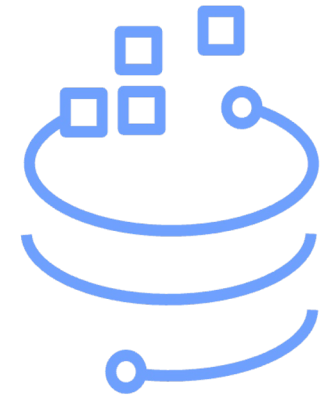


IBM Storage Scale Performance Update 2026 London User Group Meeting



Storage Scale

John Lewars, Storage Scale Performance Development Lead

Contributions from others on the IBM Storage Scale Team:

John Divirgilio, Rezaul Islam, Mara Miranda, Mayank Pandey,
Rogelio Rivera, and Olaf Weiser

io500 Update:

Preview of Upcoming Performance Improvements

IBM's IO500 submission from SC24 vs. a preview of upcoming (4Q 2026) code changes

Benchmark Storage Scale 5.1.9.0	IBM Ehningen (Single EPYC 9274F 24-Core 10 clients, 16 MB Blocksize, RoCE) – SSS6000 1 BB –SC24 Submitted	IBM Ehningen (Single EPYC 9274F 24-Core 10 clients, 16 MB Blocksize, RoCE) – SSS6000 1 BB – ISC26 code run done recently on same system	Percent Difference
mdtest-easy-write	322.9	403.7	25.02
mdtest-hard-write	27.6	26.7	-3.26
find	12203.1	6097.4	-50.03
mdtest-easy-stat	961.9	997.8	3.73
mdtest-hard-stat	327.2	490.5	49.91
mdtest-easy-delete	133.7	291.3	117.88
mdtest-hard-read	305.7	342	11.87
mdtest-hard-delete	36.1	30.5	-15.51
ior-easy-write	148	235	58.78
ior-hard-write	44.2	55.3	25.11
ior-easy-read	310.4	312	0.52
ior-hard-read	130.2	152.9	17.43
Bandwidth GiB/s	127.5	157.8	23.77
KIOPS	290.3	315.2	8.59
TOTAL Score	192.4	223.0	15.93

This comparison highlights code improvements using the exact same hardware, made between Nov. 2024 and May 2026. The May result uses a development build that adds code changes we intend to deliver in 4Q 2026. The comparison shows that IBM has improved the performance of multiple different workload types.

The regression to the **find** benchmark score does not appear to be a GPFS issue. Instead, changes to the io500 benchmark have been made. To improve these numbers:

1. There's an update to the ISC26 version (io500-isc26_v2) IBM will try.
2. We plan to rework the external version of the find benchmark we previously used so it's compatible with the later version of the suite.

The **mdtest-hard-delete** benchmark score example (30.5) is lower than other results we've seen on the latest code. For the actual io500 result submitted for ISC26 (similar score of 223.6) this benchmark achieved a score of 36.6: <https://io500.org/submissions/view/809>

Data Acceleration Tier Update

The AI Data Performance Bottleneck

The **Data Acceleration Tier (DAT)** offering provides significant performance improvements for user data access.

Small-medium read performance:

Centralized DAT (aka/NVMeoF DAT) offers a 15X improvement to 4 KiB random read on the SSS 6000, providing 28M random read IOPS at low latency, helping prevent application stalls.

Decentralized DAT (aka/DAT-local) offers performance improvements to 4 KiB random read that scale with the number of clients, according to their CPU and disks, providing scalable performance gains.

Small-medium write performance:

Both Centralized and Decentralized DAT support a burst-buffer write mode, where data is initially written to a performance tier (including local storage for DAT-local) to avoid erasure-code overheads.

While data remains in burst buffer mode, it is not fully protected against hardware failures. Policies can be used to replicate data to a protected, erasure-coded tier as part of an explicit durability workflow.

This mode preserves file-system-level consistency semantics, ensuring a coherent global view of data across nodes, as opposed to other vendors “eventually consistent” models that relax immediate visibility guarantees.

Decentralized DAT (DAT-local) deployment model allows data to be placed close to a workload, thereby reducing network overhead and freeing network resources for other workloads.



Data Acceleration Tier (DAT)

What is it?

Accelerates AI workloads by allowing user data in a filesystem to be stored in up to two pools concurrently. Possible combinations of pools:

1. a **GNR (GPFS Native RAID)** copy (replica) stored in a Storage Scale RAID- encoded pool

AND

2. a **performance** copy (replica) stored:

(a) on storage (typically an NVMe drive) physically-local to the client (decentralized DAT deployment model)

-or-

(b) on an NVMeoF-exported drive (centralized DAT deployment model)

DAT provides high IOPs without compromising reliability.

NVIDIA DGX SuperPOD with IBM ESS/ISS 



* DAT has been referred to in past presentations by its IBM internal name 'Ustore'.

Specifications

GPU	8x NVIDIA H100 Tensor Core GPUs
GPU memory	640GB total
Performance	32 petaFLOPS FP8
NVIDIA® NVSwitch™	4x
System power usage	10.2kW max
CPU	Dual Intel® Xeon® Platinum 8480C Processors 112 Cores total, 2.00 GHz (Base), 3.80 GHz (Max Boost)
System memory	2TB
Networking	4x OSFP ports serving 8x single-port NVIDIA ConnectX-7 VPI ➤ Up to 400Gb/s InfiniBand/Ethernet 2x dual-port QSFP112 NVIDIA ConnectX-7 VPI ➤ Up to 400Gb/s InfiniBand/Ethernet
Management network	10Gb/s onboard NIC with RJ45 100Gb/s Ethernet NIC Host baseboard management controller (BMC) with RJ45
Storage	OS: 2x 1.92TB NVMe M.2
Internal storage:	8x 3.84TB NVMe U.2

DAT Deployment Models

DAT-Local Model – Decentralized DAT

DAT-Local leverages storage (typically NVMe drives but can be any supported storage) on client nodes, maximizing IOPS and scaling linearly with added clients.

DAT-NVMeoF Model – Centralized DAT

DAT-NVMeoF centralizes storage via NVMe over Fabrics, offering near-local read speeds without physical NVMe drives on each node, ensuring efficient utilization of all NVMe drives by providing a central access method for all workloads.



Centralized DAT Performance

Exceptional Read Performance

NVMeoF-exported drives from ESS 6000 servers deliver up to 28 million IOPS, significantly outperforming traditional filesystems.

Low-Latency Network Fabric

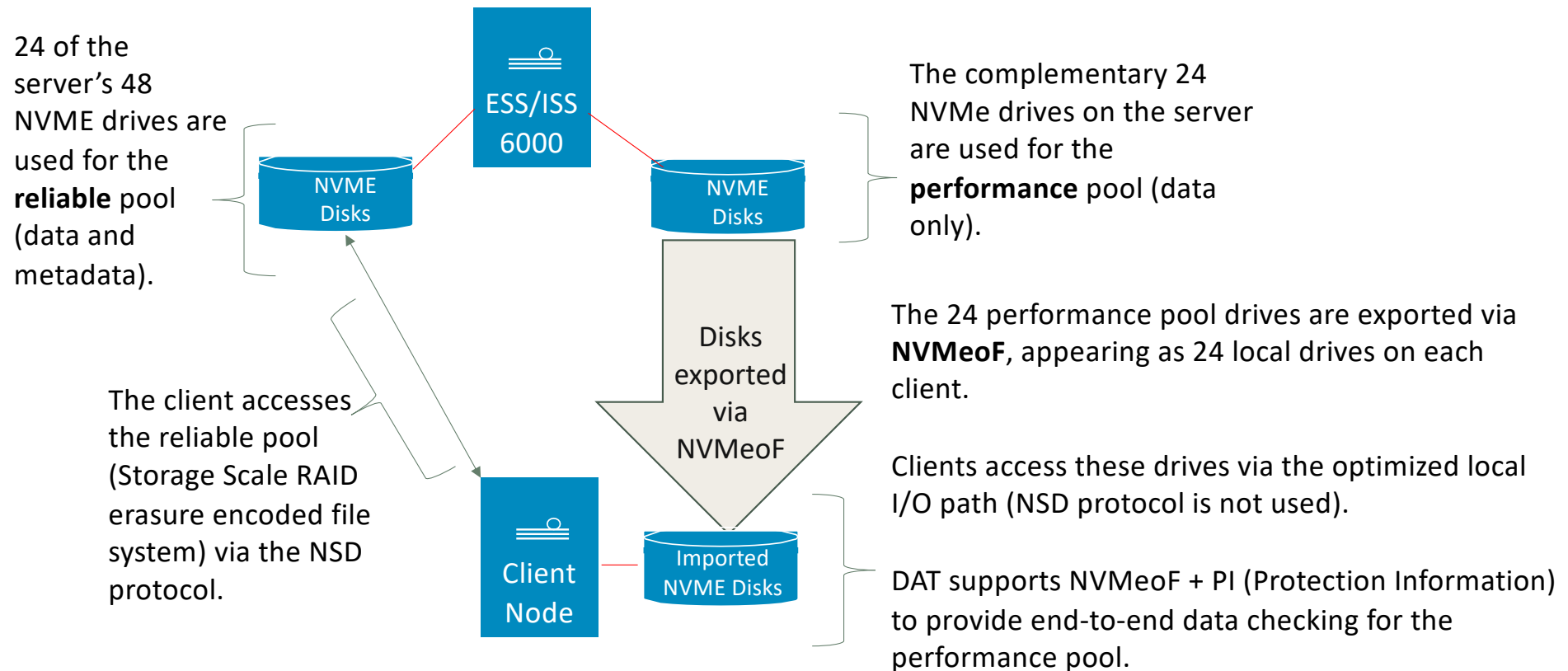
Offloading reads over RDMA-capable fabrics reduces latency, uses server CPU resources more efficiently, and enables efficient data access over the NVMeoF protocol

