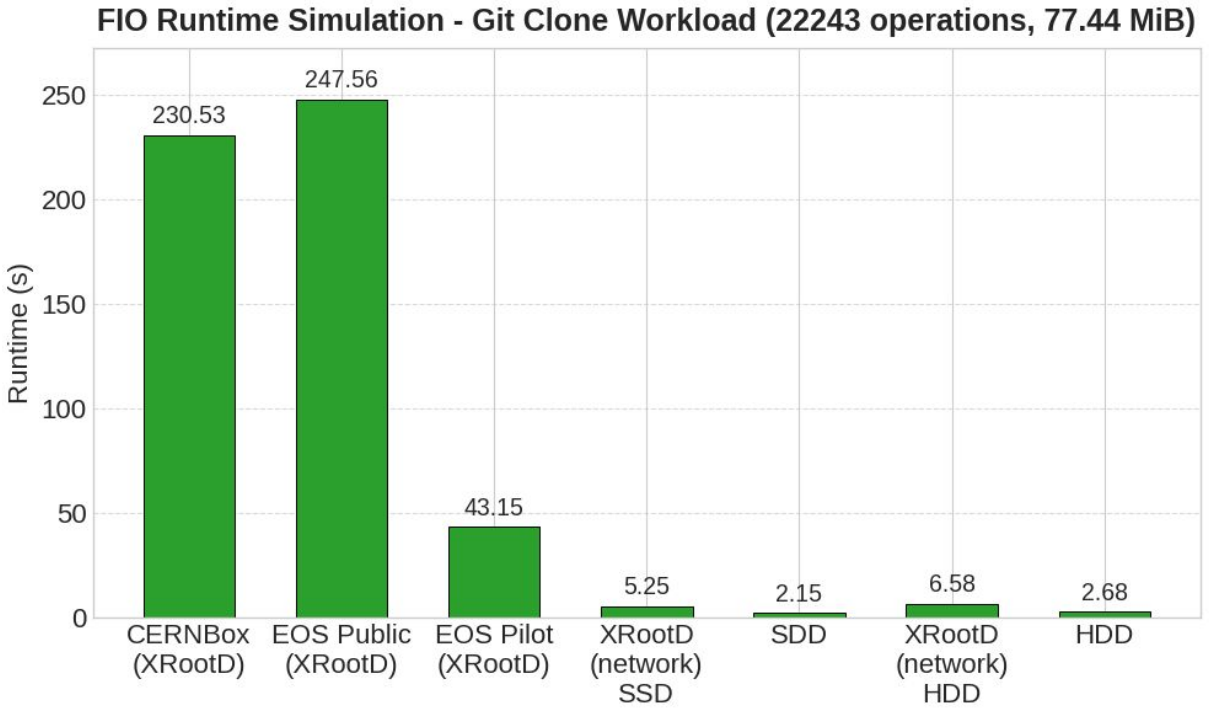


Storage Benchmarking

Comparison between EOS instances, XRootD servers and baselines



Data-heavy workload



Metadata-heavy workload

Storage Benchmarking

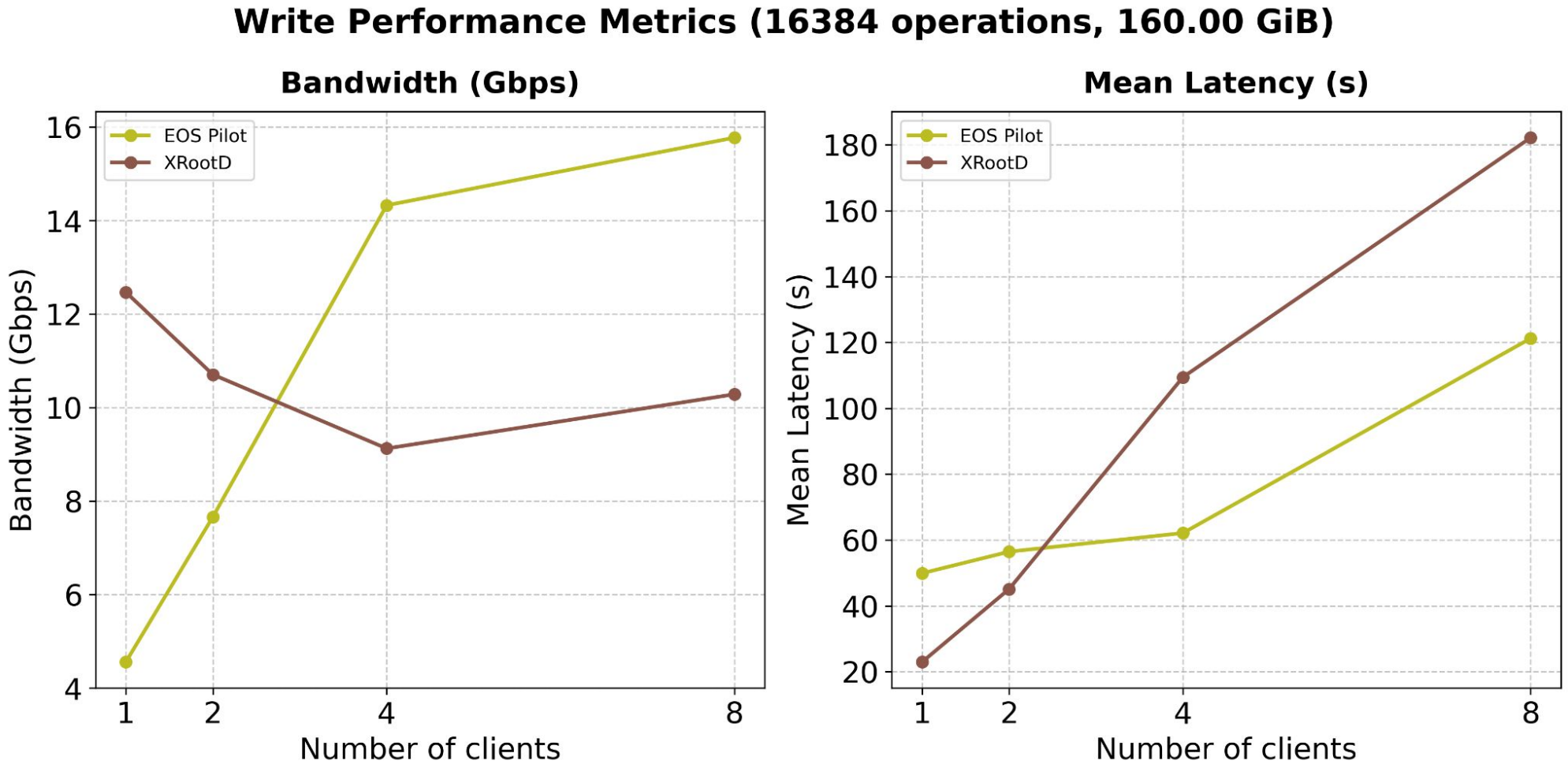
Why were CERNBox and EOS Public worse than EOS Pilot?

- **User-Intensive Constraints**
- **Unoptimised Network**
- **Greater External Influence**

```
[root@eoshome-i01-vega (mgm:master) ~]$ eos access ls | grep -A100 'Stall Rules'
# Stall Rules ...
# ......................................................................................
[ 01 ]                rate:user:*:Chmod => 500    "from_puppet"
[ 02 ]                rate:user:*:Chown => 500    "from_puppet"
[ 03 ] rate:user:*:Eosxd::ext::LS-Entry => 10000    "from_puppet"
[ 04 ]     rate:user:*:Eosxd::prot::SET => 50    "from_puppet"
[ 05 ]            rate:user:*:GetFusex => 200    "from_puppet"
[ 06 ]           rate:user:*:OpSetFile => 300    "from_puppet"
[ 07 ]               rate:user:*:Open => 500    "from_puppet"
[ 08 ]        rate:user:*:OpenDir-Entry => 20000    "from_puppet"
[ 09 ]     rate:user:*:OpenFailedCreate => 50    "from_puppet"
[ 10 ]     rate:user:*:OpenFailedENOENT => 100    "from_puppet"
