CERN

Evaluating CephFS Performance vs. Cost on High-Density Commodity Disk Servers

Andreas J. Peters, Daniel C. van der Ster
CERN IT Storage Group

20 May 2021

vCHEP
2021

Introduction

- **High luminosity** data taking will increase demands on WLCG storage:
  - Throughput, Capacity, Durability
- **Open source storage** systems (OSS) have compelling features and maturity
  - Which role can they play in future physics storage systems?
- **Off-the-shelf** software misses important high-level features
  - And offer unknown efficiency on our cost-optimized hardware
- One solution is to layer **HEP-specific gateways** on top of the OSS systems

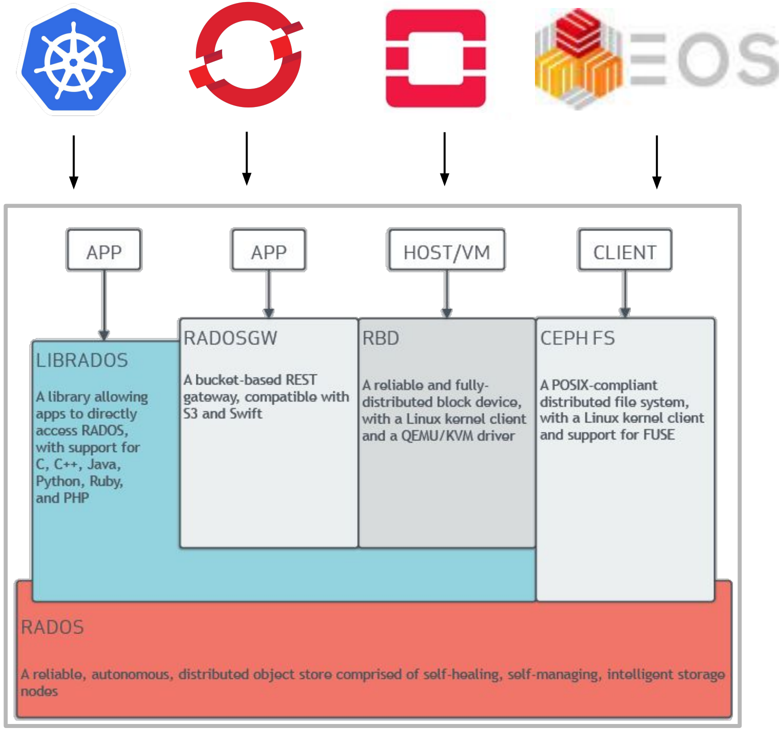
- We describe and evaluate a **novel combination** of storage systems:

CephFS + EOS

Introduction - Ceph

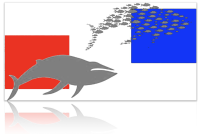
- Ceph is a popular backend component in the Open Infrastructure stack
  - **Kubernetes** + **OpenShift** to deploy containerized apps
  - **OpenStack** to manage virtual and physical resources
  - **Block/Object/File** storage on Ceph

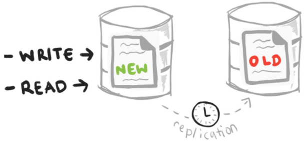
- Is it efficient to layer EOS on top to expose the infrastructure to the WLCG?


EOS
APP
APP
HOST/VM
CLIENT
LIBRADOS
A library allowing apps to directly access RADOS, with support for C, C++, Java, Python, Ruby, and PHP
RADOSGW
A bucket-based REST gateway, compatible with S3 and Swift
RBD
A reliable and fully-distributed block device, with a Linux kernel client and a QEMU/KVM driver
CEPH FS
A POSIX-compliant distributed file system, with a Linux kernel client and support for FUSE
RADOS
A reliable, autonomous, distributed object store comprised of self-healing, self-managing, intelligent storage nodes

CephFS and its application at CERN

- **Clustered filesystem** used in NFS-like scenarios:
  - Home directories, HPC scratch areas, shared storage for distributed applications
- **Scale-out** architecture for data and metadata:
  - Data and FS hierarchy persisted in RADOS, a reliable distributed object store.
  - Durability with replication (e.g. 3 copies) or erasure coding (e.g. EC4,2).
  - Objects spread across failure domains with CRUSH -- tolerate host/rack/row/switch failures.
- **Read-after-write consistency** just like a local filesystem
  - MDS servers delegate **client capabilities** to allow operations to be carried out async or sync as needed: buffered IO when possible, direct IO when needed.



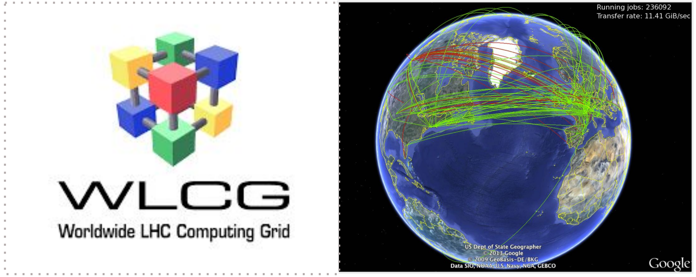


CephFS and its application at CERN

- CERN operates CephFS **in production since 2017**; currently:
  - HPC Scratch: all-flash co-located on SLURM compute nodes, switch-local MDS
    - 3x replication, 110TiB usable capacity
  - OpenStack Manila: mixed HDD/SSD for general purpose shared storage in the cloud
    - 3x replication, 1PiB usable capacity
  - Enterprise Groupware: all-flash co-located on OpenStack hypervisors
    - EC2,2; 100TiB usable capacity
- These and ~30PiB of other Ceph clusters have been **robust and performing**
  - Data consistent after infrastructure outages; failure recovery is basically transparent
  - Hardware replacement and flexibility demonstrated across 3 procurement cycles

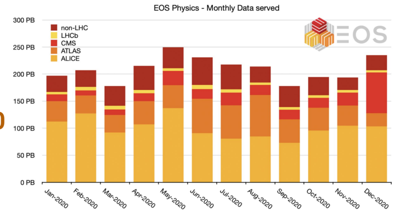
CephFS and its application at CERN

- CephFS **misses features and experience** essential to the HEP community:
  - **Authentication** mechanisms and user/group management:
    - *SciTokens, X.509, Kerberos*
    - *quota and access control via eGroups*
  - Storage **protocols** and **features**:
    - *HTTPS, XRootD, third-party copy*
  - Experience with high storage **throughputs**:
    - *sustained streaming write performance for LHC data-taking*
    - *T0 rates 10-100 GiB/s*

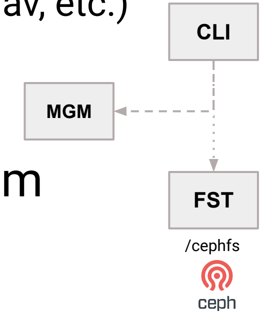


EOS Open Storage

2.5 Exabyte
read in 2020



- Large scale storage **developed at CERN for physics** and regular users: **350PB**
- Implemented as plugins to the XRootD framework:
  - Files stored replicated or erasure-coded; presented in hierarchical namespace using QuarkDB.
  - FST services provide access to data stored in local or remote storage (XFS, Webdav, etc.)
  - MGM caches metadata and maps filenames to inodes; FSTs store data by inode
    - Local or remote FST storage is a simple inode hash prefix and hex inode
- It’s therefore **straightforward to use CephFS** as a **local** FST filesystem
  - Redundancy and data high-availability delegated to the CephFS layer
  - EOS configured to store data with a single copy
  - FST filesystems can be **moved in production** from node to node in case of failures



CLI

MGM

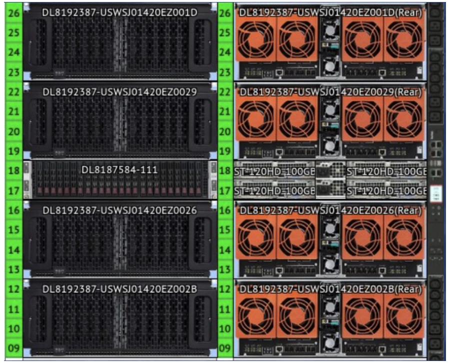
FST

/cephfs

ceph

System Architecture

- **PoC** implemented using eight servers with specs:
  - **Dual Intel Xeon** Silver 4216 CPUs and **192 GiB RAM**
  - Mellanox ConnectX-5 network interface supporting **100Gb/s Ethernet**
  - **60x 14TB enterprise SATA HDDs** connected via a SAS3616 HBA
  - 2x 1TB SSDs: one for operating system, one for Ceph
  - ~3GiB RAM per HDD
- **Different** from the current EOS production hardware:
  - 96x HDDs with 192 GiB RAM
    - too little RAM per HDD for efficient Ceph OSD operations