Optimized I/O Path

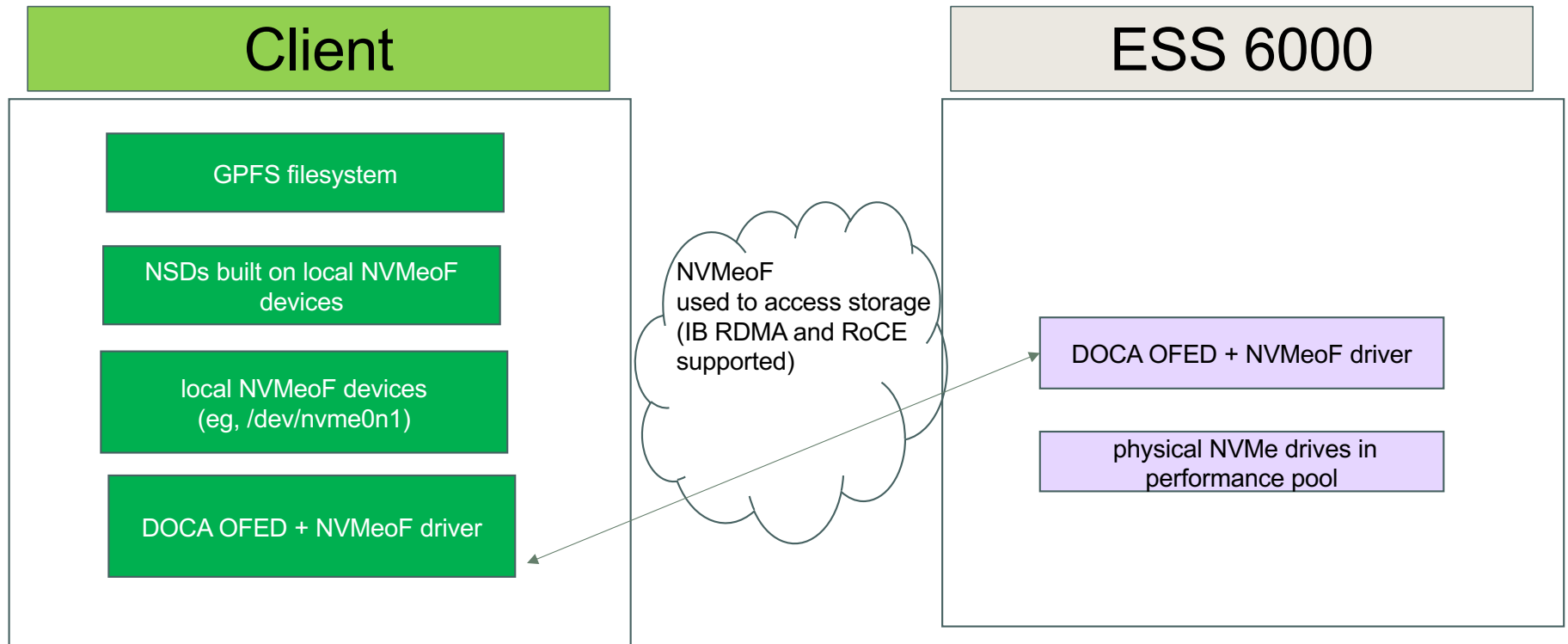
NVMeoF block devices bypass filesystem overheads present for small to medium size I/O's, maximizing throughput for AI workloads with random access.



Centralized DAT Deployment

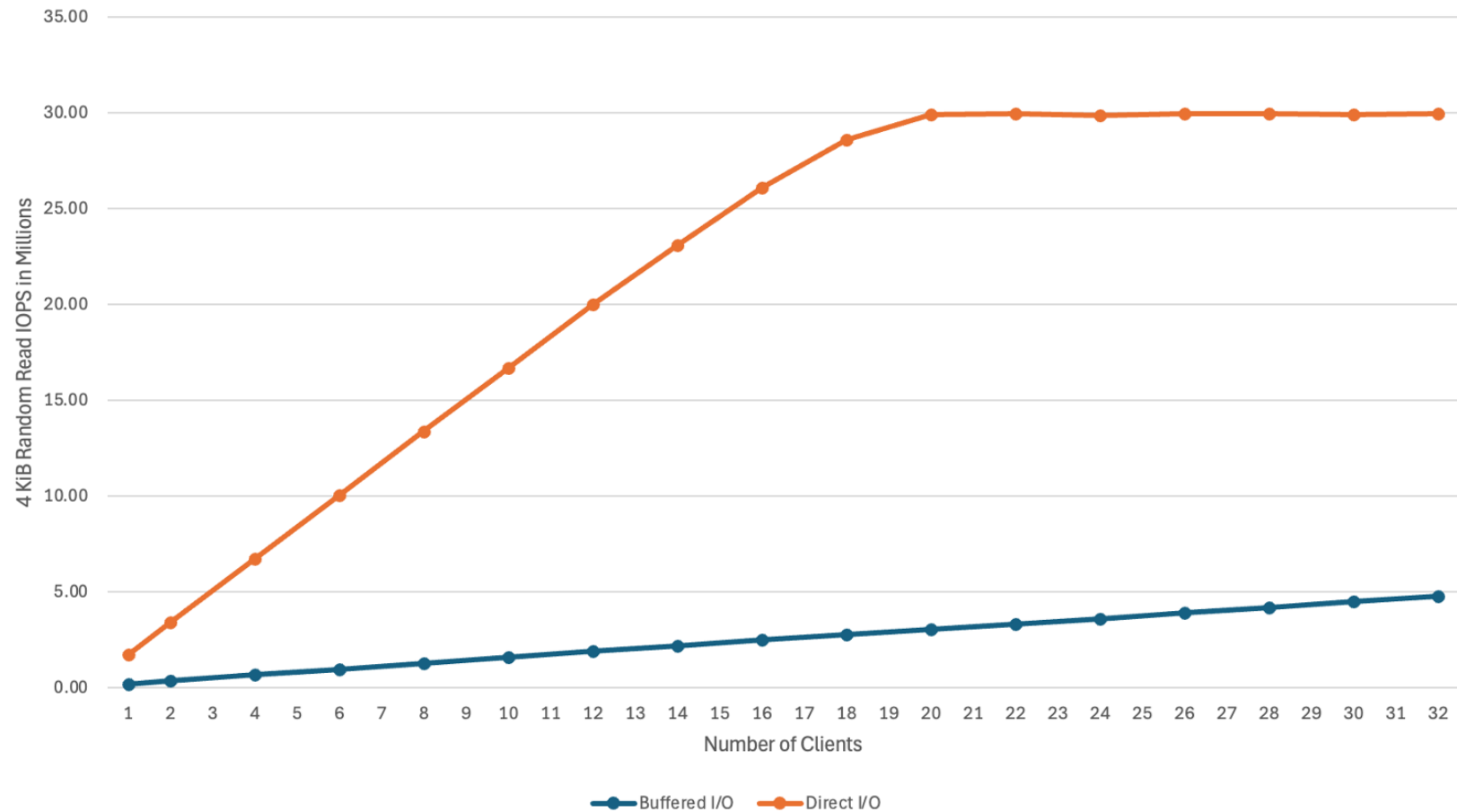


Centralized DAT Performance Pool Block Diagram



Centralized DAT Performance

4 KiB DIO Random Read with Centralized DAT on SSS 6000
NVMeoF **offload is Enabled and PI is Disabled (PI enabled is the default)**
(average of 5 three-minute runs – each thread reads an 8 GB data file)
`ior -a POSIX -F -z -b 8g -t 4k -o <dir> -e -r -v -i 1 -k --posix.direct`