[ 11 ]     rate:user:*:OpenFailedExists => 100    "from_puppet"
[ 12 ]   rate:user:*:OpenFailedNoUpdate => 50    "from_puppet"
[ 13 ] rate:user:*:OpenFailedPermission => 50    "from_puppet"
[ 14 ]      rate:user:*:OpenFailedQuota => 10    "from_puppet"
[ 15 ]            rate:user:*:OpenProc => 1000    "from_puppet"
[ 16 ]            rate:user:*:OpenRead => 500    "from_puppet"
[ 17 ]           rate:user:*:OpenWrite => 300    "from_puppet"
[ 18 ]                 rate:user:*:Rm => 100    "from_puppet"
[ 19 ]               rate:user:*:Stat => 2000    "from_puppet"
[ 20 ]     rate:user:ams:Eosxd::ext::LS => 1
[ 21 ] rate:user:ams:Eosxd::ext::LS-Entry => 10
[ 22 ] rate:user:dbaronmo:Eosxd::ext::LS-Entry => 100
[ 23 ]          rate:user:gazis:AttrLs => 50
[ 24 ]         rate:user:gcorreia:Stat => 50
[ 25 ]         rate:user:jefeselk:Stat => 50
[ 26 ]           rate:user:weiy:AttrLs => 50
[ 27 ]                    threads:* => 100
[ 28 ]                threads:127699 => 10
[ 29 ]                threads:134918 => 10
[ 30 ]                threads:138858 => 10
[ 31 ]                threads:140860 => 10
[ 32 ]                threads:147772 => 1
[ 33 ]                   threads:max => 4000
```