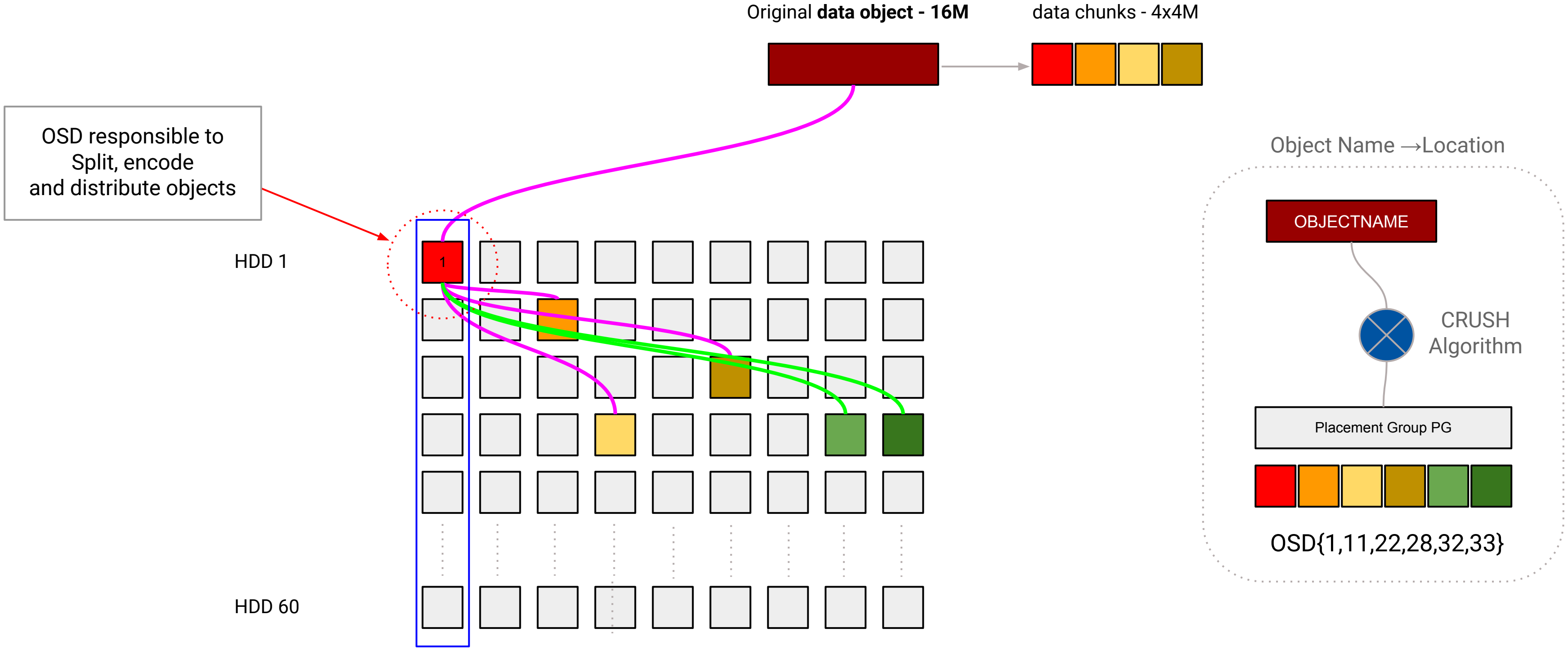

26
25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
09
DL8192387-USWSJ01420EZ001D
DL8192387-USWSJ01420EZ0029
DL8187584-111
DL8192387-USWSJ01420EZ0026
DL8192387-USWSJ01420EZ002B
DL8192387-USWSJ01420EZ001D(Rear)
DL8192387-USWSJ01420EZ0029(Rear)
ST-120HD-100GE
ST-120HD-100GE
ST-120HD-100GE
ST-120HD-100GE
DL8192387-USWSJ01420EZ0026(Rear)
DL8192387-USWSJ01420EZ002B(Rear)

High Density JBOD

Ceph Backend Storage

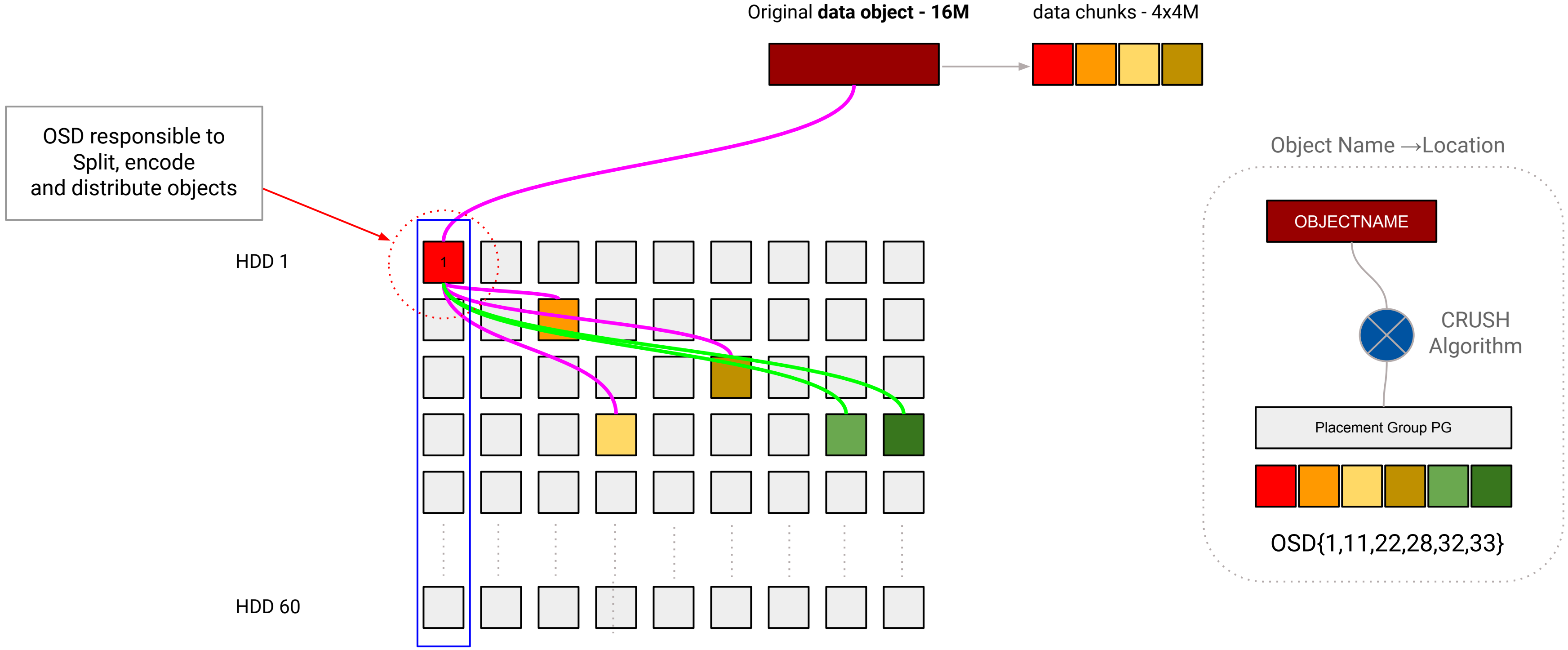
- **Ceph Octopus** version **15.2.8**
  - Single VM, not local to the disk servers, runs the MON, MGR, MDS daemons
- CephFS with various RADOS pool configurations:
  - **`cephfs_metadata`** on SSD-only OSDs; **`cephfs_data`** pools on HDD-only OSDs
  - **/ec42**: Reed-Solomon coding with k=4, m=2
    - each host has at most one object chunk; 4096 placement groups, 51.2 per OSD
  - **/ec82**: Reed-Solomon coding with k=8, m=2
    - each host has at most two object chunks; 2048 placement groups, 42.6 per OSD
  - **/ec162**: Reed-Solomon coding with k=16, m=2
    - each host has at most three object chunks; 1024 placement groups, 38.4 per OSD
  - The RADOS placement groups were balanced to a max deviation of one per OSD.

