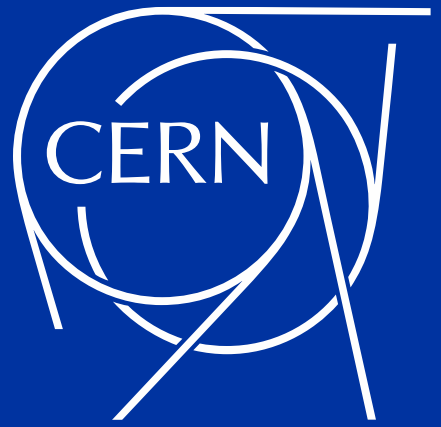


EOS and XCache data access performance for LHC analysis at CERN

Dirk Duellmann, Bernd Panzer-Steindel, Markus Schulz, Andrea Sciabà, David Smith (CERN IT-SC)

HEPiX Spring online Workshop, 25-29 April 2022

Introduction

LHC analysis at CERN

- Hundreds of physicists using a variety of methods, tools and resources
  - Batch and interactive
  - Machine Learning and columnar data formats increasingly popular

Support in IT for analysis

- Are the hardware and the services adequate to current (and future) needs?
- Does anything special need to be done about analysis (e.g., having a dedicated analysis facility)?

Data access performance for analysis

- A working group was created to give an answer to these questions
- Experts from the experiments on software and computing and from IT
- The scope was only CERN and Run 3: no attempt at providing answers for WLCG or other sites or for longer time scales!

Structure of the talk

- **Overview of analysis resources at CERN**
- **EOS usage for analysis**
- **Data access performance studies with XCache**

- **Final report**
  - https://zenodo.org/record/6337728
  - A document summarizing the report was prepared, will be published in the next days
- **Contact**
  - it-dep-cern-lhc-analysis-studies@cern.ch

Analysis resources overview

- **Three main classes of resources locally available at CERN**
  - Grid (CERN is also a Tier-2)
  - Batch (direct submission to LXBATCH)
  - Interactive (LXPLUS, Spark/HADOOP, SWAN, …)
- **Heterogeneous resources are increasing**
  - GPUs available since long, ARM will be soon
- **The first step is to understand how much they are used**

Measuring utilization for analysis

- **First question is: what is analysis?**
  - Easy to answer for Grid jobs (from experiments' job monitoring)
  - In other cases, it is less clear cut, can only calculate upper bounds
- **Grid and batch analysis are by far the largest contributions**
  - Peaks sometimes 3-4 times the average
- **Interactive analysis still at low scale**
  - Usually, cores allocated to users are mostly idle
    - Example: SWAN cores used for about **5%** of their time

Experiment	Completed jobs	Running jobs	CPU time	Wallclock time
ATLAS	100K	2500	70 years	80 years
CMS	600K	8000	140 years	170 years

CERN Grid analysis in a 7-day period in October

	LXBATCH	LXPLUS
ATLAS	4400 cores	100 cores
CMS	8800 cores	70 cores

Average number of busy cores in a 7-day period

	SWAN	Spark/YARN
ATLAS	8.5K hours · cores	18.5K hours · cores
CMS	16K hours · cores	22K hours · cores

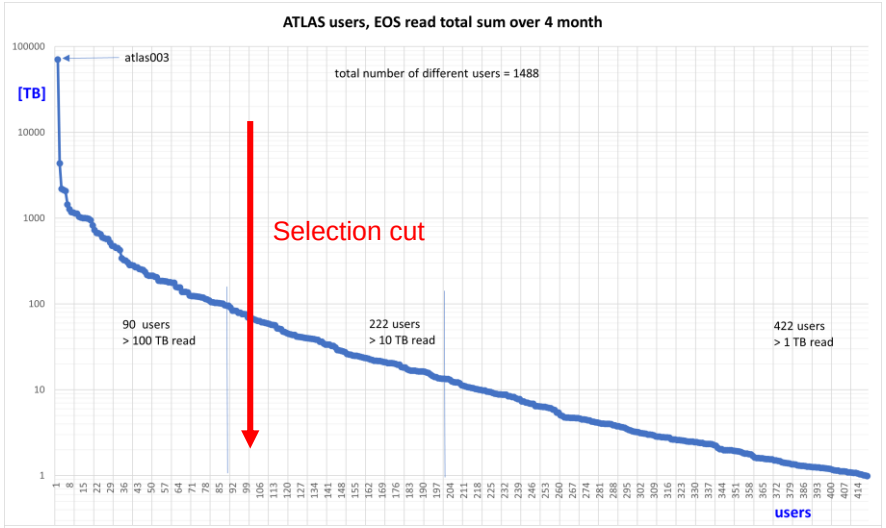
Wallclock time in a 10-day period

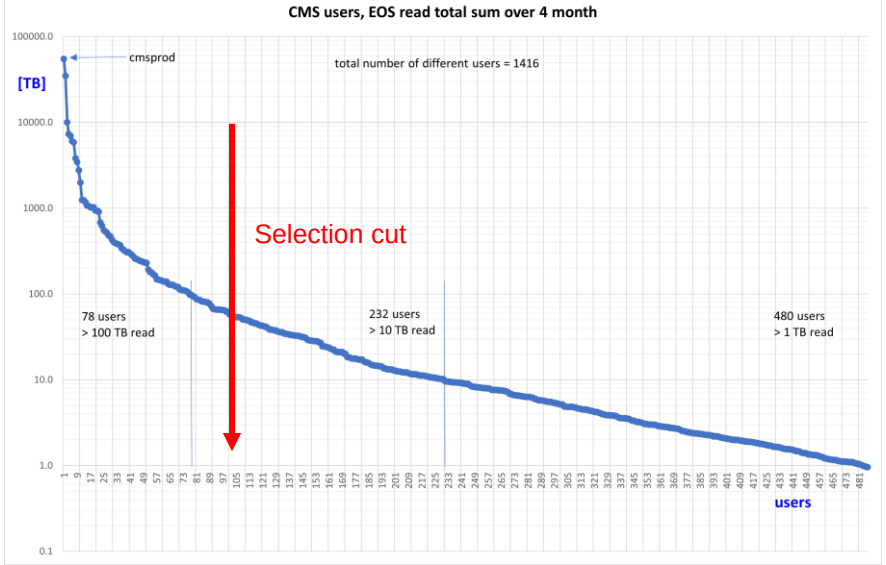
Data access performance and EOS

- **Goal: study data access patterns on EOS and determine if there are bottlenecks in the system**
  - Used EOS log files between February and May 2021
  - Internal EOS activities (rebalance, etc.) and production accounts filtered out

	ATLAS	CMS
Distinct users	1488	1416
Total data read	117 PB	169 PB
from LXPLUS	7%	2%
from batch	74%	61%
from eosftp	15%	33%
from external IPs	4%	4%