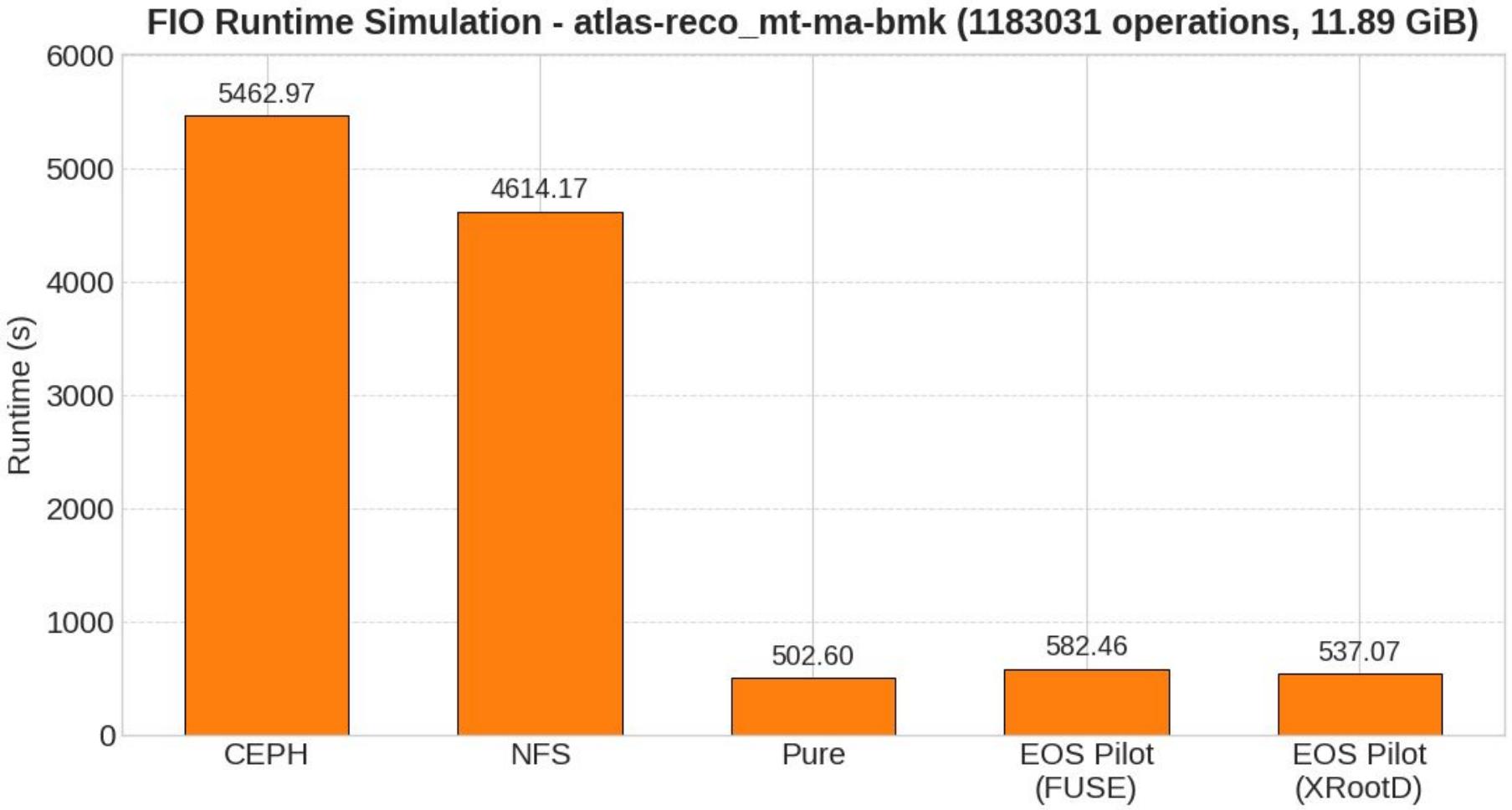
Storage Benchmarking

EOS scalability with increased client concurrency



Storage Benchmarking

Distributed storage systems: Data-heavy workload