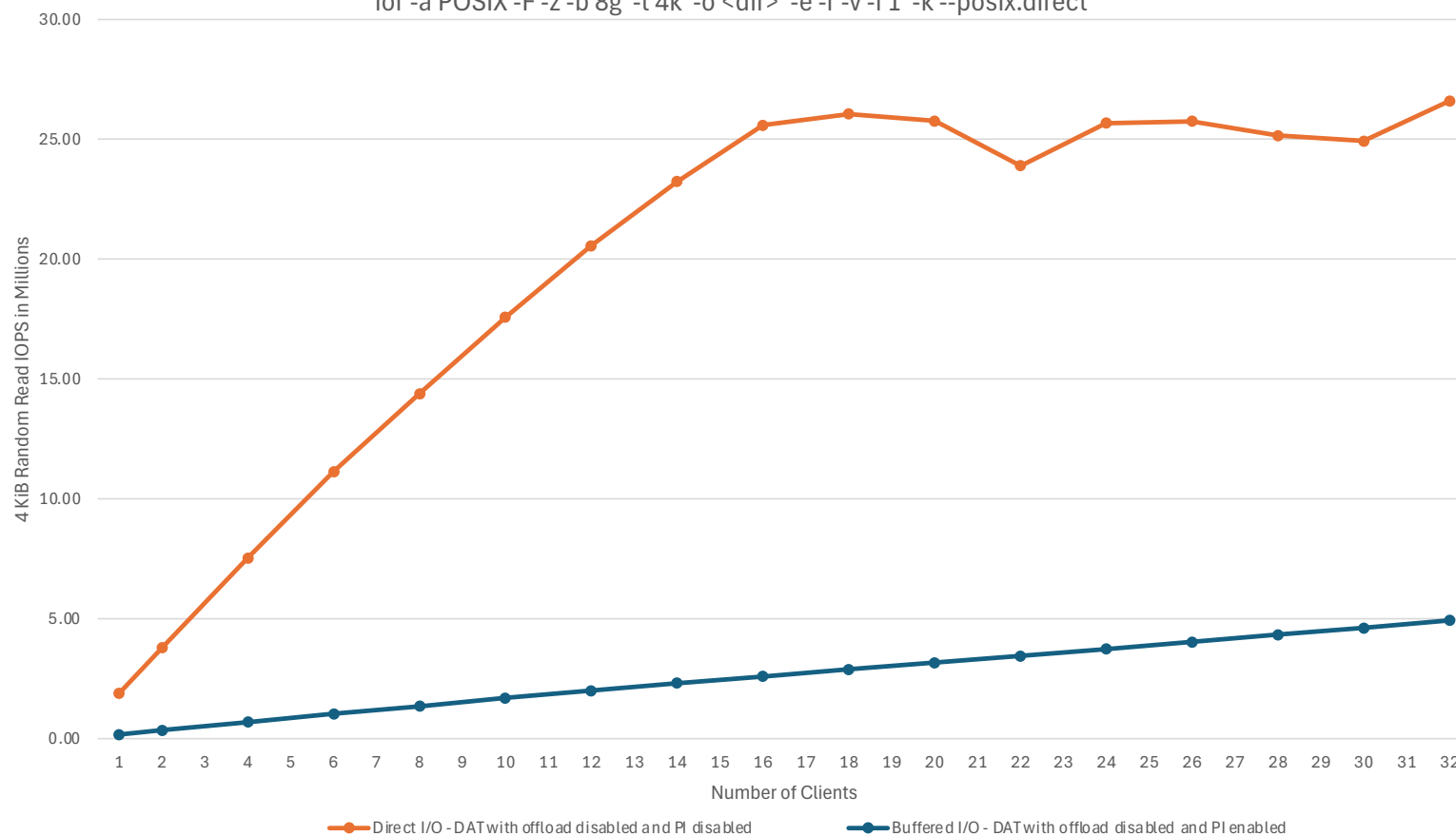
Centralized DAT Performance (cont.)

4 KiB DIO Random Read with Centralized DAT on SSS 6000

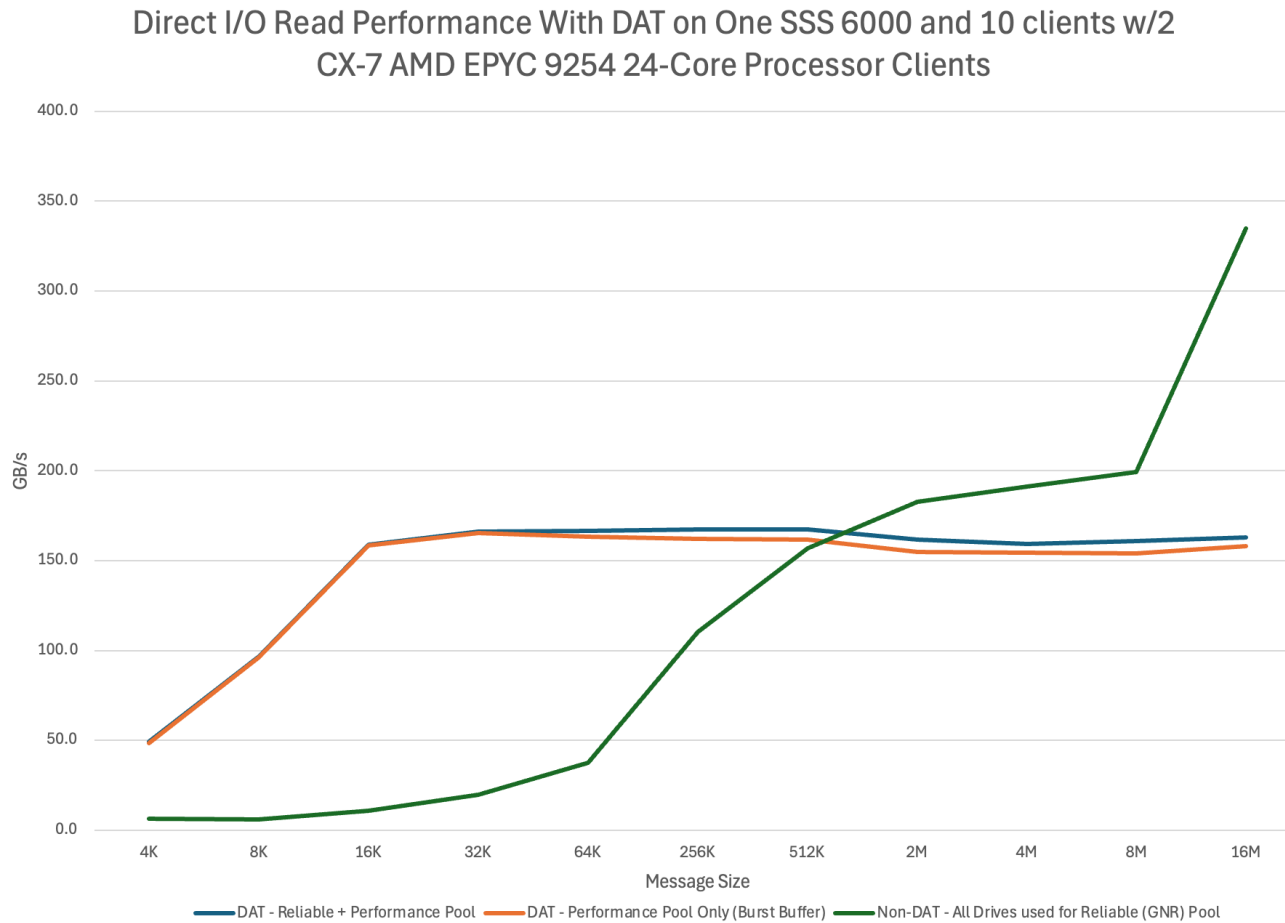
NVMeoF **offload is Disabled** and **PI is enabled**