- **Selected only “regular” users who read more than 100 TB (“heavy analysis users”)**
  - Only O(100) users for ATLAS and CMS remaining





Read distributions

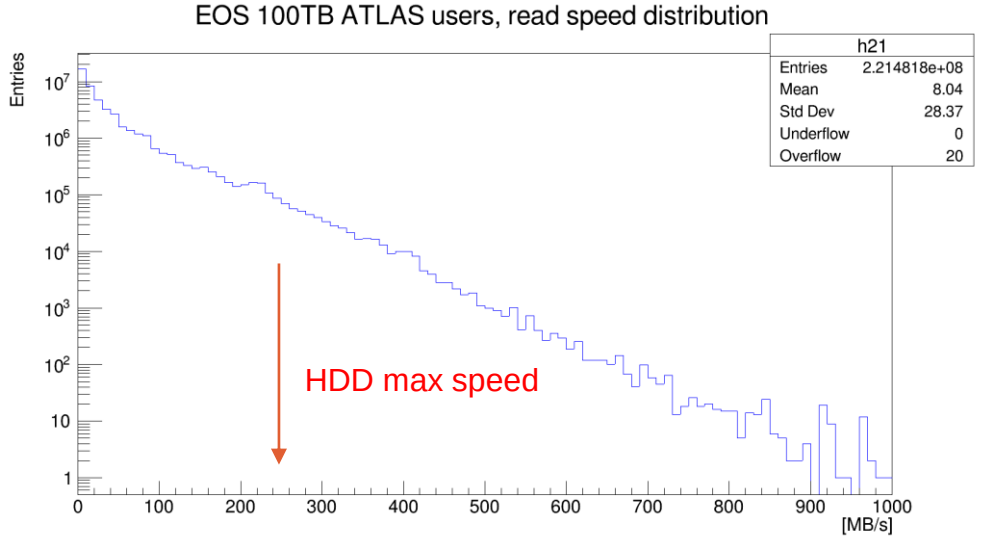
Average read rates are rather low and dominated by small reads

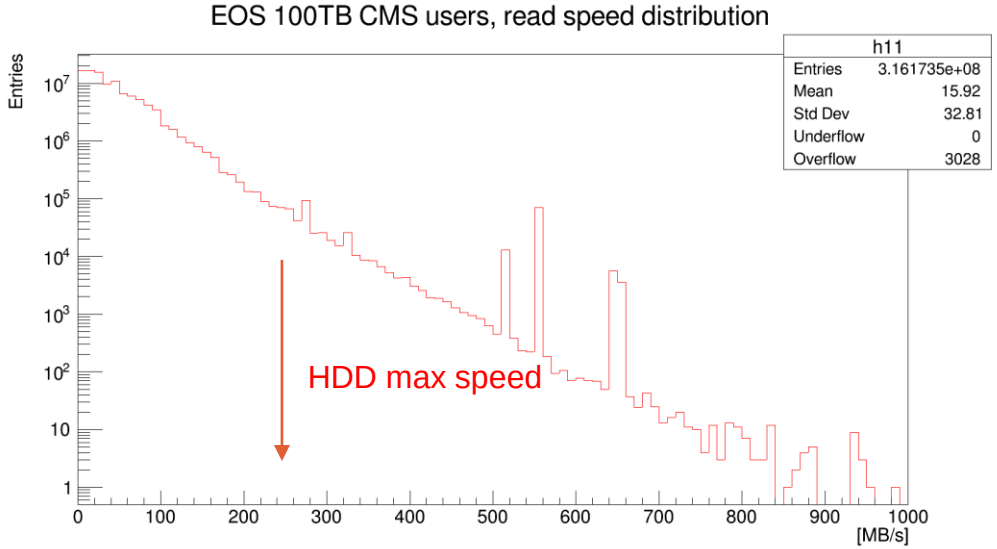
- ATLAS: 8 MB/s (52 MB/s excluding reads < 10 MB/s)
- CMS: 16 MB/s (24 MB/s excluding reads < 10 MB/s)

High transfer rates (> 250 MB/s) can be explained as accessed through memory caches

- Estimating 15-20 TB of memory cache installed for 30 PB of EOS disk space
- Less than 1% of accesses uses the cache

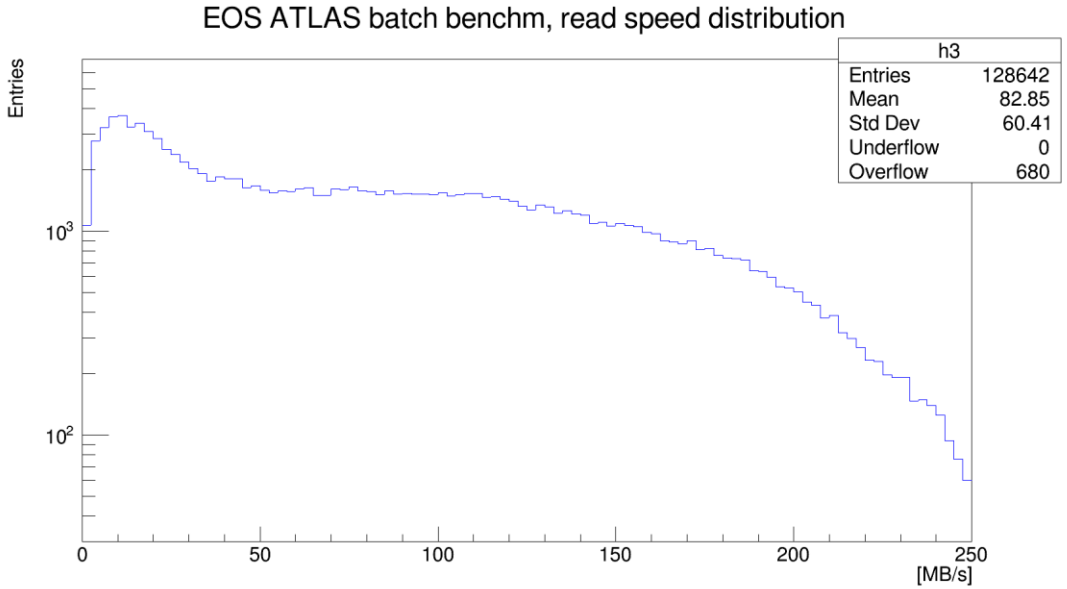
	ATLAS	CMS
Number of “heavy analysis users”	87	74
Total data read	40 PB	108 PB
Total size of unique files read	0.5 PB	11 PB
Number of unique files read	0.7M	5.3M
Number of files read (any number of times)	155M	38M