RADOS Erasure Coding - IO path for Writing

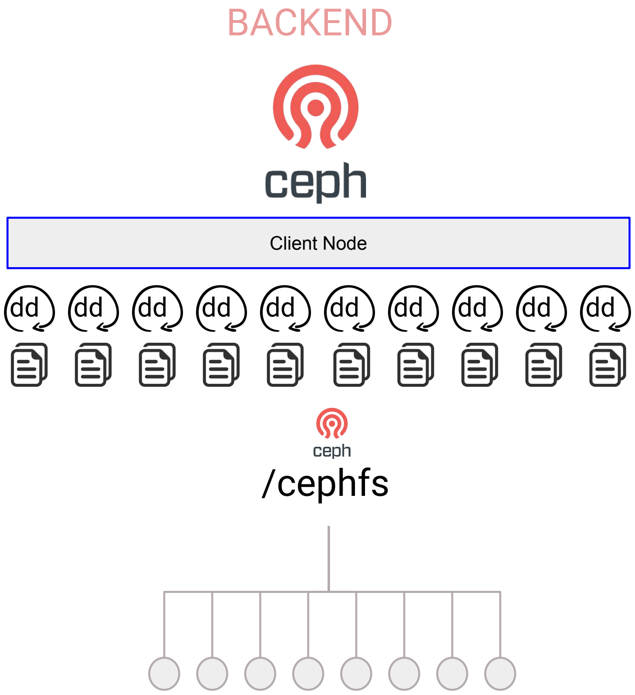


RADOS Erasure Coding - IO path for Writing

Different object sizes impact on performance

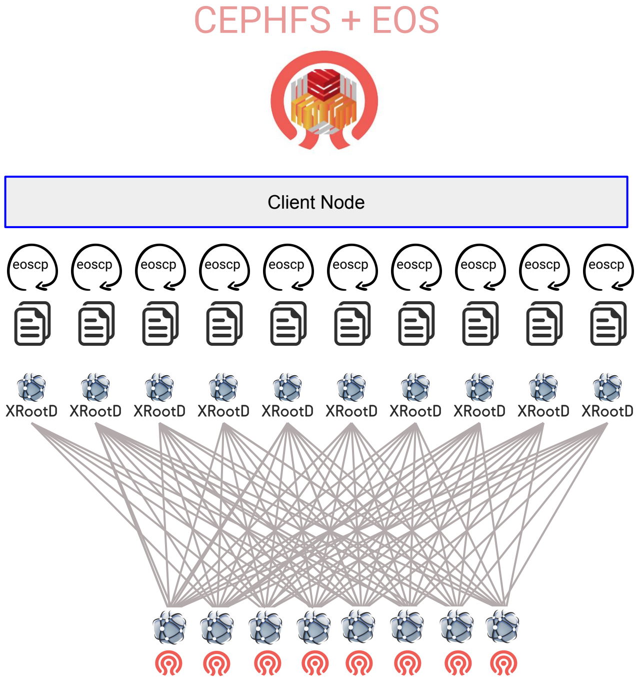


Test Setup on client nodes

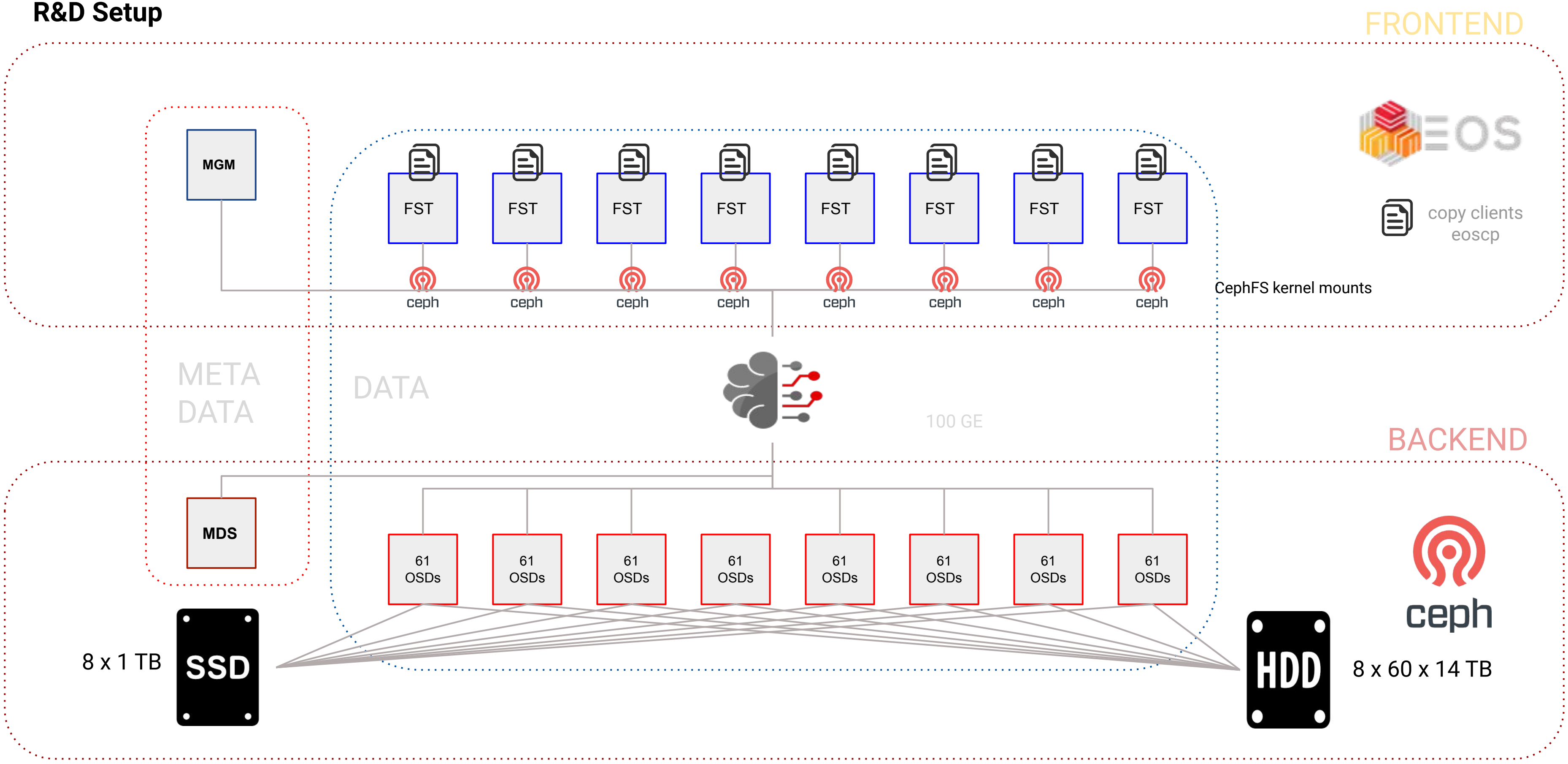


10 looping copy applications per client node

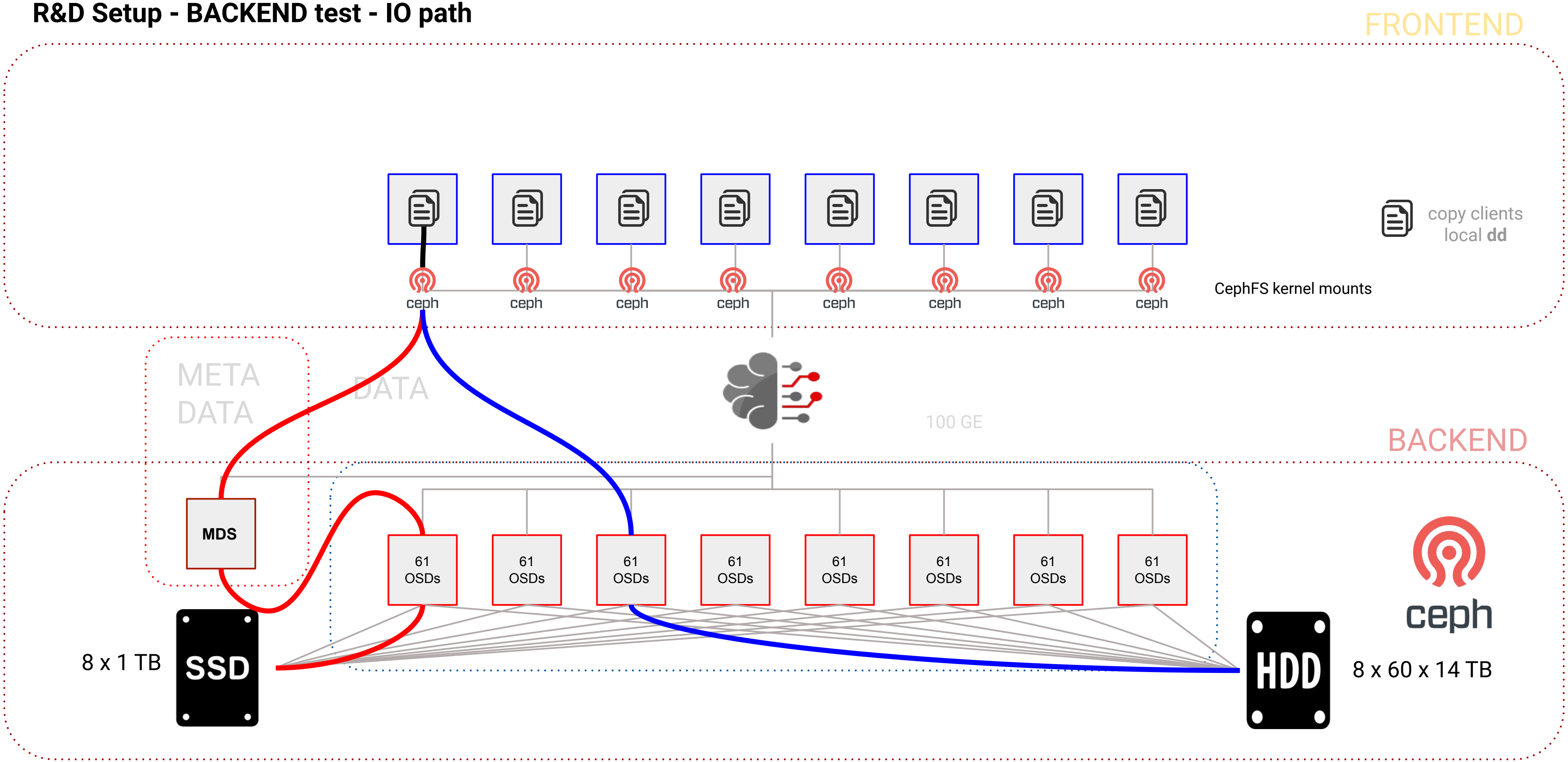
- *dd* for backend testing