(average of 5 three-minute runs - each task has its own 8 GB data file)

`ior -a POSIX -F -z -b 8g -t 4k -o <dir> -e -r -v -i 1 -k --posix.direct`

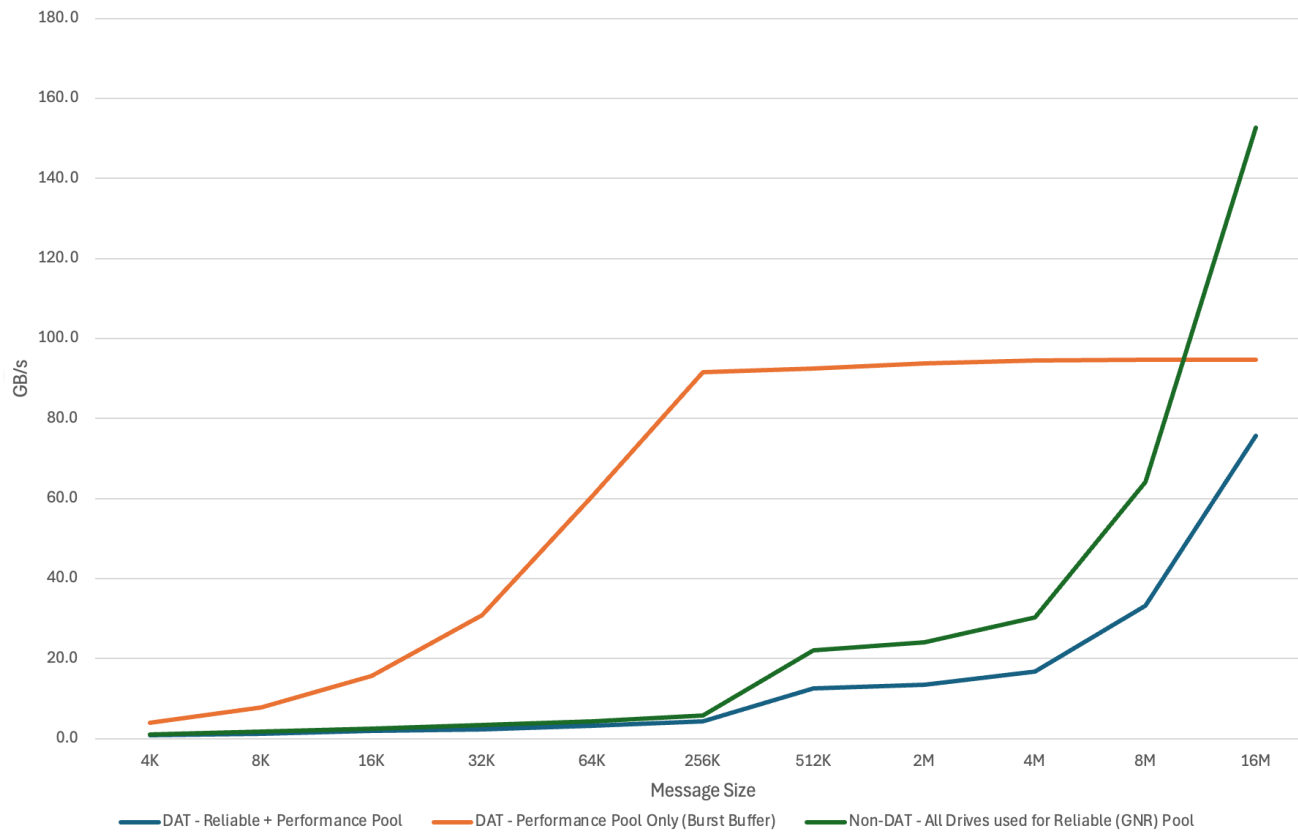


Centralized DAT Performance cont.

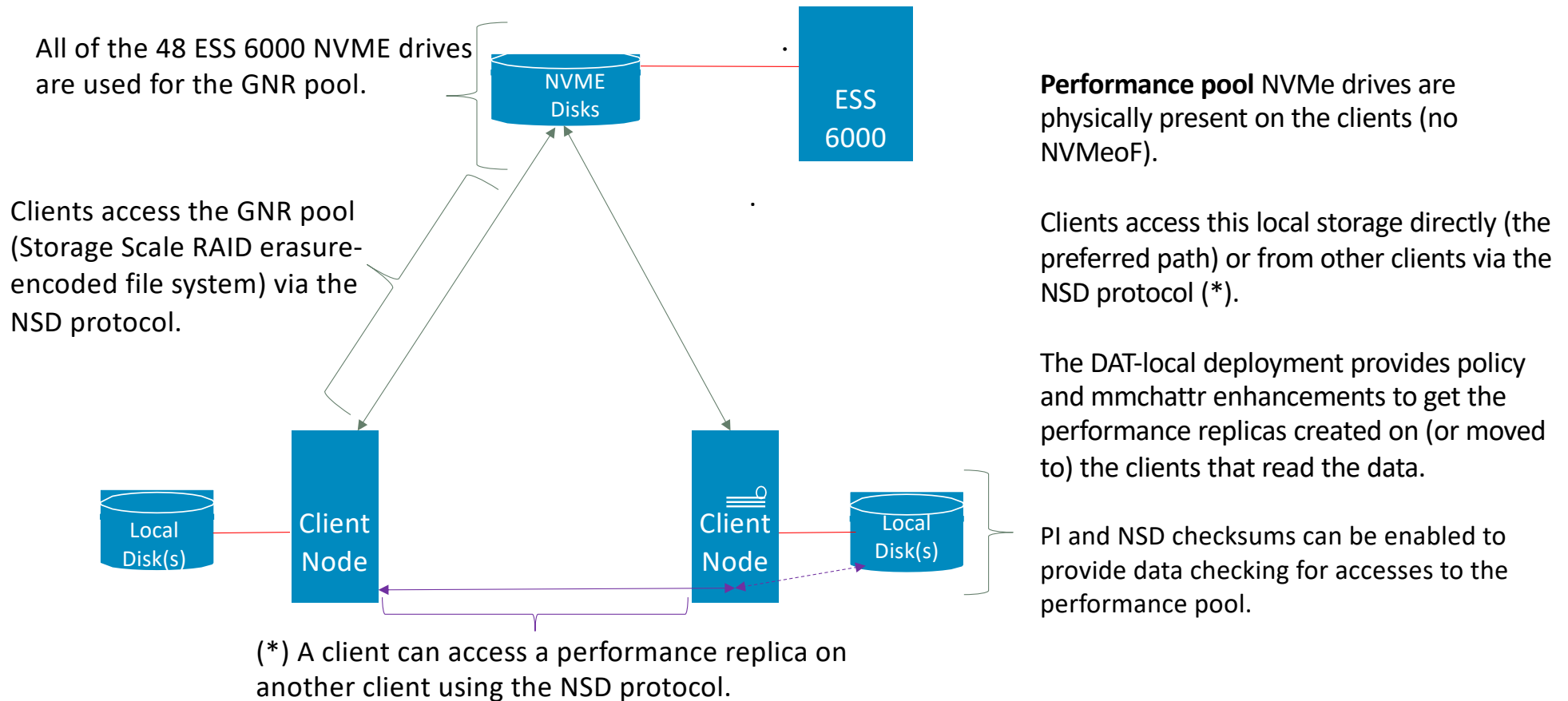


Centralized DAT Performance cont.