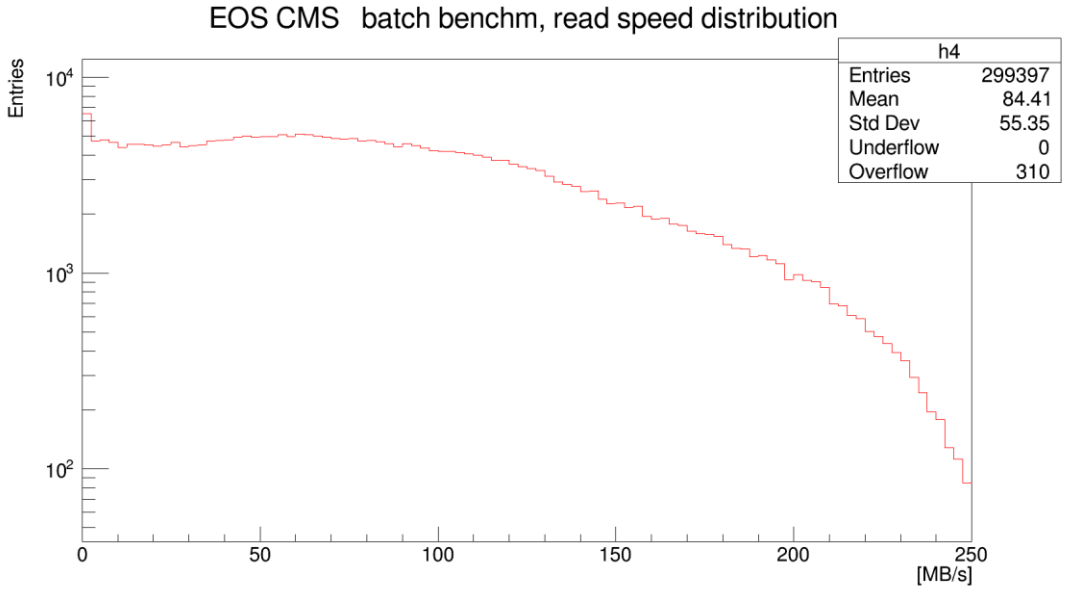




“Synthetic” benchmarks

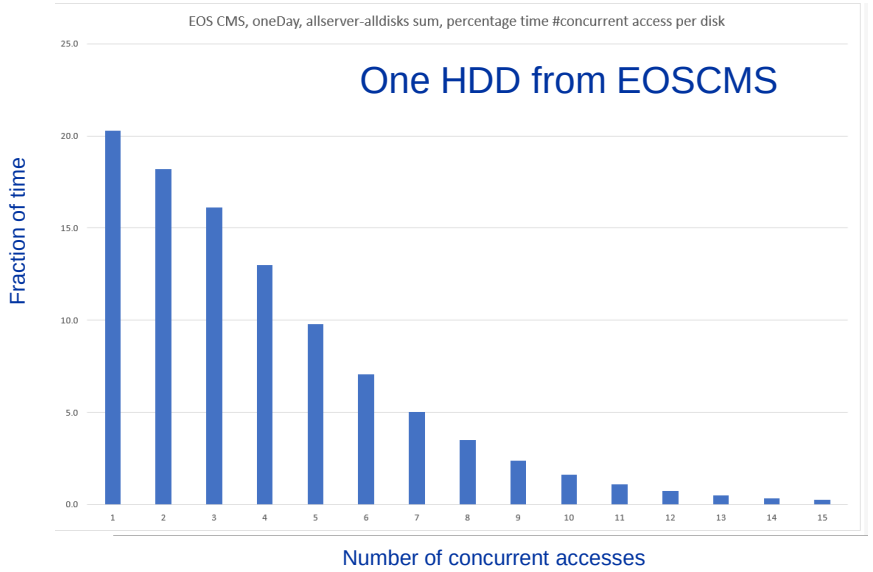
- **A simple check to measure “achievable” data read rates**
  - Thousands of trivial jobs randomly reading files accessed by analysis jobs, via EOS FUSE
  - Rates about 80 MB/s
  - Even higher rates (~100 MB/s) achieved using xrdcp
- **Other file statistics**
  - Open files per second:
    - ATLAS: 70% of entries, less than 20 file opens/sec
    - CMS: 85% of entries, less than 20 file opens/sec
    - Reaching into the thousands, no bottlenecks
  - Seeks per file:
    - ATLAS: 94% less than 10 seeks per file
      - Using local copies?
    - CMS: 60% less than 10 seeks per file

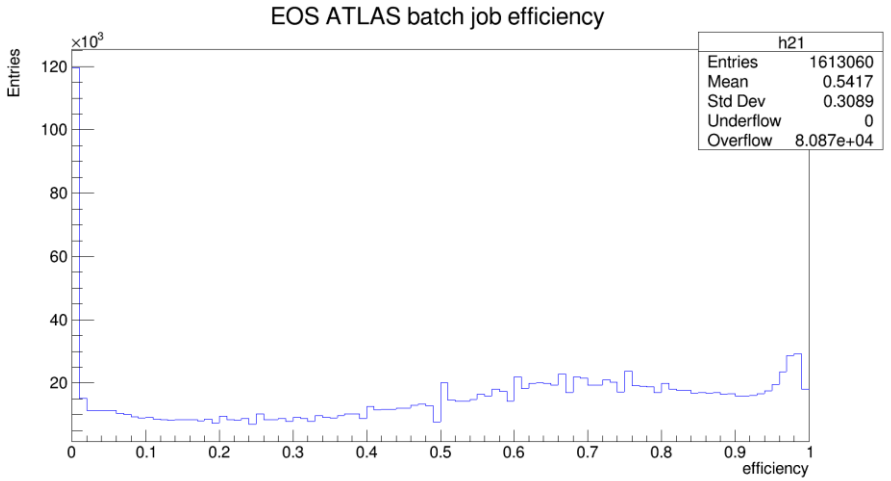


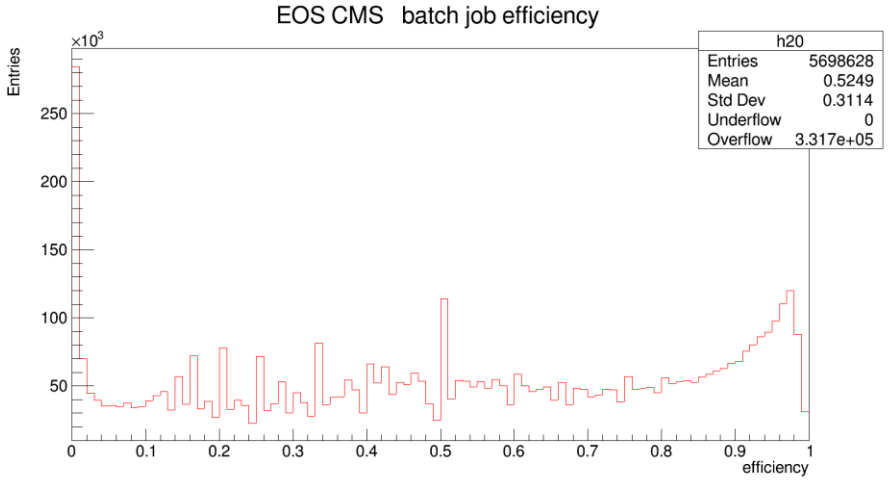


Job efficiencies and disk contention

- **CPU efficiencies (from batch log files) relatively low**
- **A quick look at disk contention shows that concurrent accesses to the same disks are the norm**
  - Which explains the spread in data read rates