- *eoscp* using `root://` for frontend testing
- 80 clients; 2GB files



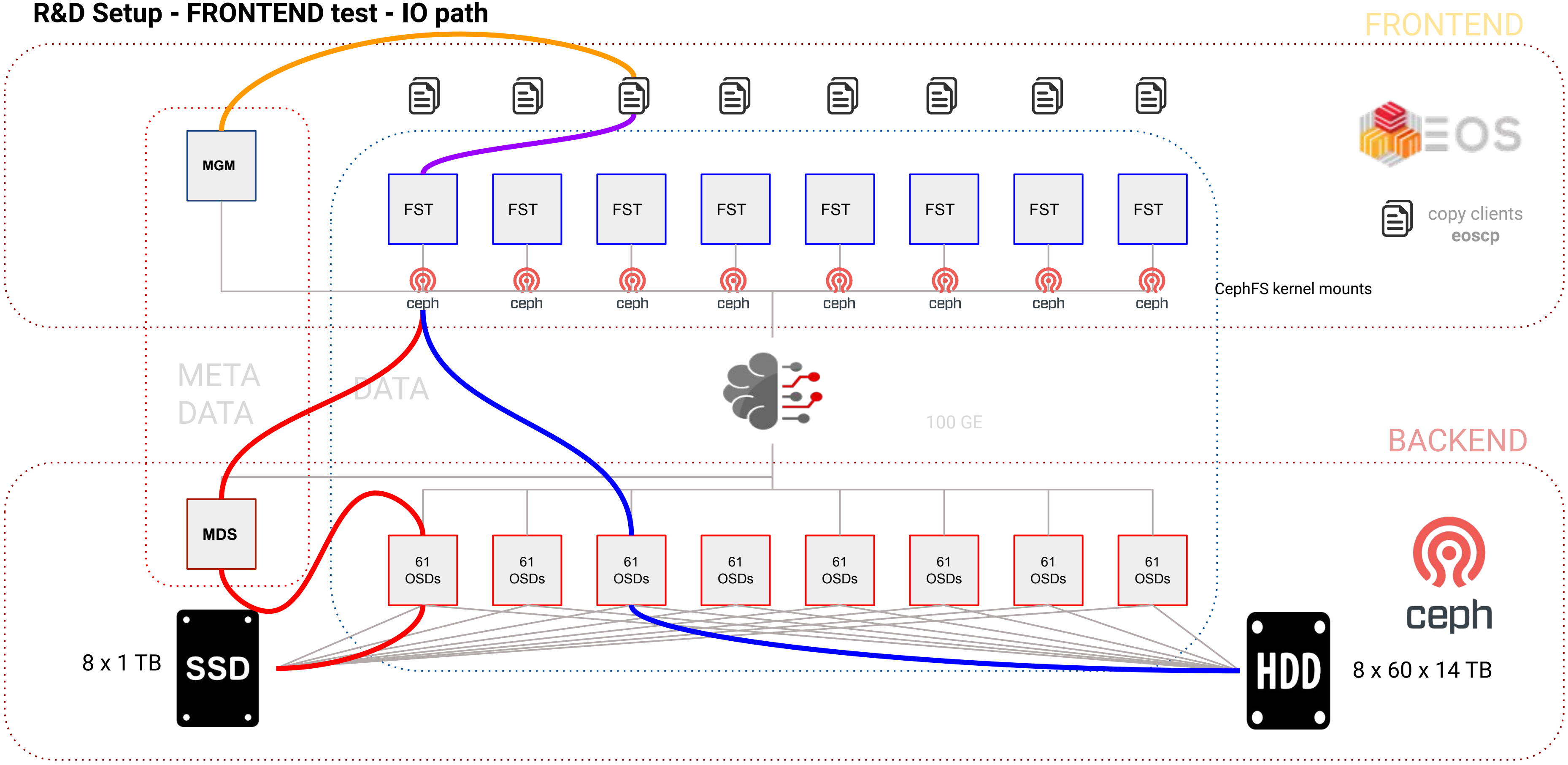
R&D Setup


FRONTEND
EOS
copy clients
eoscp
MGM
FST
FST
FST
FST
FST
FST
FST
FST
ceph
ceph
ceph
ceph
ceph
ceph
ceph
ceph
CephFS kernel mounts
META
DATA
DATA
100 GE
BACKEND
MDS
61
OSDs
61
OSDs
61
OSDs
61
OSDs
61
OSDs
61
OSDs
61
OSDs
61
OSDs
ceph
8 x 1 TB
SSD
HDD
8 x 60 x 14 TB