Storage Benchmarking

Why were CEPH and NFS the worst?

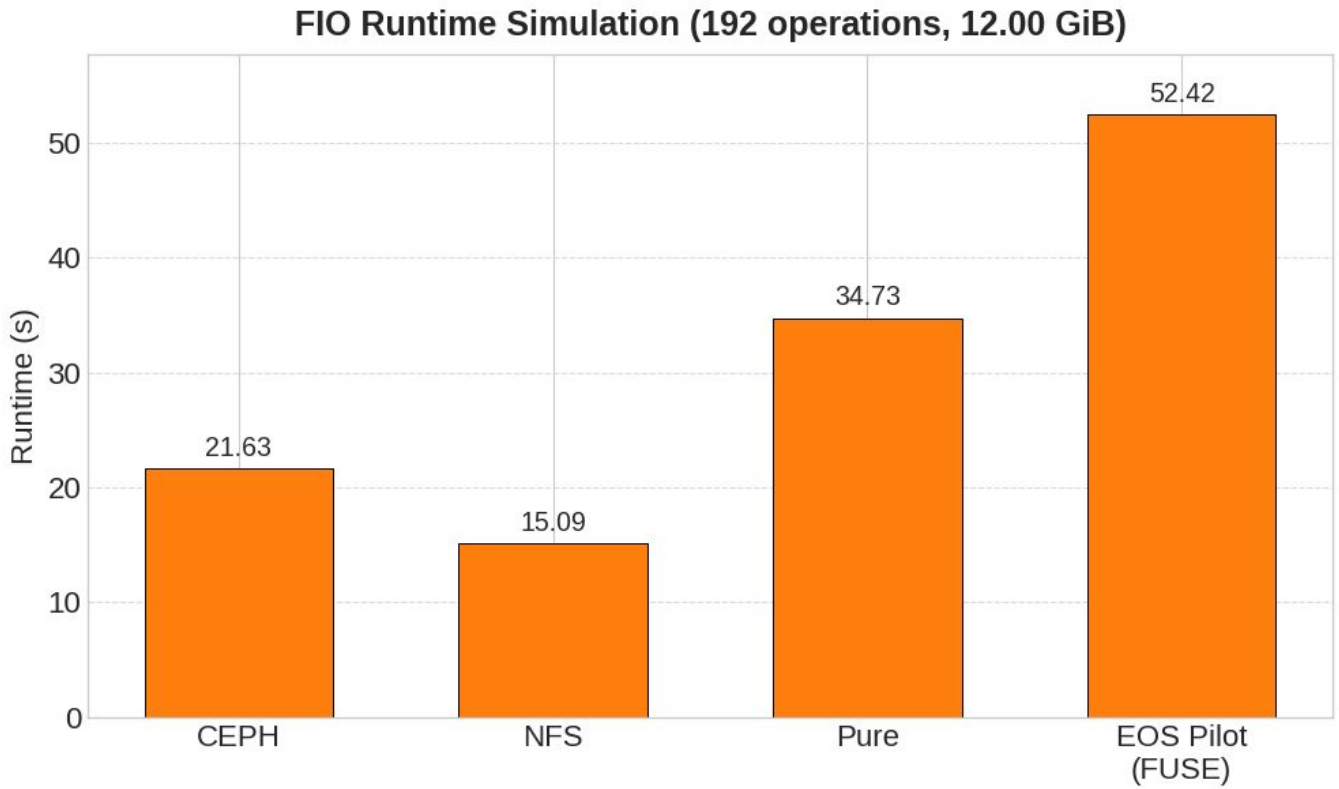
Performance Analysis

Small I/O overhead

Ceph and NFS incurs higher overhead for small I/O requests.