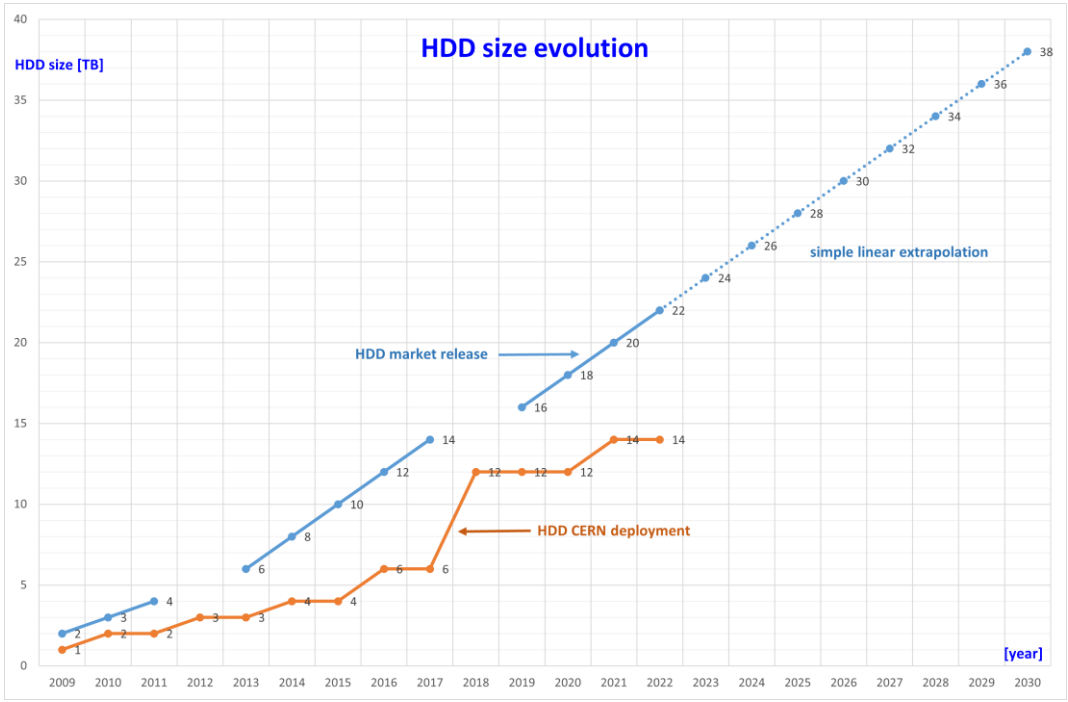






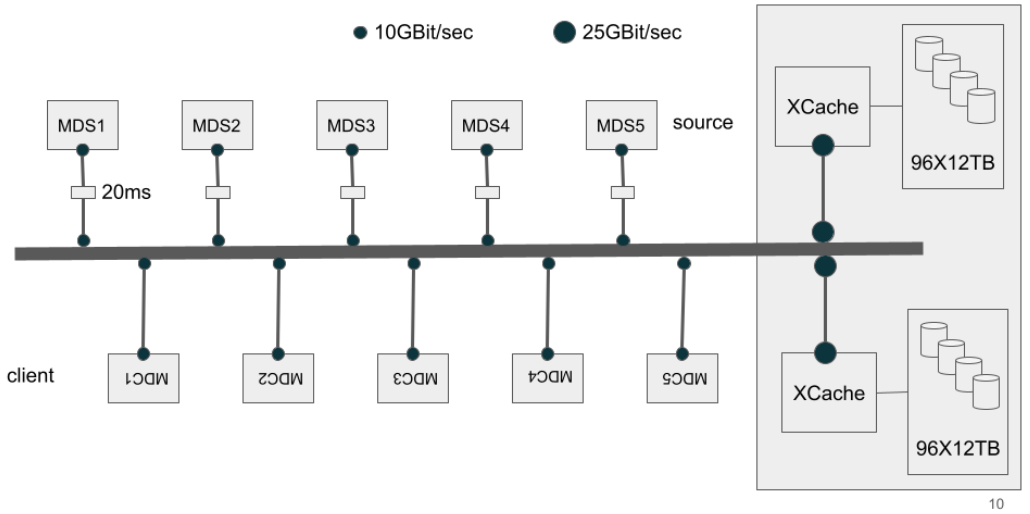
HL-LHC lookout and wrap-up

- **Extrapolations to 2028**
  - Resource request of 500 PB of HDD storage per ATLAS and CMS each
  - HDD size of 40 TB, 100 disks per server, server memory 512 GB, 500 MB/s per disk
  - Mix of erasure coding and two copies
  - → twice as many spindles as today, total throughput 1 TB/s
- **EOS measurements wrap-up**
  - Currently, **no performance bottlenecks observed** on EOS infrastructure for the bulk of analysis workloads
    - Still considerable headroom in I/O and network
  - **No apparent need for caching layers**



XCache data access performance

- **The goal is to measure performance and scalability of both SSD and HDD based XCache instances with respect to analysis workloads. For example:**
  - When do SSDs give an advantage?
  - How many clients can an XCache instance support?
- **Three types of tests**
  - Synthetic tests using *fio* and *dd*
  - “Baseline” tests simulating real access patterns (MockData)
  - Tests using “real” analysis applications from ATLAS and CMS



- **Testbed consisting of**
  - 2 nodes with 96 12TB HDD each
  - 2 nodes with 16 2TB SSD each
  - Both pairs used for Xcache and with 25 Gb/s connections, 64 logical cores each node
  - Several client nodes (mostly VMs)

Synthetic tests

- **Single HDD performance**
  - fio with 32 KB blocks for **random** access: max IOP ~ 220
  - dd with 10 MB blocks for **sequential** access: average rate ~ 200 MB/s
- **Single SSD performance**
  - IOP ~ 4500 using MockData *high seek job* (see next slide)
- **HDD-based XCache node performance**
  - Aggregate data rate of 6.67 GB/s reading locally with dd from all disks (theoretical maximum 7.88 GB/s)
- **SDD-based XCache node performance**
  - Aggregate data rate of 6.99 GB/s (theoretical maximum 7.88 GB/s)