R&D Setup - BACKEND test - IO path

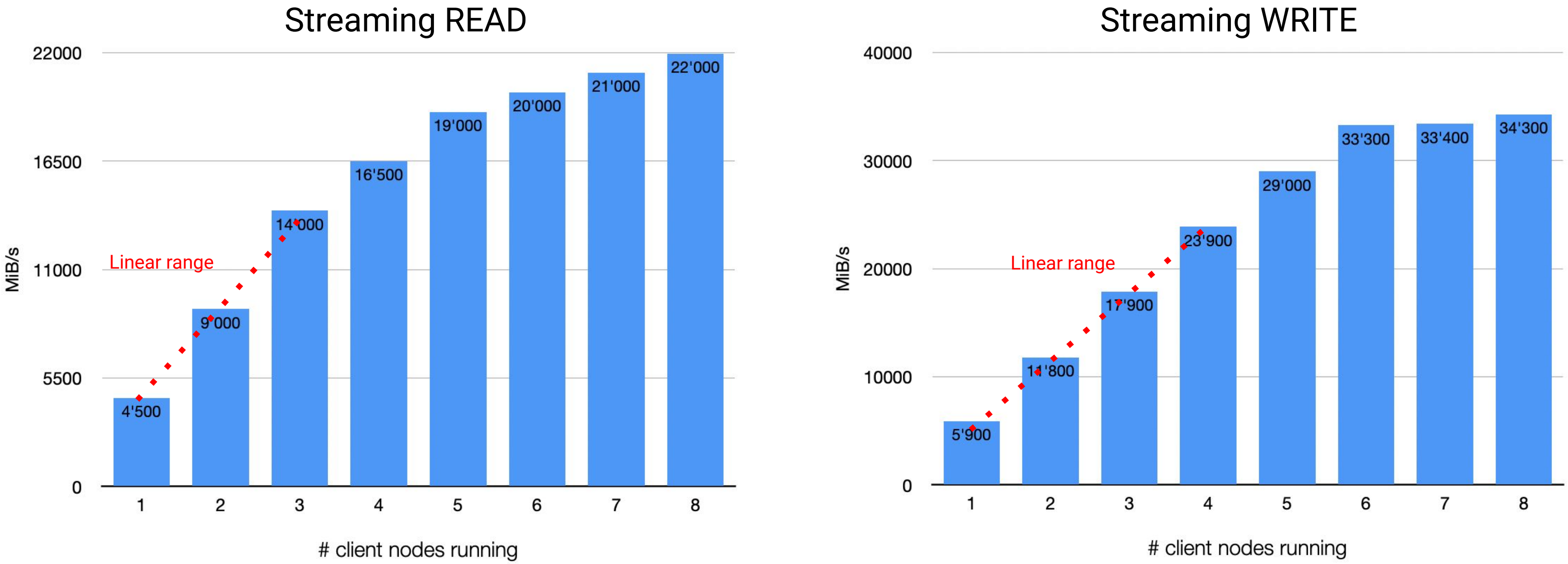


R&D Setup - FRONTEND test - IO path



CephFS Client Scalability Measurements

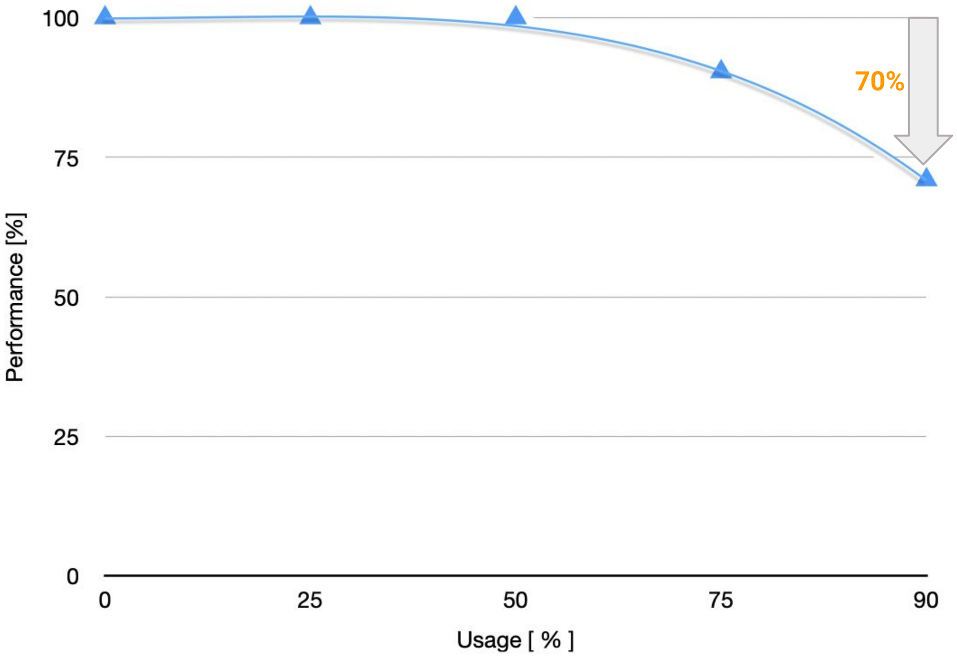
Aggregated instance streaming bandwidth vs number of active client nodes with EC4,2 CephFS mount



Streaming READ

Streaming WRITE

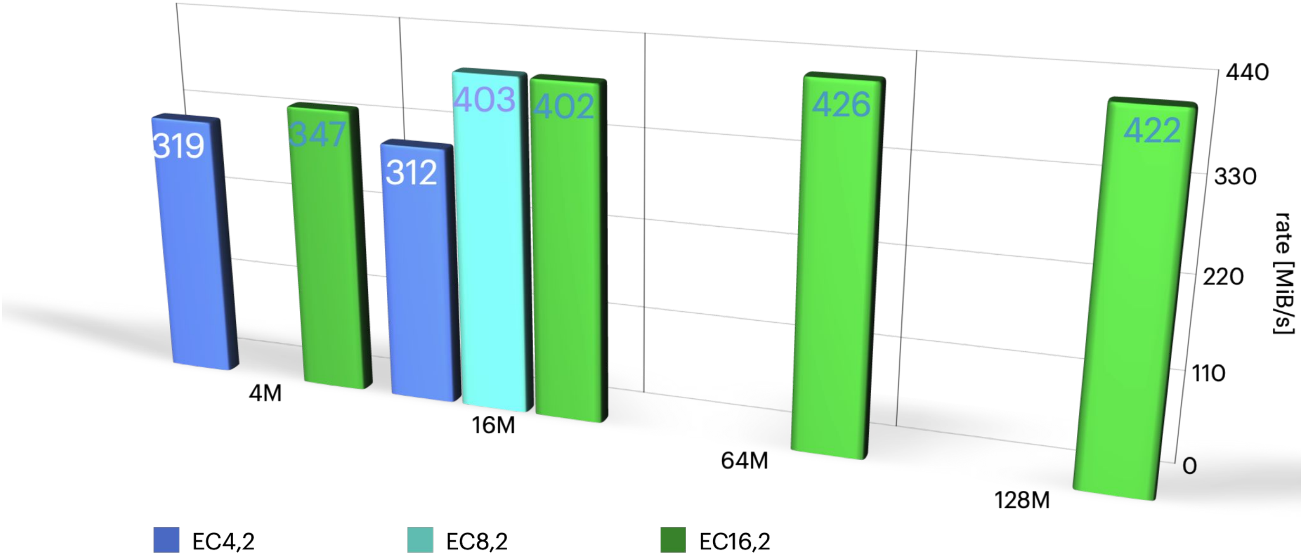
Impact of OSD usage on write performance



- Demonstrated to fill OSDs up to 95% with active reweighting
- Over 50% volume usage write performance degrades slowly towards 70% of initial instance performance
- Correlated with increased IO wait on disks

Streaming Write performance

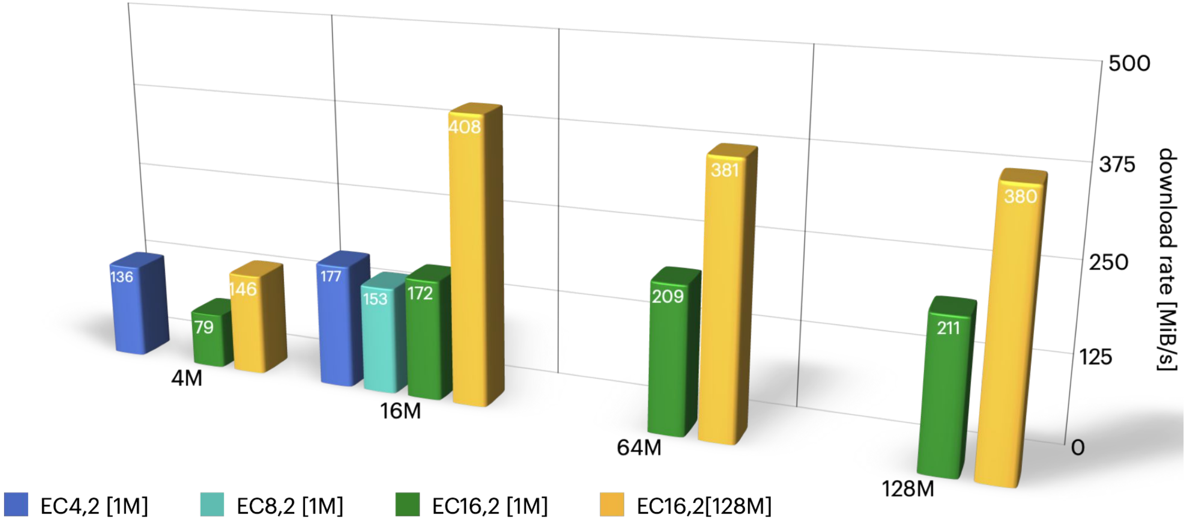
Write performance for various erasure coding layouts EC k,m and object sizes. The test runs 80 concurrent streams and files with 2 GB size. Shown is the single stream IO bandwidth.



- Per stream write bandwidth sensitive to object size
- **Default 4M not optimal - favour larger sizes**
  - Trade-off performance Memory consumption

Streaming Read Performance

Read performance for various erasure coding layouts EC (k,m), object sizes (x axes) and [1M] and [128M] *IO blocksizes*. The test runs 80 concurrent streams and files with 2 GB size and a default kernel read-ahead setting of 8MB. Shown is the single stream IO bandwidth.