Direct I/O Write Performance With DAT on One SSS 6000 and 10 clients w/2
CX-7 AMD EPYC 9254 24-Core Processor Clients



Decentralized DAT Deployment



Decentralized DAT Scaling Performance

Linear Performance Scaling

Performance scales linearly when each node reads from locally-attached performance pool NVMe devices, reducing network latency. The clients' performance characteristics (mainly local disk and CPU performance) will dictate performance

Network Bottleneck Elimination

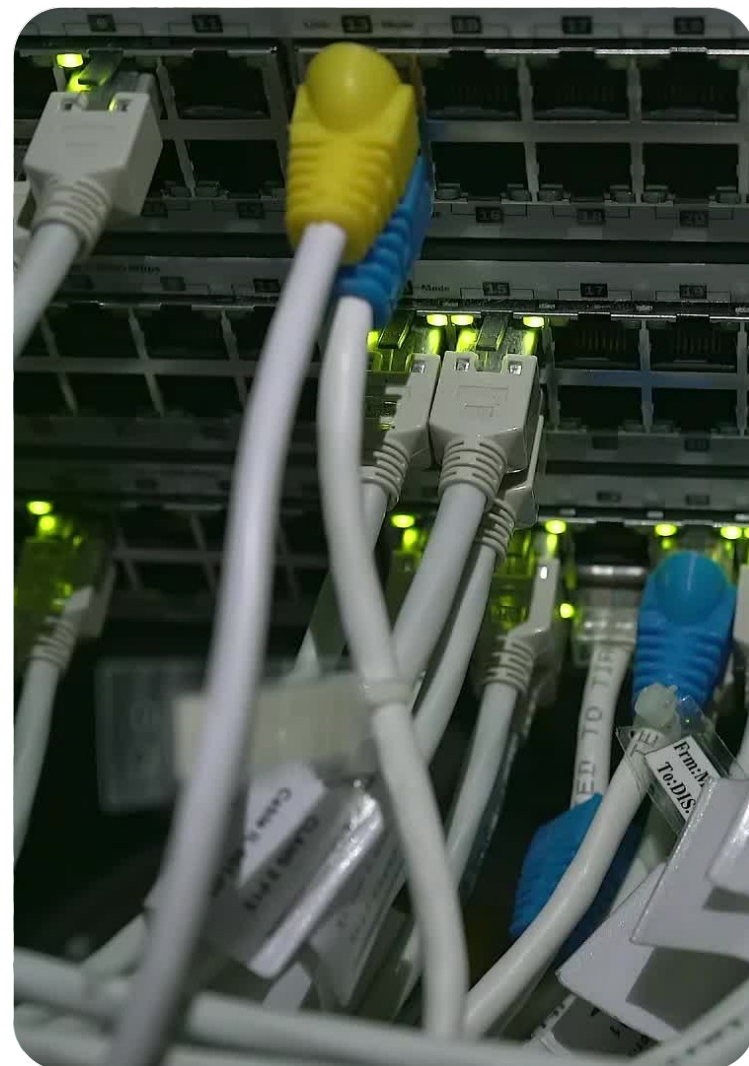
Eliminating network traversal for most reads increases GPU utilization and data pipeline efficiency.

Lab Measurement Results

IBM lab results show between **1.0 - 1.4 million** 4KiB random read IOPs per client on systems when reading local data

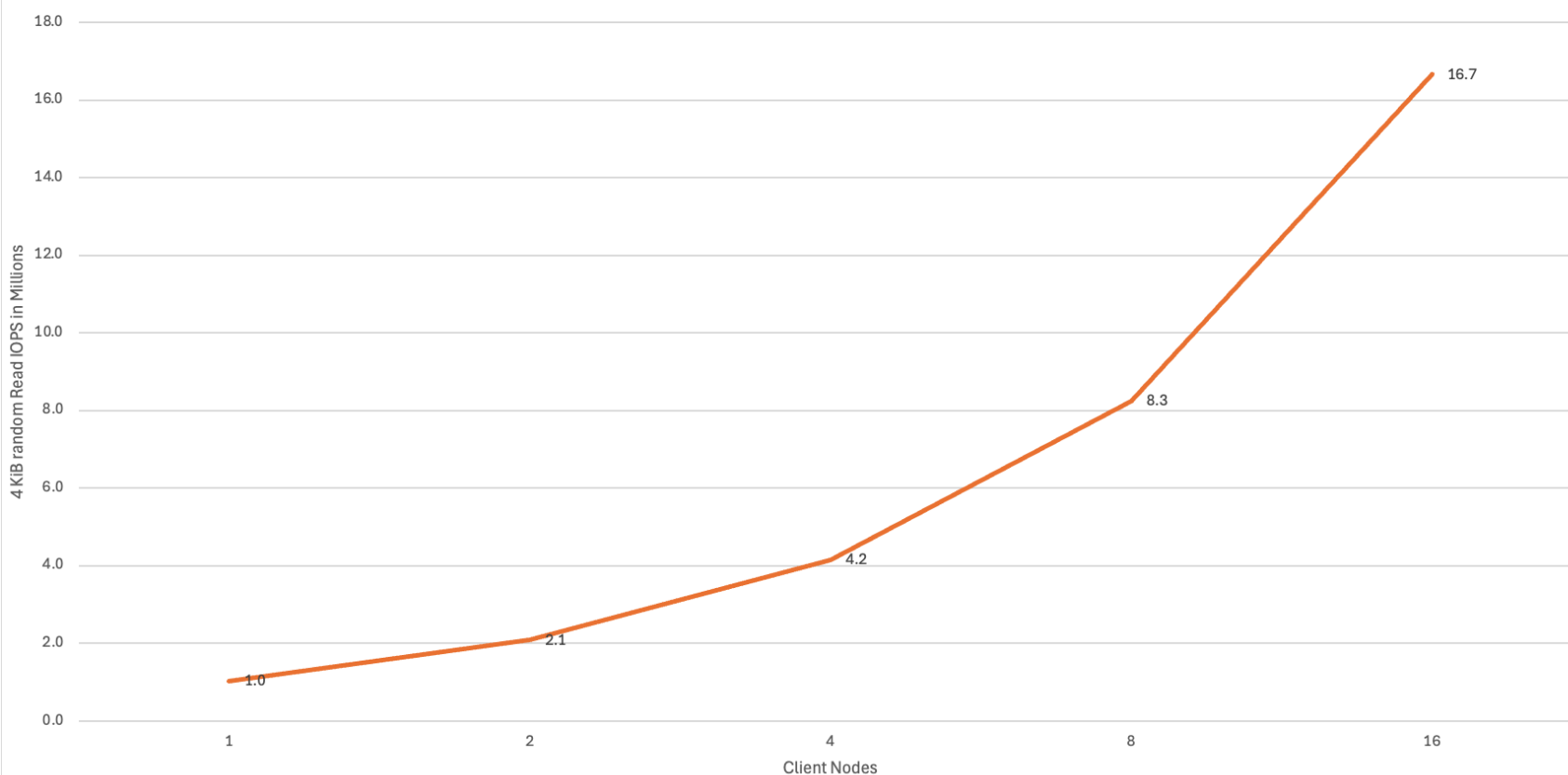
Test configuration for Scaling runs:

Clients: 10, each client has 48 AMD cores and 2 NVMe drives
ESS 6000 (reliable pool)



4 KiB Random Read IOPS Performance with Decentralized DAT (Local Drives in Client Nodes)

IOPS Ehningen bb8 ISS 6000 + 2 NVME drives per client - 4 KiB Random Read IOPS (**Millions**)
Local Drive Variant of DAT - 32 clients qdepth=256 njobs=32 - Scaling Up Number of Clients