“Baseline” tests using MockData

- **MockData is a custom tool to test data access performances by replaying a recorded history of data accesses by multiple jobs**
  - Presented at CHEP 2019
- **Sequential tests**
  - *randomreader*: ran 5 clients against the HDD-based instance and reached 5 GB/s (to be compared with the 6.25 GB/s of network bandwidth)
- **“High seek” access tests**
  - A *coordinator* process dispatches transfer job definitions to several *loadgenerator* processes to mimic real data access patterns
    - Used the actual file names, file sizes and open times from a 6-month period of ATLAS jobs at Prague in 2018. The files are “fake” and generated on the fly
    - File accesses via XrdCl
    - Can tune the transfer jobs to aim at a given target transfer rate

Storage type	Access pattern	Rate (overruns) GB/s	Rate (net monitoring) GB/s
HDD	Sequential	5.5	6.0
HDD	High seek	1.2	1.7
SSD	Sequential		~6
SSD	High seek		~6

- **Achievable rates on SSD- and HDD-based XCaches measured in two ways**
  - Rate when the transfer jobs start to be cancelled for exceeding their targeted time
  - Maximum rate measured at the network interface level
- **As expected, SSDs cope well even with random access**

Tests with “realistic” analysis workloads

- **Example workloads received from ATLAS and CMS**
- **The XCache nodes are also used as clients due to memory requirements**
  - SSD nodes clients for the HDD XCache and vice versa
  - Also local access tested
  - Tests use all available cores
- **Aggregate gradients**
  - ROOT script allocating 90 GB of RAM, processing 189 root files (1.5 TB in total), writes results
  - Multithreaded (one copy with 64 threads)
  - Used XRD_PARALLELEVTLOOP=16 to improve performance (env)
  - Forced the client to establish multiple TCP connections to the servers instead of multiplexing (to improve performance) (multi conn)

Name	Experiment	Data	Events
top-xaod	ATLAS	1 DAOD_PHYS file	35000
cms-skimmer	CMS	640 NANOAODSIM files	670M
Aggregate gradients	CMS	189 root files	

- **cms-skimmer**
  - Uses CMSSW
  - Builds the executable, processes a single file
  - Run in 64 copies, each reading one file at random
- **top-xaod**
  - Reads an input file and writes an output file
  - The only input file (1.16 GB) was copied 256 times
  - Run in 64 copies, each processing a random number of events from a random input file

Results from aggregate gradients

Configuration	Devices	Rate (GB/s)	CPU eff (%)	active devs (%)	rreq/s/dev	MB/s/dev	kB/rreq	util (%)
Local filesystem	1 HDD	0.15	2.38	100	346	181	523	99.9
Local filesystem	96 HDD	4.2	45.0	53.6	77.7	40.6	524	20.5
Local filesystem	16 SSD	5.5	61.5	92.5	567	297	524	55.3
XC, env only	96 HDD	0.21	5.95	32.3	4.23	2.21	523	1.06
XC, multi conn	96 HDD	2.5	28.8	56.7	47.6	24.9	524	14.0
XC, env only	32 SSD	0.90	10.5	87.5	53.0	27.7	522	5.34
XC, multi conn	32 SSD	2.8	32.6	89.5	187	85.5	446	17.0
XC, multi conn, no env	32 SSD	1.9	20.8	79.7	121	55.1	455	11.0

- Very I/O intensive workload
- Local access data rates 60-70% of maximum from synthetic tests
- The impact of the env and multi conn settings is huge: XCache rates approach the network limit
- SSD-based XCache only marginally better in rate than HDD-based

Results for cms-skimmer

Configuration	Rate (MB/s)	CPU eff (%)	active devs (%)	rreq/s/dev	MB/s/dev	kB/rreq	util (%)
EOS	92.8	76.0	-	-	-	-	-
XCache HDD, 2 nodes	93.0	81.4	35.2	6.04	0.938	155	1.41
XCache SSD, 2 nodes	95.5	81.5	64.1	60.7	2.43	40.0	0.932
Local filesystem, 16 SSD	-	92.4	100	91.8	12.3	135	3.72

- Relatively low I/O: low rates, low device utilization and good CPU efficiency
- No advantage with respect to reading directly from EOS

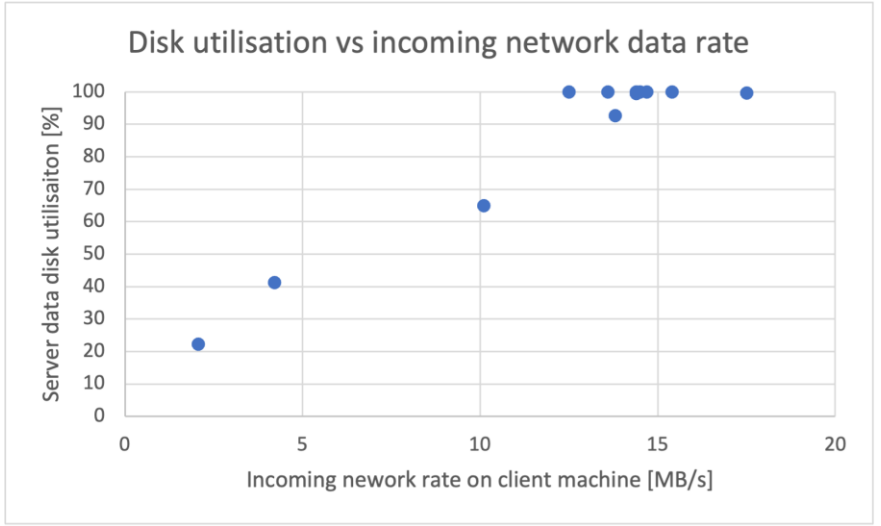
Results for top-xaod

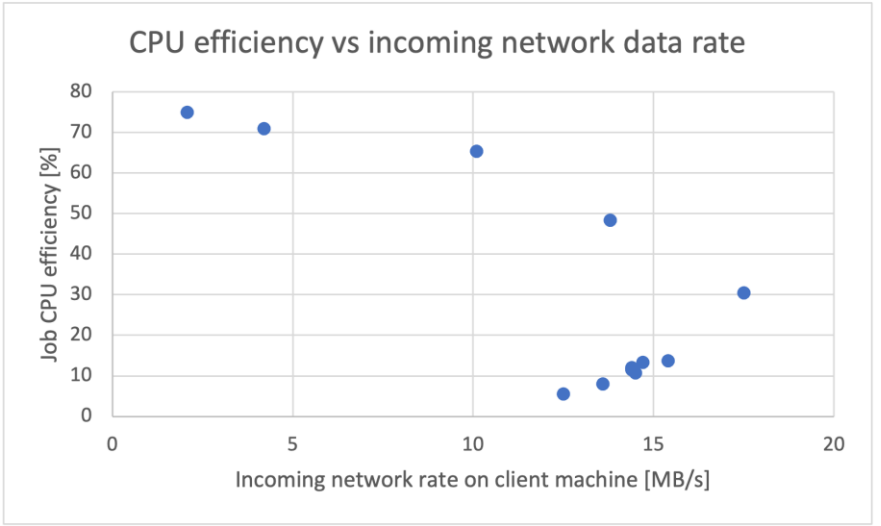
Configuration	Rate (MB/s)	CPU eff (%)	active devs (%)	rreq/s/dev	MB/s/dev	kB/rreq	util (%)
EOS, Nominal	-	99.0	-	-	-	-	-
XCache HDD, 2 nodes	13.6	99.3	16.9	0.338	0.137	406	0.0787
XCache SSD, 2 nodes	13.0	99.5	70.3	1.91	0.781	404	0.162
Local filesystem, 16 SSD	-	99.5	92.6	4.18	1.76	421	0.386