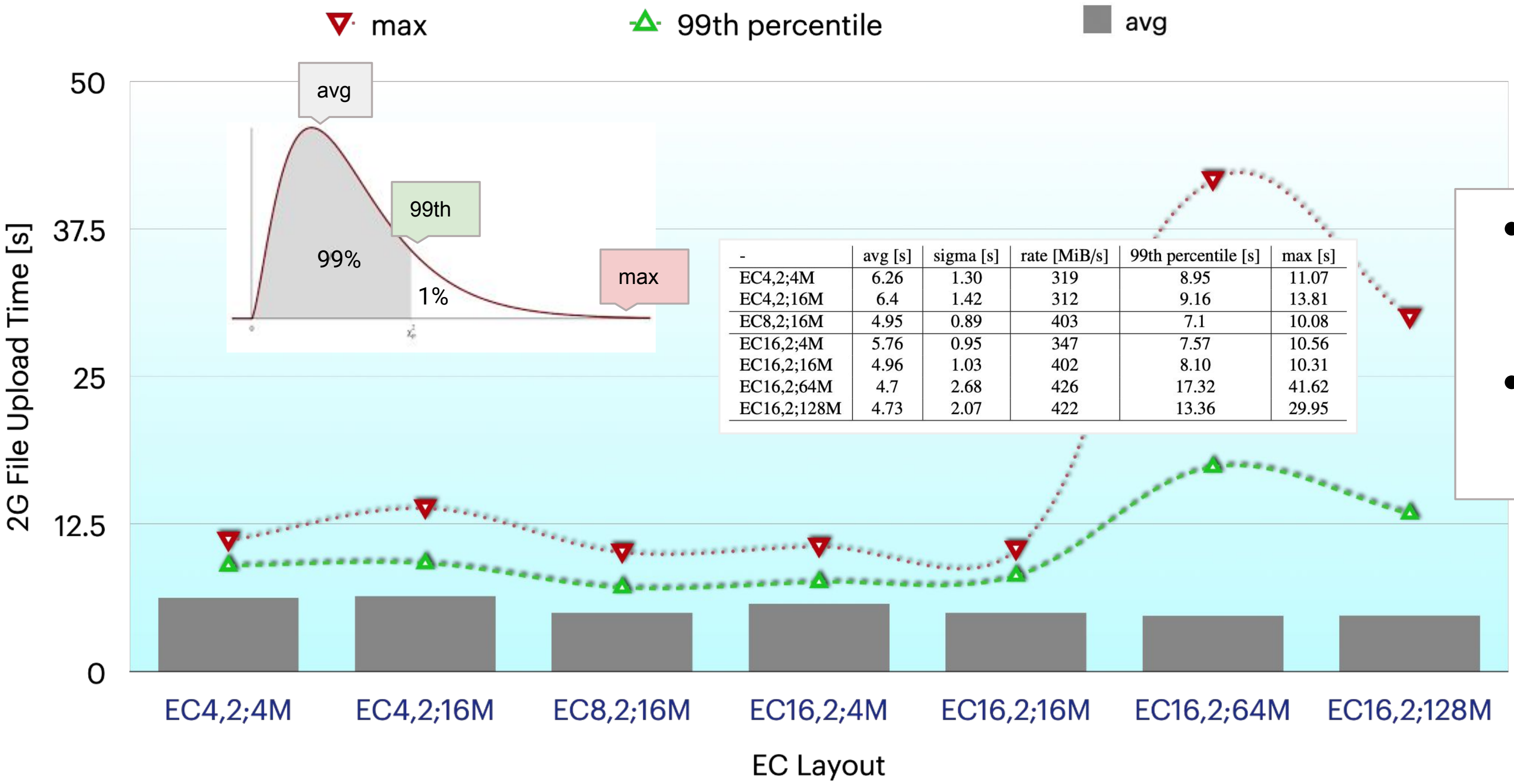


- Per-stream read bandwidth is sensitive to object and IO blocksize
- Performance is modest for small IO blocksizes
- Best read performance is obtained for the largest tested IO blocksize (128M)

Def'n: "blocksize" is the size of each r/w IO

Write performance tails

BACKEND

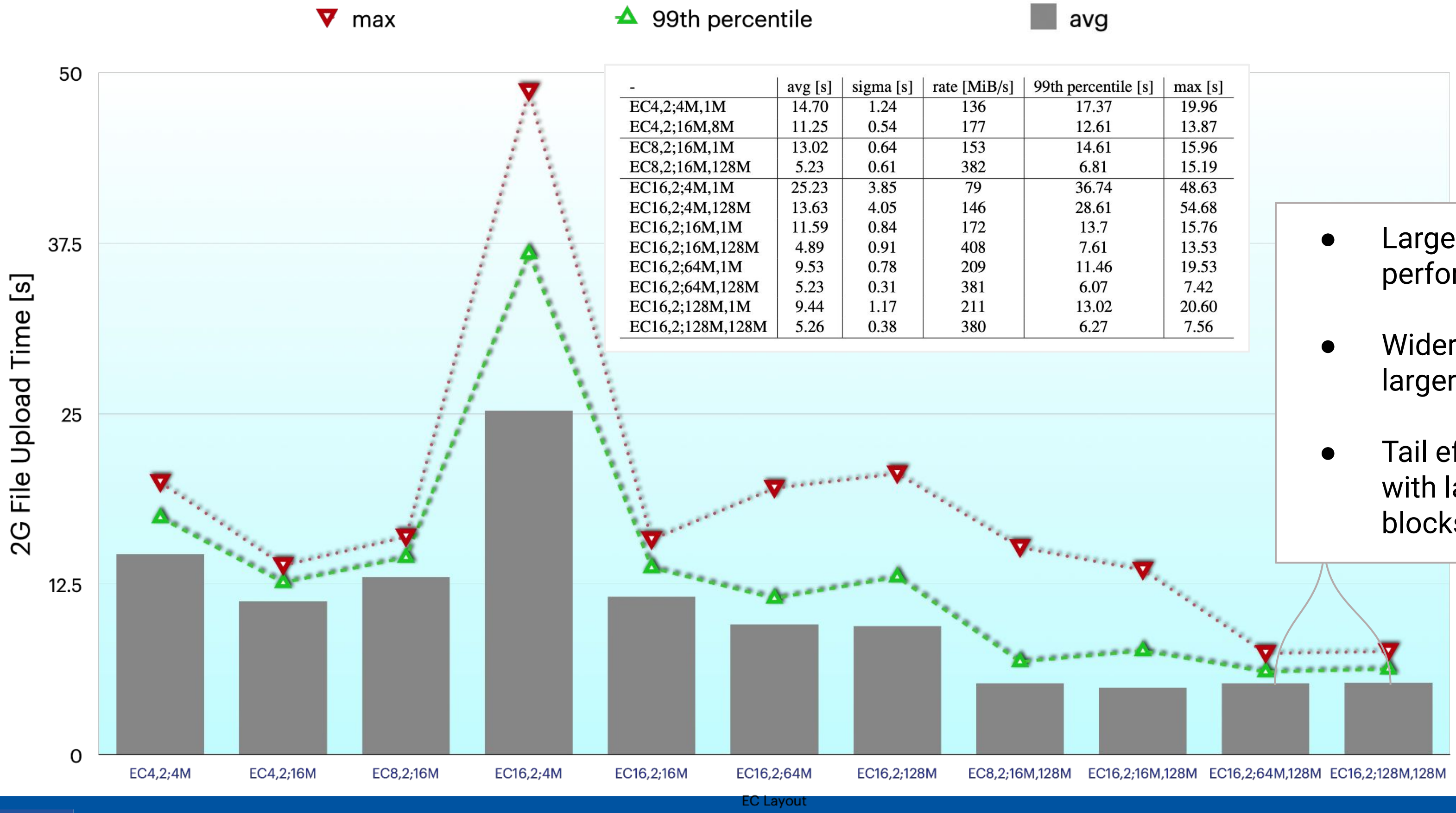


-	avg [s]	sigma [s]	rate [MiB/s]	99th percentile [s]	max [s]
EC4,2;4M	6.26	1.30	319	8.95	11.07
EC4,2;16M	6.4	1.42	312	9.16	13.81
EC8,2;16M	4.95	0.89	403	7.1	10.08
EC16,2;4M	5.76	0.95	347	7.57	10.56
EC16,2;16M	4.96	1.03	402	8.10	10.31
EC16,2;64M	4.7	2.68	426	17.32	41.62
EC16,2;128M	4.73	2.07	422	13.36	29.95