IBM io500 Submission on the Saguaro System

<https://io500.org/submissions/view/784>

Node Configurations

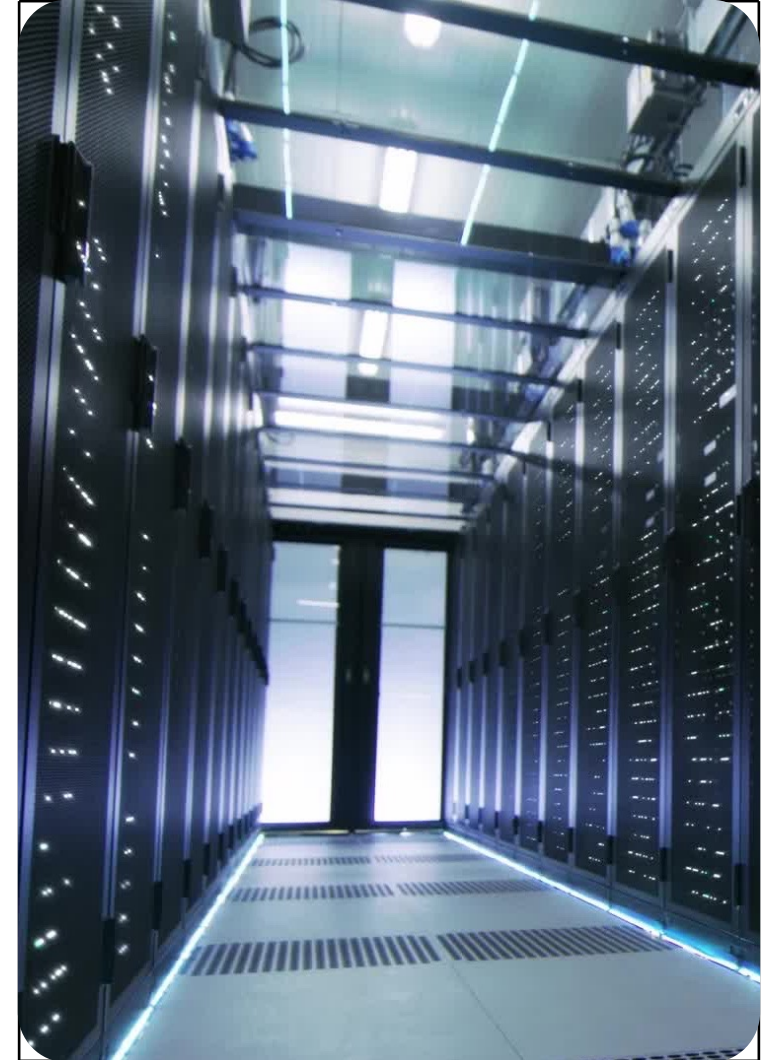
IBM had been using the Saguaro system for io500 runs but we had not been exploiting the NVMe drives in the clients

Clients NVMe Drives Used for ior-easy files

In our latest (SC25) io500 runs we enabled the decentralized DAT feature to exploit the existing NVMe drives in the client nodes for ior-easy files only. These slides describe the policy used and the performance results

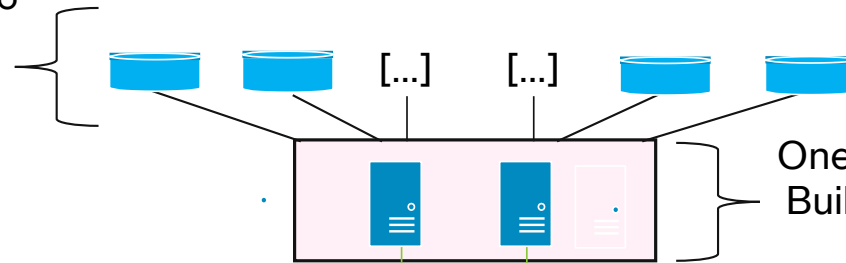
Network Usage

More efficient use of network resources is achieved by using a policy to, in burst buffer mode, direct the ior-easy files created by io500 to the local NVMe drives on the Saguaro client nodes, writing the data to the same location it will be consumed.



Saguaro System Storage Details

IBM Storage Scale RAID Pool (46 twin-tailed NVMe Drives, connected to Storage Scale System 6000 servers - 2 failed drives)

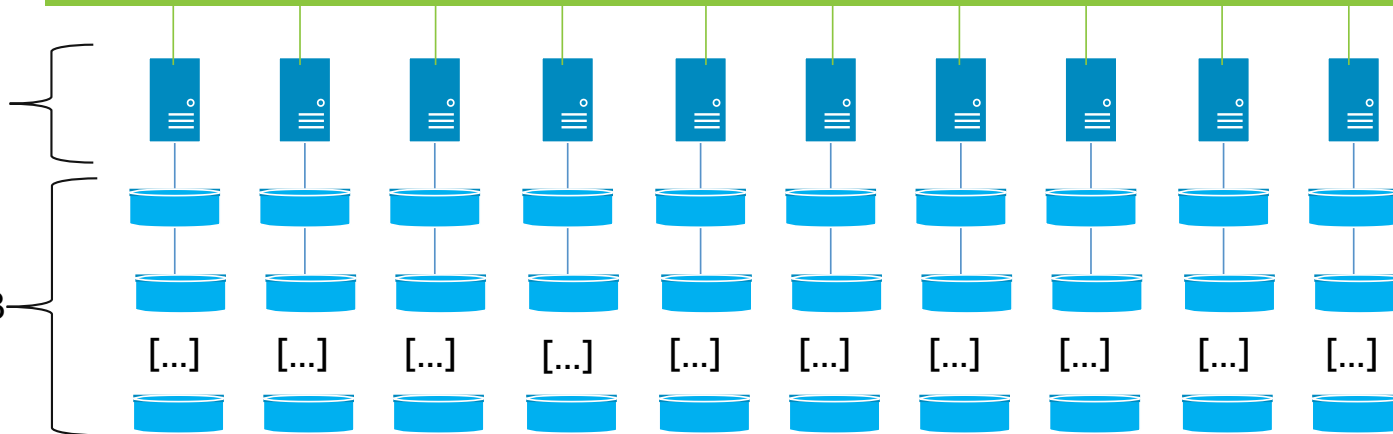


One Storage Scale System 6000 Building Block has two canisters (server nodes)

NDR400 InfiniBand Network with RDMA Enabled

Ten Saguaro Client Nodes

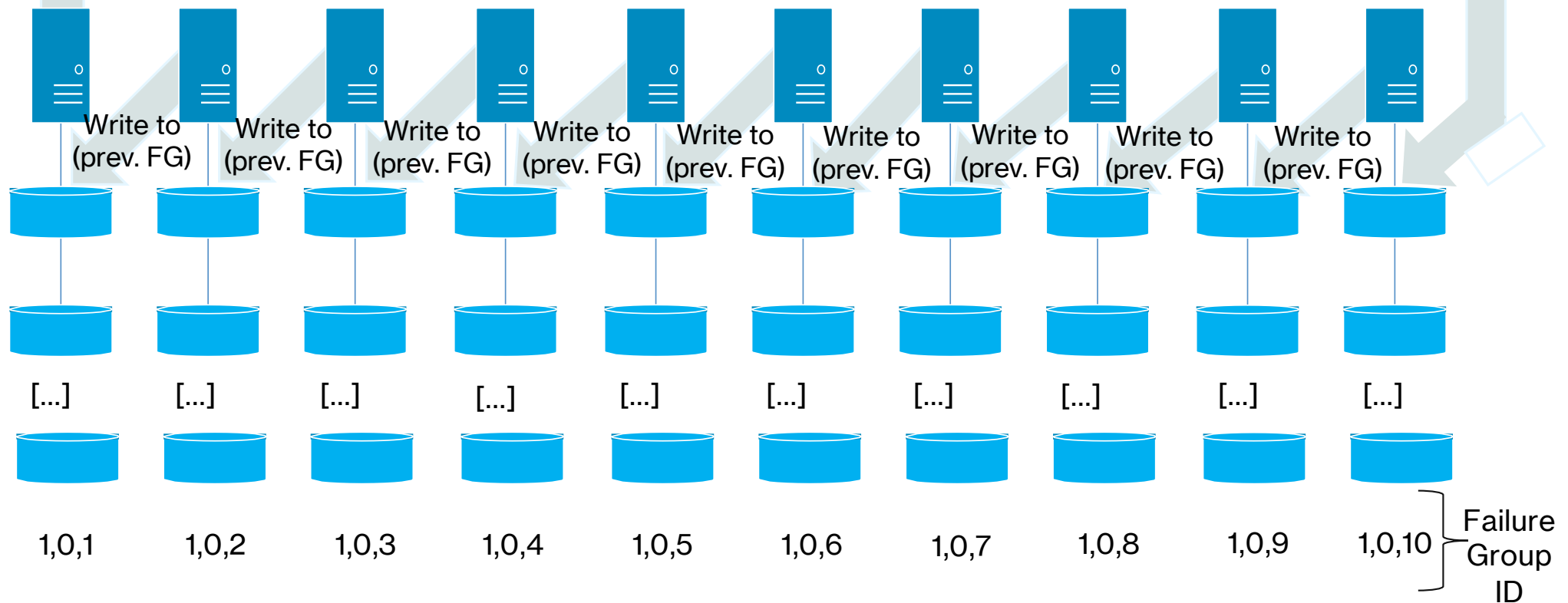
Decentralized DAT Pool (uses 7 of the 8 client NVMe drives available in each client)



Saguaro System uses Policy to Facilitate Local Disk Reads for ior-easy Files

Each client writes to decentralized DAT pool disks that are local to the node where the data will later be read