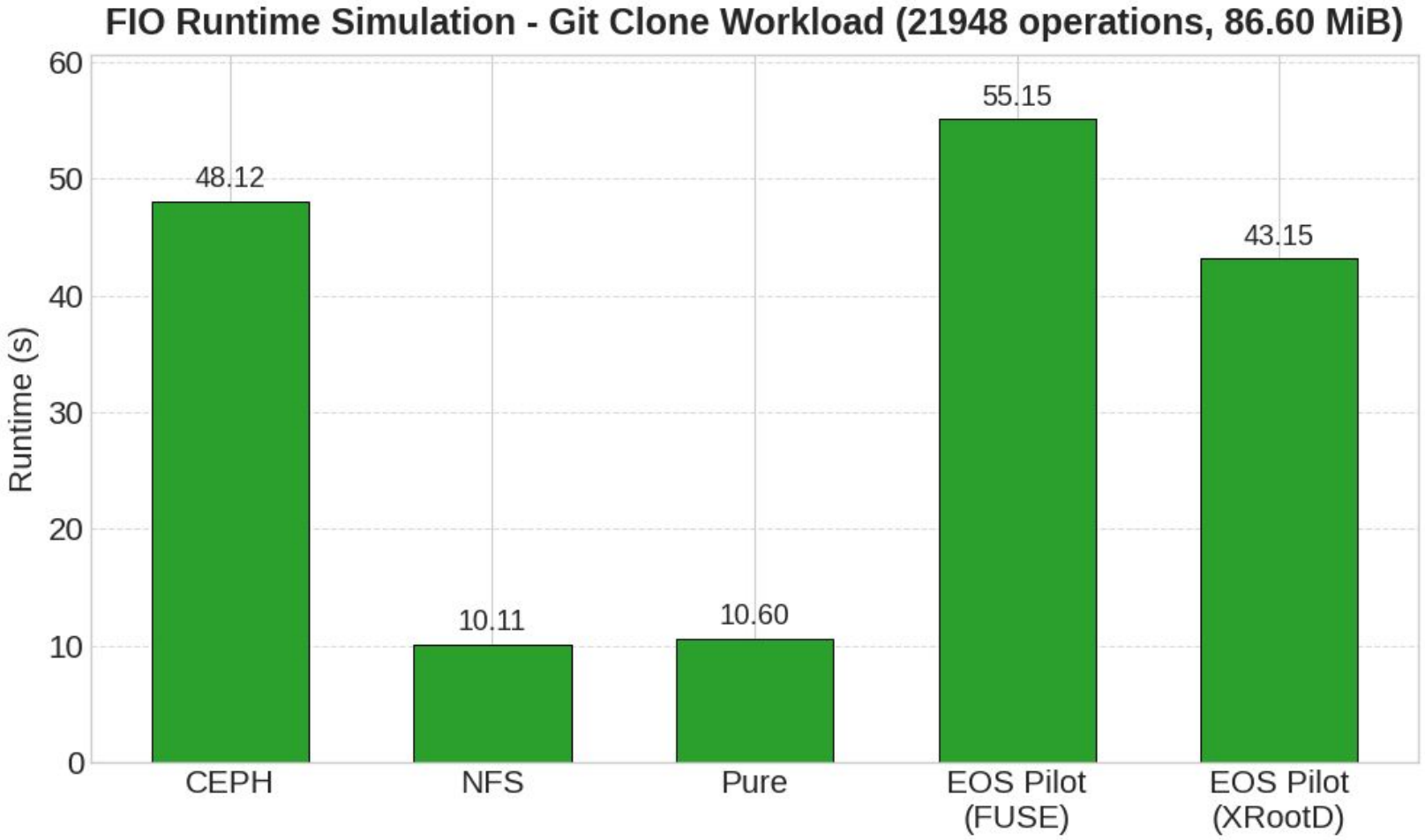
Uniform block size solution

Using 64 MiB blocks significantly improves their performance.



Storage Benchmarking

Distributed storage systems: Metadata-heavy workload



Thank you!

Gustavo Mattos Lopes

August 2026