- Wider EC layouts provide higher performance
- **Large object sizes perform better but tail effects increase**

Read performance tails

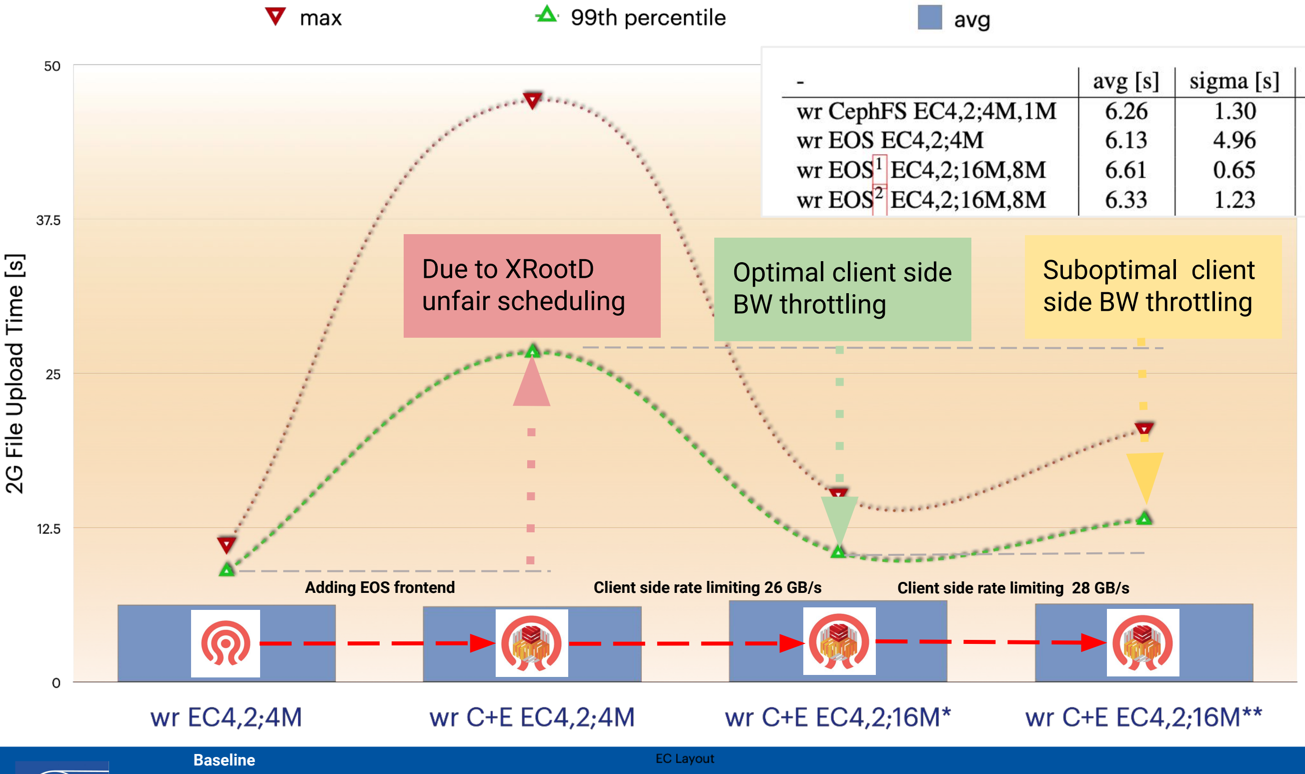
BACKEND



-	avg [s]	sigma [s]	rate [MiB/s]	99th percentile [s]	max [s]
EC4,2;4M,1M	14.70	1.24	136	17.37	19.96
EC4,2;16M,8M	11.25	0.54	177	12.61	13.87
EC8,2;16M,1M	13.02	0.64	153	14.61	15.96
EC8,2;16M,128M	5.23	0.61	382	6.81	15.19
EC16,2;4M,1M	25.23	3.85	79	36.74	48.63
EC16,2;4M,128M	13.63	4.05	146	28.61	54.68
EC16,2;16M,1M	11.59	0.84	172	13.7	15.76
EC16,2;16M,128M	4.89	0.91	408	7.61	13.53
EC16,2;64M,1M	9.53	0.78	209	11.46	19.53
EC16,2;64M,128M	5.23	0.31	381	6.07	7.42
EC16,2;128M,1M	9.44	1.17	211	13.02	20.60
EC16,2;128M,128M	5.26	0.38	380	6.27	7.56

- Large IO blocksizes perform better
- Wider EC layouts prefer larger object sizes
- Tail effects minimized with large object & blocksizes

CephFS+EOS Write Performance Impact?



Due to XRootD unfair scheduling

Optimal client side BW throttling

Suboptimal client side BW throttling

Adding EOS frontend

Client side rate limiting 26 GB/s

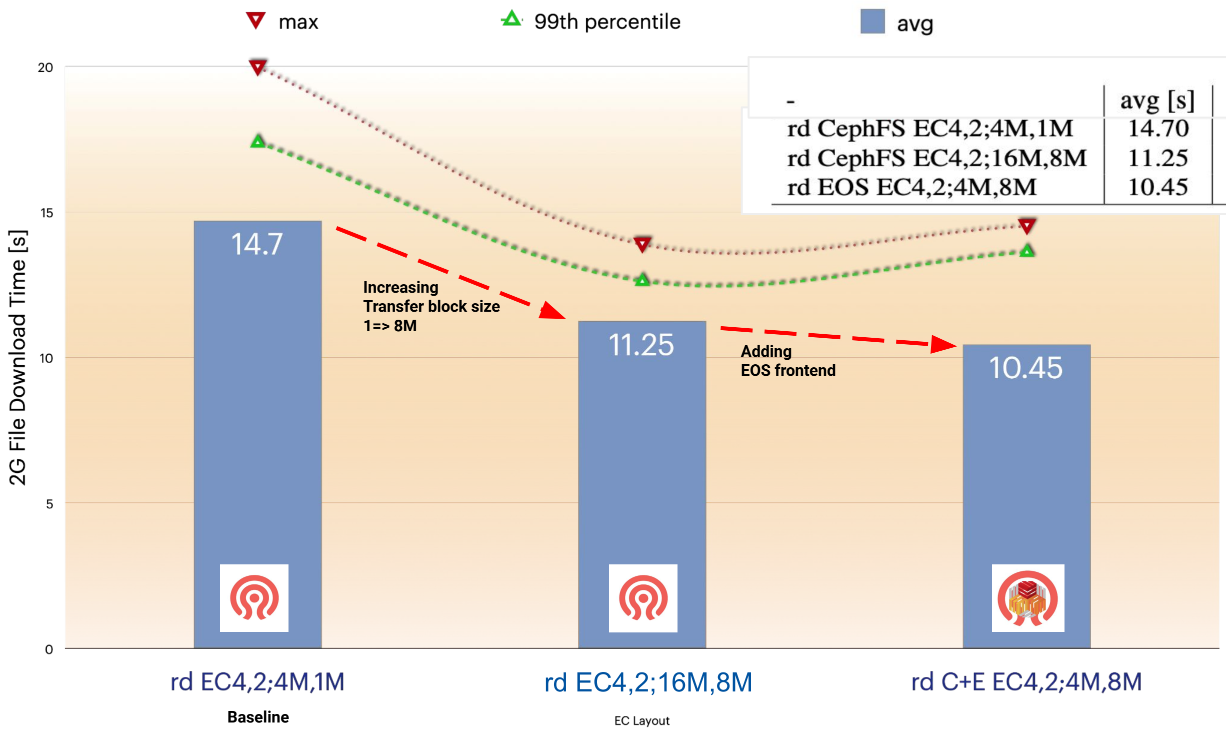
Client side rate limiting 28 GB/s

-	avg [s]	sigma [s]	rate [MiB/s]	99 perc. [s]	max [s]
wr CephFS EC4,2;4M,1M	6.26	1.30	319	8.95	11.07
wr EOS EC4,2;4M	6.13	4.96	326	26.67	47.10
wr EOS[1] EC4,2;16M,8M	6.61	0.65	302	10.43	15.03
wr EOS[2] EC4,2;16M,8M	6.33	1.23	315	13.11	20.34

- Observation: Adding frontend does not change averages but creates long tail effects
- Tails can be reduced using client-side bandwidth throttle

*[1] = 325 MiB/s **[2] = 350 MiB/s

CephFS+EOS Read Performance Impact?