Node with FG ID = (1,0,1) writes to disks attached to the node with Failure Group ID (1,0,10)



Example Policy from Saguaro SC25 io500 Submission

Below is an example of a policy used to place ior-easy file for the io500 benchmark to allow reads of the file to be to local disk:

```
RULE 'ior_file_easy.0-24' SET POOL 'system' REPLICATE(0:1) WHERE (name >= 'ior_file_easy.00000000' AND name <= 'ior_file_easy.00000024') AND (setWADFGP(1,0,10))
```

```
RULE 'ior_file_easy.25-49' SET POOL 'system' REPLICATE(0:1) WHERE (name >= 'ior_file_easy.00000025' AND name <= 'ior_file_easy.00000049') AND (setWADFGP(1,0,1))
```

```
RULE 'ior_file_easy.50-74' SET POOL 'system' REPLICATE(0:1) WHERE (name >= 'ior_file_easy.00000050' AND name <= 'ior_file_easy.00000074') AND (setWADFGP(1,0,2))
```

```
RULE 'ior_file_easy.75-99' SET POOL 'system' REPLICATE(0:1) WHERE (name >= 'ior_file_easy.00000075' AND name <= 'ior_file_easy.00000099') AND (setWADFGP(1,0,3))
```

```
RULE 'ior_file_easy.100-124' SET POOL 'system' REPLICATE(0:1) WHERE (name >= 'ior_file_easy.00000100' AND name <= 'ior_file_easy.00000124') AND (setWADFGP(1,0,4))
```

```
RULE 'ior_file_easy.125-149' SET POOL 'system' REPLICATE(0:1) WHERE (name >= 'ior_file_easy.00000125' AND name <= 'ior_file_easy.00000149') AND (setWADFGP(1,0,5))
```

```
RULE 'ior_file_easy.150-174' SET POOL 'system' REPLICATE(0:1) WHERE (name >= 'ior_file_easy.00000150' AND name <= 'ior_file_easy.00000174') AND (setWADFGP(1,0,6))
```

```
RULE 'ior_file_easy.175-199' SET POOL 'system' REPLICATE(0:1) WHERE (name >= 'ior_file_easy.00000175' AND name <= 'ior_file_easy.00000199') AND (setWADFGP(1,0,7))
```

```
RULE 'ior_file_easy.200-224' SET POOL 'system' REPLICATE(0:1) WHERE (name >= 'ior_file_easy.00000200' AND name <= 'ior_file_easy.00000224') AND (setWADFGP(1,0,8))
```

```
RULE 'ior_file_easy.225-249' SET POOL 'system' REPLICATE(0:1) WHERE (name >= 'ior_file_easy.00000225' AND name <= 'ior_file_easy.00000249') AND (setWADFGP(1,0,9))
```

```
RULE 'default' SET POOL 'system'
```

SC25 io500 Submission 784 (Saguaro) Results

	Saguaro SC24	Saguaro SC25	
	IBM Storage Scale 5.2.2-0 (early pre-GA build) with ESS 6.2.1-0	IBM Storage Scale 6.0.0.0 with ESS 7.0.0.0 (early pre- GA build)	
Benchmark/Metric			Percent Improvement for SC25
ior-easy-write	151.1	238.7	58
mdtest-easy-write	191.2	223.6	17
ior-hard-write	28.3	39.7	41
mdtest-hard-write	24.4	23.9	-2
find	7516.7	4803.1	-36 *
ior-easy-read	284.2	422.2	49
mdtest-easy-stat	571.3	546.6	-4
ior-hard-read	107.5	142.4	32
mdtest-hard-stat	272.7	292.6	7
mdtest-easy-delete	165.3	218.9	32
mdtest-hard-read	321.6	365.3	14
mdtest-hard-delete	29.4	28.0	-5
ior-rnd4K-easy-read	N/A	13.0	N/A
Bandwidth (GiB/s)	106.9	154.5	45
IOPS (kiops)	232.5	234.6	1
Total	157.7	190.4	21

* The regression to find is seen with non-DAT runs and related to a benchmark issue and not the use of DAT

Migrating an Existing ESS 6000 Filesystem to a Decentralized DAT Filesystem

1. Install ($\geq$) Scale 6.0.0.1 and ESS 7.0.0.0 software and create a file system on the SSS 6000 using the standard deployment approach
2. Run “mmchconfig release=LATEST” to upgrade the daemon version. *
3. Run “mmchfs -V full” to upgrade the filesystem format version. *
4. Add the performance pool on client-local NVMe drives using “mmcrnsd” and “mmvdisk”

* If cluster is upgraded BEFORE creating file system, these steps aren't required



Current DAT Restrictions

Single Filesystem Limit

Each ESS 6000 supports only one DAT filesystem and that filesystem cannot span multiple ESS units (a single building block limit)

Performance Model Requirement

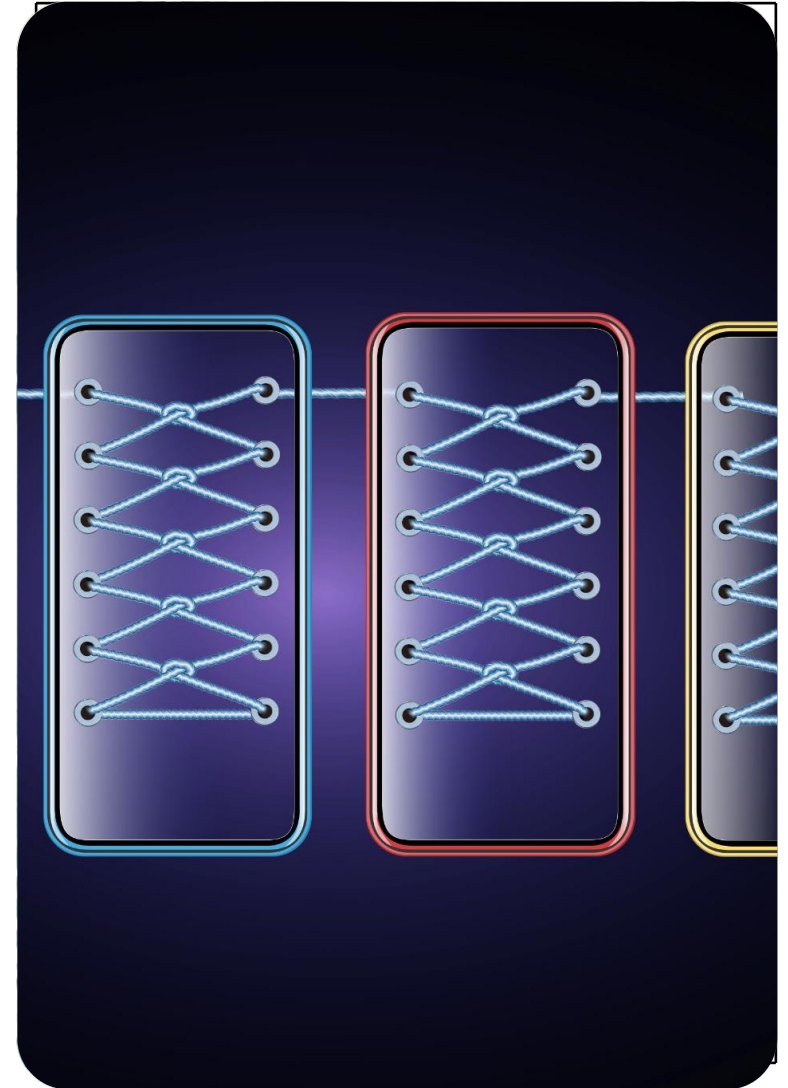
DAT requires fully configured ESS 6000 models with all NVMe drives populated to ensure consistent performance.

Unsupported Configurations

Hybrid SSD setups and remote storage clusters are not supported under current DAT restrictions.

Future Expansion Plans

Future releases aim to relax restrictions and enable multi-ESS scaling for larger clusters and deployments.