- Extremely low I/O: very low rates and almost 100% CPU efficiency
- The storage is de facto irrelevant

Disk utilization and job performance: an example

- **Used cms-skimmer and varied the number of concurrent jobs from 1 to 64**
  - Each accessing a different data file via xrootd from a plain xrootd server
  - All files on a single HDD
- **Measured total data rate vs. disk utilization vs. CPU efficiency**
  - Increasing number of jobs increases disk utilization and total rate until the former saturates and the rate eventually falls
  - CPU efficiency decreases with increasing number of jobs and does so dramatically once 100% disk utilization is achieved
- **Storage scalability is completely dependent on the access patterns and the application**





XCache scalability estimates

- **Estimate the number of client threads that would saturate the XCache instance**
  - Network limit: saturation of the network interface bandwidth (2800 MB/s)
  - Disk utilization limit: saturation of the storage devices
  - Results are rescaled to assume a 100% CPU efficiency, so they are a lower limit

Workload	Network limit	Disk limit (96 HDD)	Disk limit (16 SSD)
top-xaod	14000	40000	20000
cms-skimmer	1500	1800	2800
aggregate gradients	20	130	60

- **Planning hardware-friendly hardware requires to first study the relevant workloads!**
  - Match the disk performance and the network
  - The optimal storage architecture depends (strongly) on the application

Data access performance wrap-up

- **Observations generally match the expectations**
  - Higher rates with sequential patterns (difference less important for SSDs)
  - High throughput jobs have lower CPU efficiencies
  - Access to locally attached storage is fastest (no network bottlenecks)
  - On a single HDD, data rate approximately proportional to disk utilization until near saturation
  - The scalability of an XCache instance is critically dependent on the workload

Conclusions

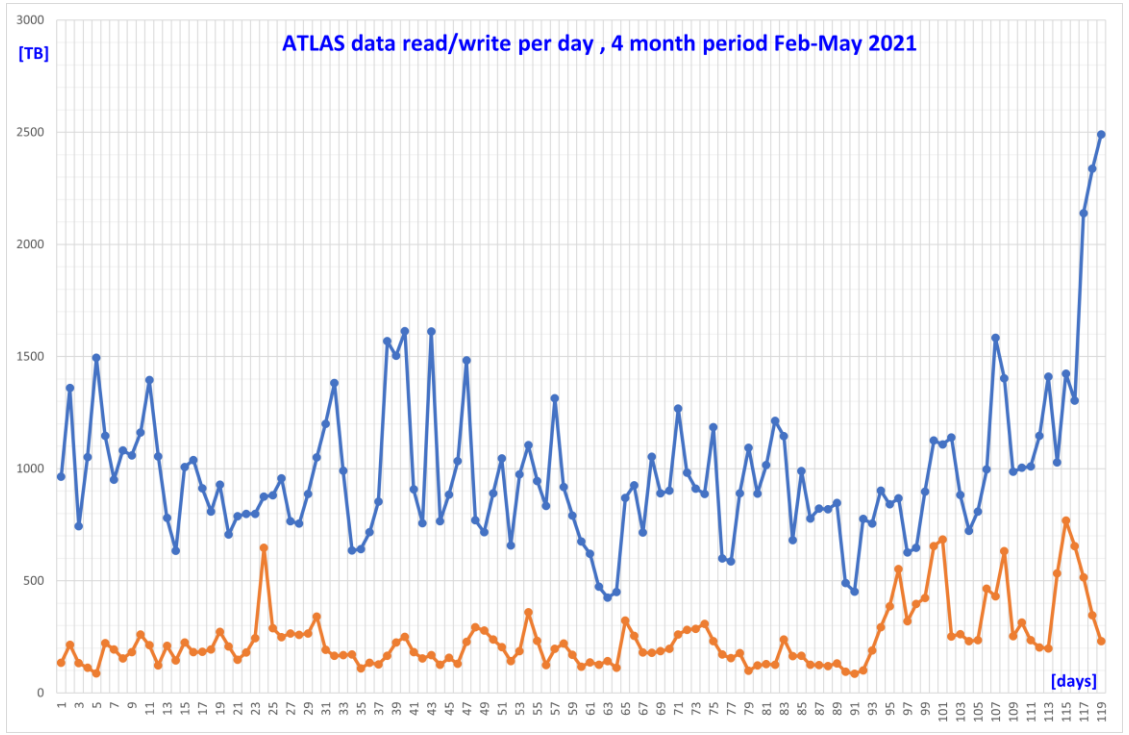
- **The scope of these investigations is physics analysis, the goal is to determine if new resources had to be invested on for Run 3**
  - Find out if there are bottlenecks in EOS for LXPLUS/LXBATCH analysis
  - Study data access performance of (interactive) analysis on high performance nodes for different storage configurations
- **Results**
  - The existing hardware in the IT CC is generally able to handle current analysis activities
    - High performance/throughput servers can be requested by physics groups
  - The EOS disk servers operate well below saturation with the current hardware and software
- **Focus on analysis is rapidly increasing in many communities**
  - CERN IT, SFT, experiments, HSF, WLCG, …
  - Continue performing performance studies to improve our understanding of analysis use cases
  - Encourage interaction with analysis users to discuss together their needs