-	avg [s]	sigma [s]	rate [MiB/s]	99 perc. [s]	max [s]
rd CephFS EC4,2;4M,1M	14.70	1.24	136	17.37	19.96
rd CephFS EC4,2;16M,8M	11.25	0.54	177	12.61	13.87
rd EOS EC4,2;4M,8M	10.45	1.25	191	13.26	14.5

- Observation: Adding frontend improves reads due to larger relative IO blocksize
- No long tails effects for reading

BACKEND

Tuning Ceph: Client Bytes on the network

- **RADOS** client **throttles** bytes in flight to 100MB by default, which is limiting in this environment
  - E.g suppose you want to dispatch 128 concurrent 4MB writes… that's 500MB in flight

- Fix: simply increase this throttle for our PoC:

```
objecter_inflight_op_bytes = 1GB
```

- Note: does not apply to the kernel CephFS client; only fuse or librados clients

Tuning Ceph: Client capabilities recall

- **EOS `fsck`** is continually scanning all backend inodes in CephFS
  - Puts pressure on the MDS to load/cache/trim thousands of inodes while staying under it’s configured memory limit
  - Each inode consumes around 3kB: 64GB cache holds around 21M inodes at a time
- Can cause MDS to **OOM** if inode caps are granted more rapidly than released:
  - E.g: for each client MDS grants inode caps at 50kHz, but only recalls them at 5kHz
    - (8*50kHz*3kB is more than 1GB per second of inode cache growth)

- Fix: increased recall rates proposed and released upstream: Ceph PR 38574
  - Also new capabilities acquisition throttle to prevent this even without tuning

Tuning Ceph: Unexplained Performance Drop

- One day the throughput **dropped** from **25GB/s to 5GB/s** with no obvious explanation
  - Observed in `eoscp` testing and confirmed with low level `rados bench`
- Root-cause: **one *sick* HDD**
  - Poor SATA connection: 2s latency on small IOs
  - No IO errors, No SMART errors, just slow
  - Quick fix: stop the OSD process so this HDD is no longer used, data backfilled elsewhere
- Ceph has **internal metrics and displays** to help identify these issues manually, but auto detection is difficult
  - E.g. `ceph osd perf` shows recent op_commit latencies for all OSDs
  - **Working on better op latency anomaly detection**
    - to complement existing high network latency detection and SMART failure prediction

Discussions: Performance and Capacity

- PoC demonstrated excellent performance with various EC schemes
  - Up to **4GiB/s** read and write per FST frontend node for streaming access

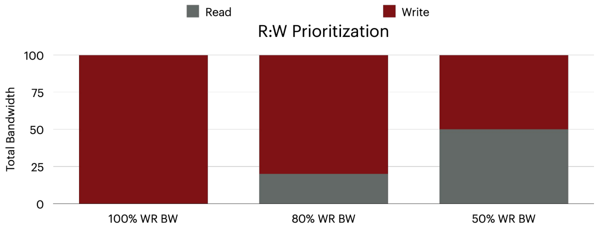
- We tested while filling CephFS up to **95% capacity**
  - ongoing automatic balancing OSD utilization with upmap is required

- **Operating** when the cluster is **nearly** full is **hazardous**:
  - Dramatic performance cutoff: probably caused by high disk fragmentation, IO seek latencies
    - might be improved by putting OSD block.db on flash
  - Must reserve adequate spare capacity to account for failure recovery:
    - keep at least (1-<failure domain>)% free

Danger
This is a
hazardous area

Discussions: Network Utilization

- Write **performance** is **limited** by **network** connectivity
  - CPU and disk IO util were not bottlenecks
  - Read performance probably limited by HDD seek latencies
- CephFS **EC model roughly doubles traffic** for reading and writing
  - Write tests: 9GiB/s inbound, 5GiB/s outbound, 5GiB/s disk output
  - **Doubling** the network **connectivity** could saturate the available disk IO throughput
    - Possible using 2x interfaces and Ceph's public/cluster network isolation
- Observed that CephFS prioritizes writes over reads: OK ☑
  - Ceph Pacific distributed QoS may allow this to be adjusted as needed



Discussions: FST performance

- EOS **frontend** has **marginal impact** on overall performance c.w. native RADOS

- **IO write tails** to be investigated w.r.t. sync stream scheduling in XRootD

- Possible to **co-locate EOS FST** and **Ceph OSDs**, e.g. mount CephFS locally
  - However: local mounts can cause kernel deadlock under memory pressure
  - Safe alternatives: user-mode CephFS client

    FUSE - ok for medium bandwidth requirements ☑

    libcephfs - low performance in multithreaded environments ⊗

Discussions and Conclusions

- **CephFS & EOS** are **easily stackable** and provide **excellent performance** on high-density commodity disk server and 100Gig-E technology
  - CephFS provides
    - an extremely reliable **high-performance** and **flexible** storage backend with tunable EC QoS
    - a large and active storage **user community** beyond HEP
  - EOS provides
    - high-level functionality as **strong authentication**
    - **remote access** protocols & third party copy (root/https)
    - fine-grained access and resource **control**
    - **add-on** services as
      - Sync&Share
      - Tape Storage

CERNBox

CERN Tape Archive

Future Outlook

- Prototype: **Test** in EOSHOME (CERNBox) to **evaluate** real life **usability, performance** and **operations** gains
    - Removes file-size limits & gives high bandwidth file IO; special high-IOPS or low latency areas

- Improvement: **Unify Namespaces and Localize IO**
    - Plug-in to use CephFS namespace as EOS namespace
        - Directly access the namespace without the EOS MGM
    - Local XRootD redirect from *root:*//eos to local */eos* read-only CephFS mounts
    - Hide CephFS kernel mount in *private namespace* within eosxd client (/eos mounter)
        - Better client-side security than native CephFS: krb5, GSI, OAUTH2 support
        - Allows to add HSM functionality in combination with CERN Tape Archive
        - IO redirect to local *fd* providing 100% native CephFS IO performance for the data path

Future Outlook

- **CephFS+EOS** is one of the candidates for **future WLCG storage** deployments

- Tier 1 example:
  - Leverage large existing RADOS installations to provide WLCG disk and tape storage
- Tier 2/3 example:
  - Reliable and cost-effective solution with flexibility for low-latency areas

CephFS + EOS

Thanks!

Extra slides

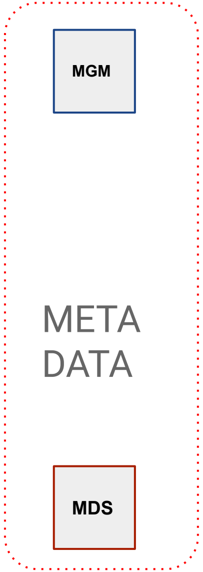
EOS Frontend Server

- **Eight** additional servers as EOS FST: mount CephFS with the **CentOS 8.2 kernel**
- Each FST has a **separate data directory** in CephFS, configured as **8 EOS filesystems**
- Filesystems can be **moved in production** from node to node in case of node failures

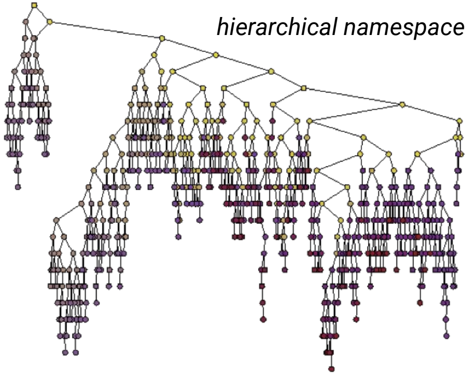
type	name	groupsize	groupmod	N(fs)	N(fs-rw)	sum(usedbytes)
spaceview	ceph	10	10	8	8	2.96 PB

host	port	id	uuid	path	schedgroup	headroom	boot	configstatus	drain	active	scaninterval	health
st-120hd-100gb009	1095	479	data-09	/o2/eos/data-09/	ceph.0	0.00	booted	rw	nodrain	**online**	604800	OK
st-120hd-100gb010	1095	480	data-10	/o2/eos/data-10/	ceph.0	0.00	booted	rw	nodrain	**online**	604800	OK
...	...	...	...	...	...	...	...	...	...	**...**	...	...
st-120hd-100gb016	1095	486	data-16	/o2/eos/data-16/	ceph.0	0.00	booted	rw	nodrain	**online**	604800	OK

R&D Setup - NAMESPACES



MGM provides user visible frontend namespace **/eos/**



hierarchical namespace

MDS provides transparent backend namespace **/cephfs/**
- shared on EOS FSTs as inode object storage for EOS data

pseudo flat namespace