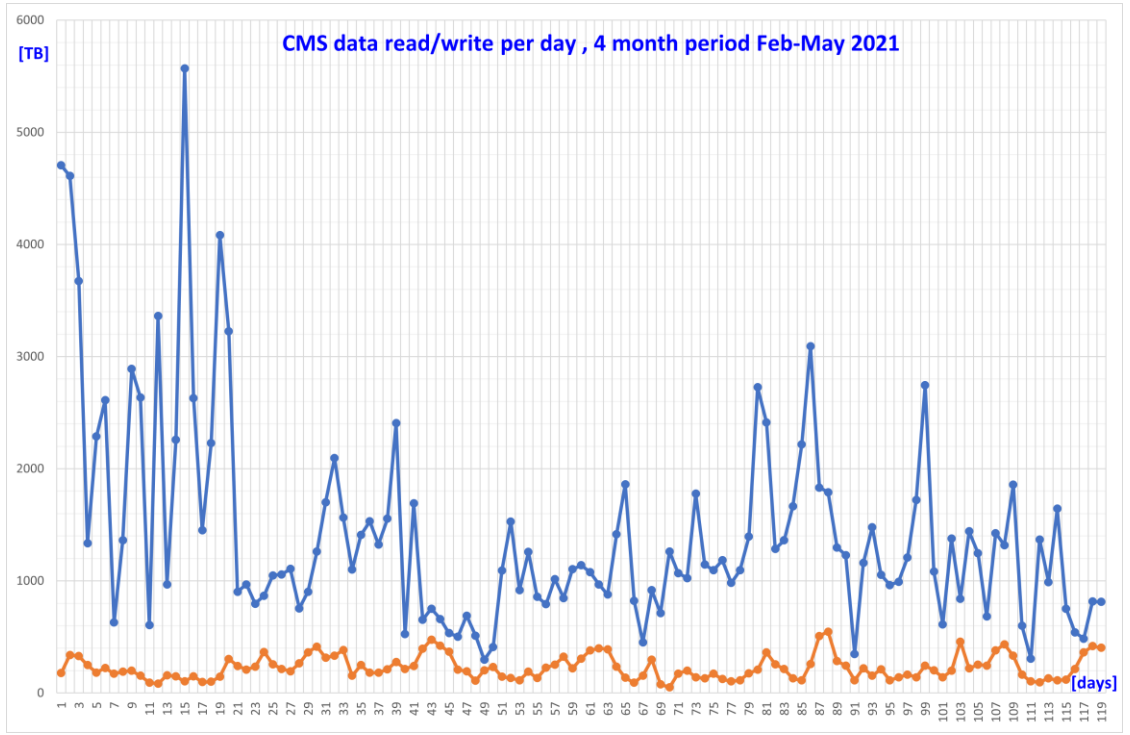
Acknowledgements

- **Many thanks to all members of the working group**
  - Josh Bendavid, Alessandro Di Girolamo, Enrico Giraud, Lukas Heinrich, Michael Holzbock, Ben Jones,Oliver Keeble, Matt Leblanc, James Letts, Zach Marshall, Helge Meinhard, Tadej Novak, Andreas Peters, Danilo Piparo
- **And to**
  - Riccardo Castellotti, Andrey Kiryanov, the SWAN team, the Spark/HADOOP team

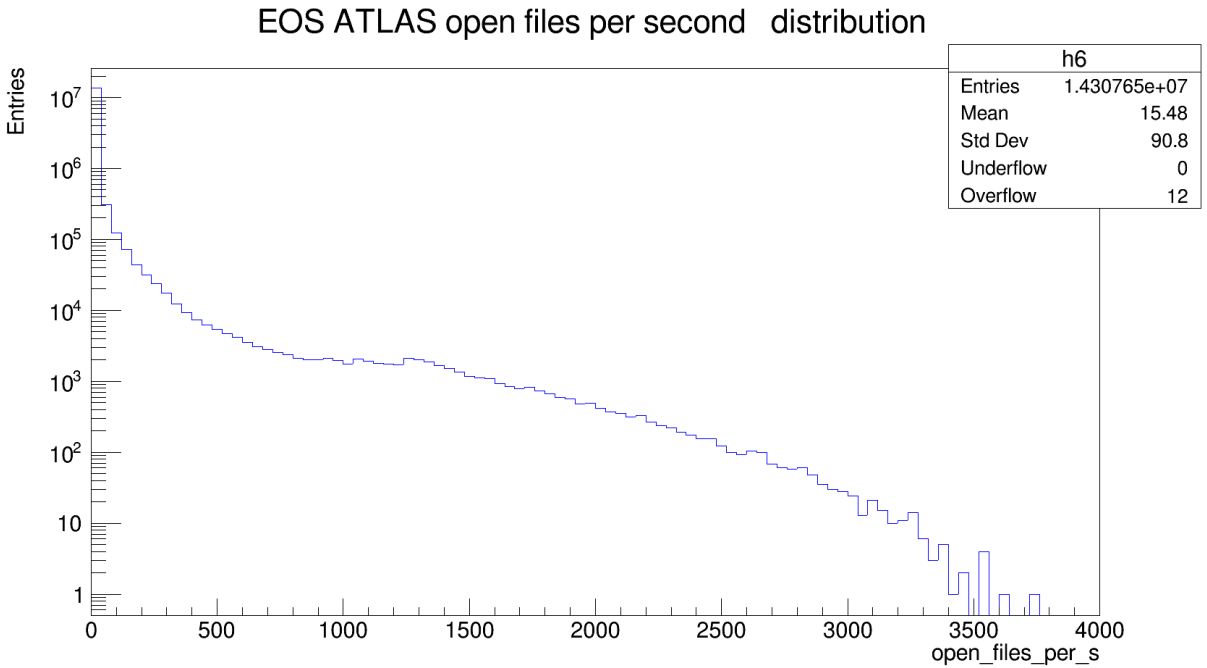
Backup slides

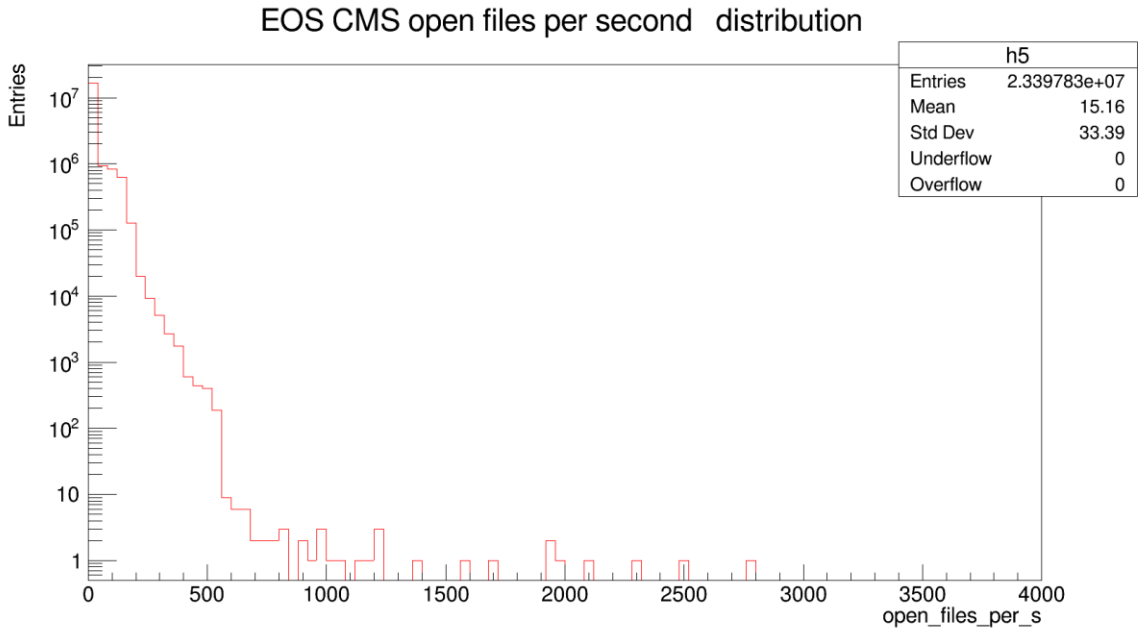
EOS daily amounts of data read/written



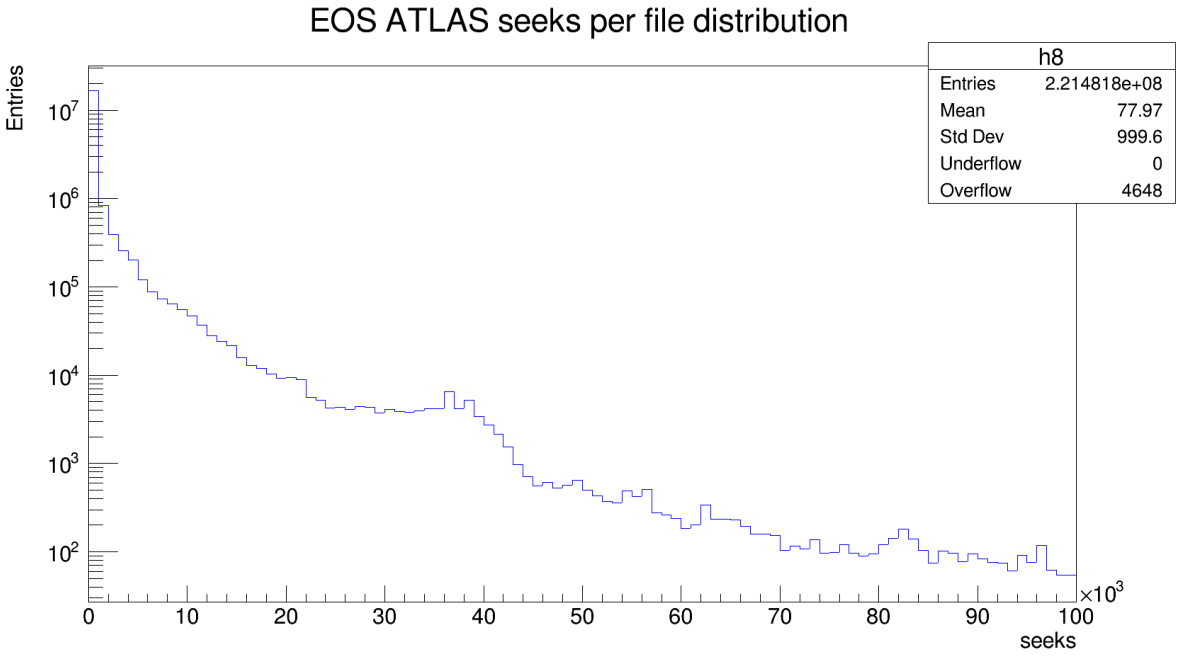


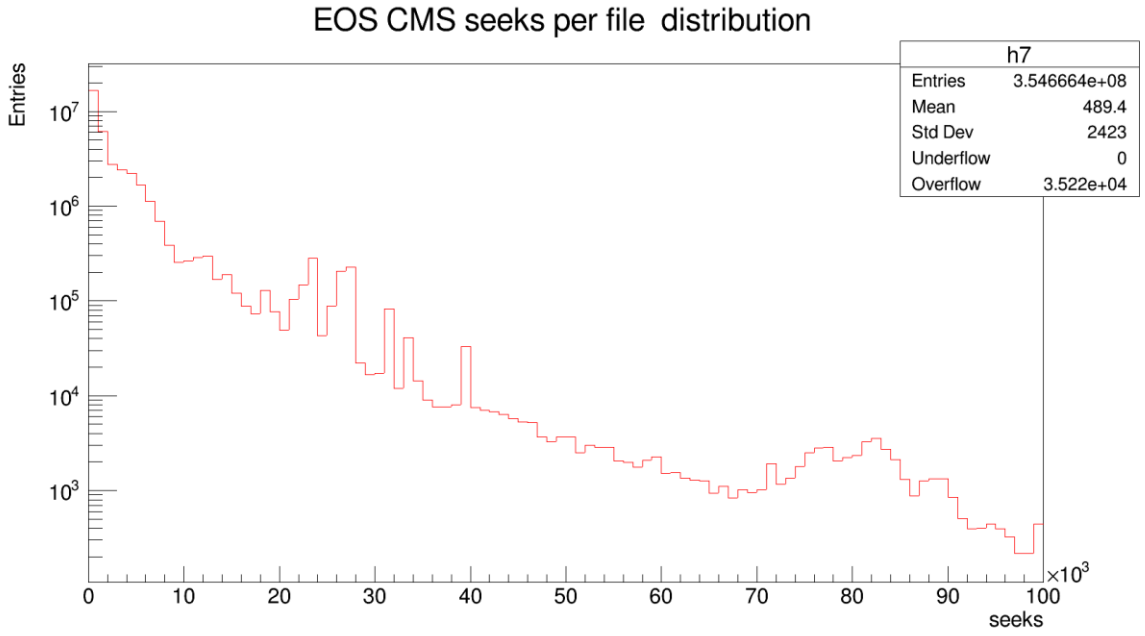
File opens per second





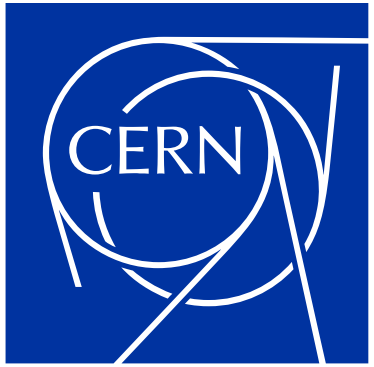
Seeks per file





Cms-skimmer jobs against a single HDD: all metrics

Concurrent jobs	Net (MB/s)	CPU eff (%)	rreq/s/dev	MB/s/dev	kB/rreq	util (%)
1	2.07	75.0	46.8	4.94	106	22.3
2	4.21	70.9	85.0	9.49	112	41.3
4	10.1	65.3	155	20.3	131	64.9
8	13.8	48.3	223	28.3	127	92.7
16	17.5	30.4	244	33.7	138	99.6
32	14.7	13.3	239	30.3	127	100
33	15.4	13.7	235	30.8	131	100
34	14.4	12.0	230	29.4	128	99.5
36	14.4	11.5	232	29.6	127	100
40	14.5	10.7	229	29.4	128	100
48	13.6	7.97	221	27.3	124	100
64	12.5	5.51	215	23.7	111	100



home.cern