QLC Drive and JBOF in Scale System 6000



JBOF (5149-F26)



Rezaul Islam

IBM Storage Scale System Development

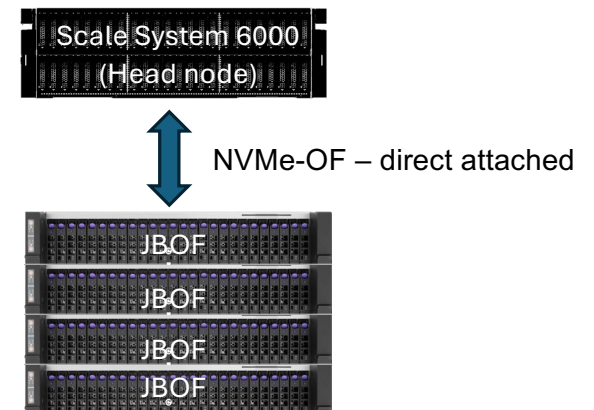
Extreme Deep Capacity with performance

- Deeper Flash Capacity by increasing number of QLC in Scale System 6000 head node – flexible mixed tier
- Provide Extreme deep Flash Capacity of a building block by JBOF attachment

5149-F26 2U26 JBOF

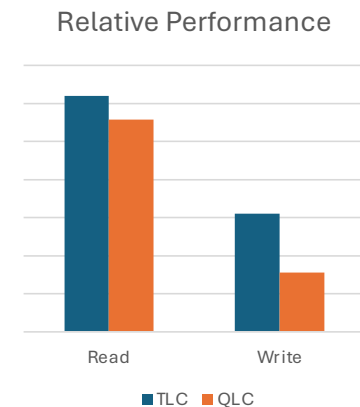
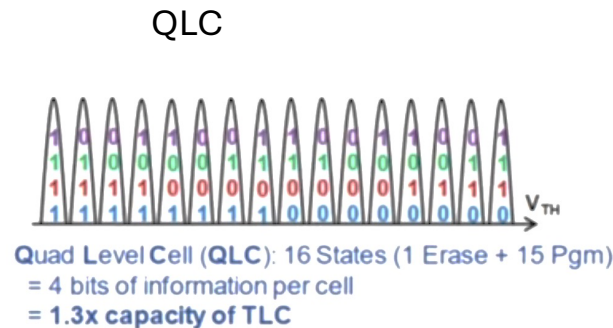
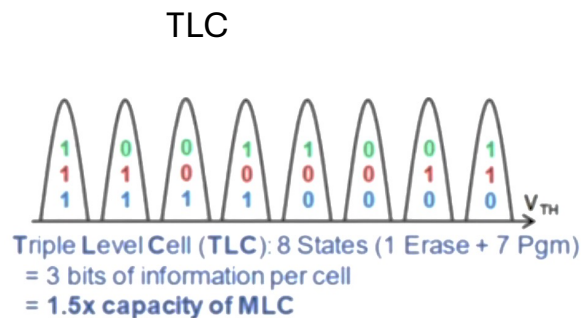
Shared Enclosure for Deep Flash Capacity

- Shared enclosure to attach additional QLC tiered Flash capacity to head controller
- Cost optimized Deep Flash capacity – performance of Flash and capacity of QLC drive
- High availability – no single point of failure with dual-ported QLC drives
- Security - SED support of the QLC drives using key servers and TPM

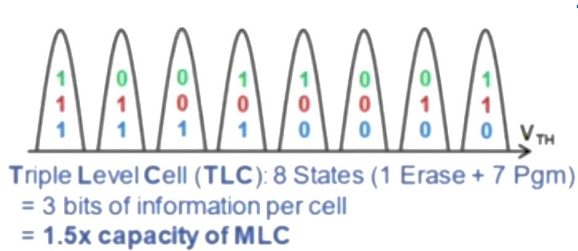


Deep Flash Capacity via QLC NVMe Drives

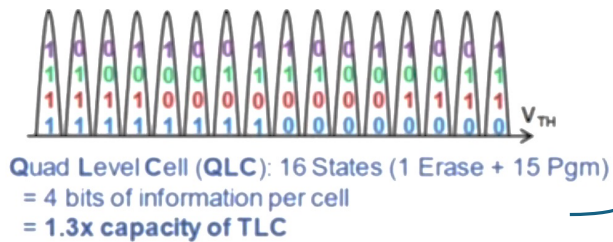
- QLC drive store 4 bits per cell vs 3 bits for TLC, leading to higher capacity and lower cost
- QLC drives typically have slower write speeds and lower endurance due to the increased density



TLC



QLC



- Combining best of the both worlds
- In ESS 7.0.0.0 (Dec. 14, 2025): Tiered Flash storage, TLC + QLC in same enclosure head unit:
 - 24 TLC + 24 QLC
 - 12 TLC + 36 QLC
 - 30 TB, 61 TB, and 122 TB QLC

QLC / TCL Performance in the SSS 6000 Head Unit

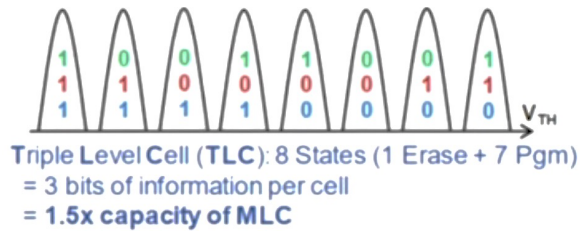
- For the following measurements, two pools – Tier1-NMVe and QLC were created with placementpolicy set.
- IOs can be run to either or both pools and, when IO are run to both pools, it can deliver close to **current** maximum performance of SSS6000

Tier-1 NVMe drives used here are PCIe-Gen5 drives, which has higher write performance

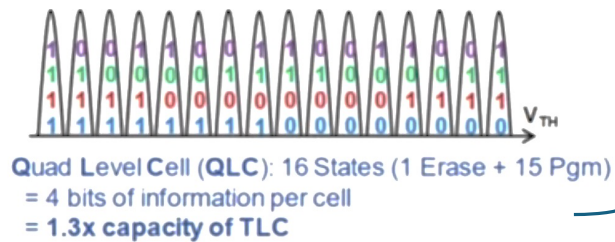
16 MB File System Block Size, 4 NDR-IB Adapters per Canister

Drives receiving IO		Write 8+2p MB/s	Read MB/s
Tier1-NVMe	QLC		
24	24	150000	300000
24	0	125000	160000
0	24	35000	160000

TLC

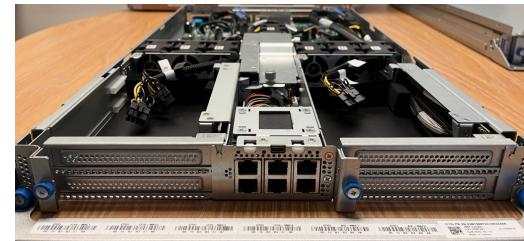
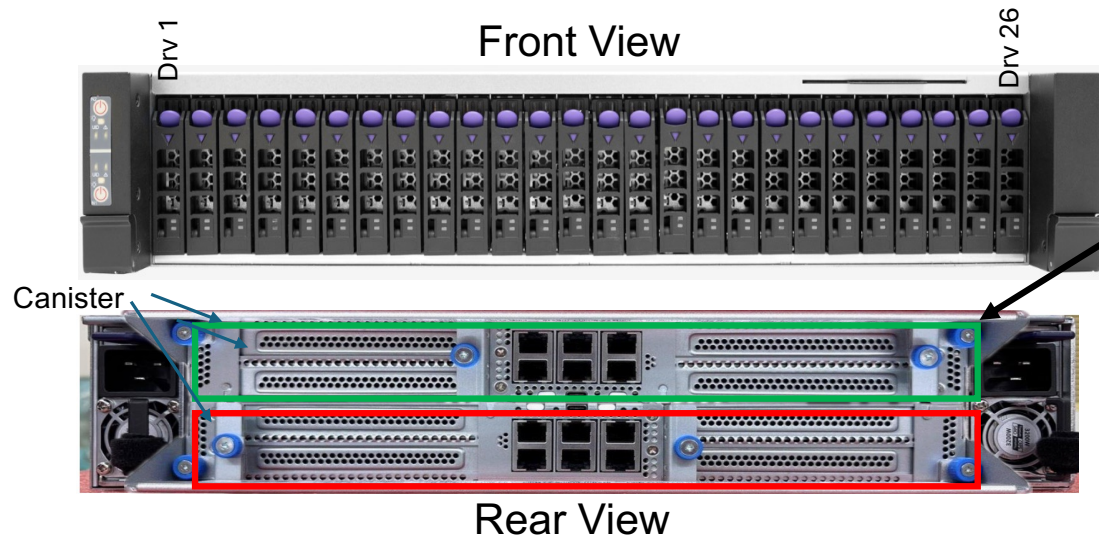


QLC

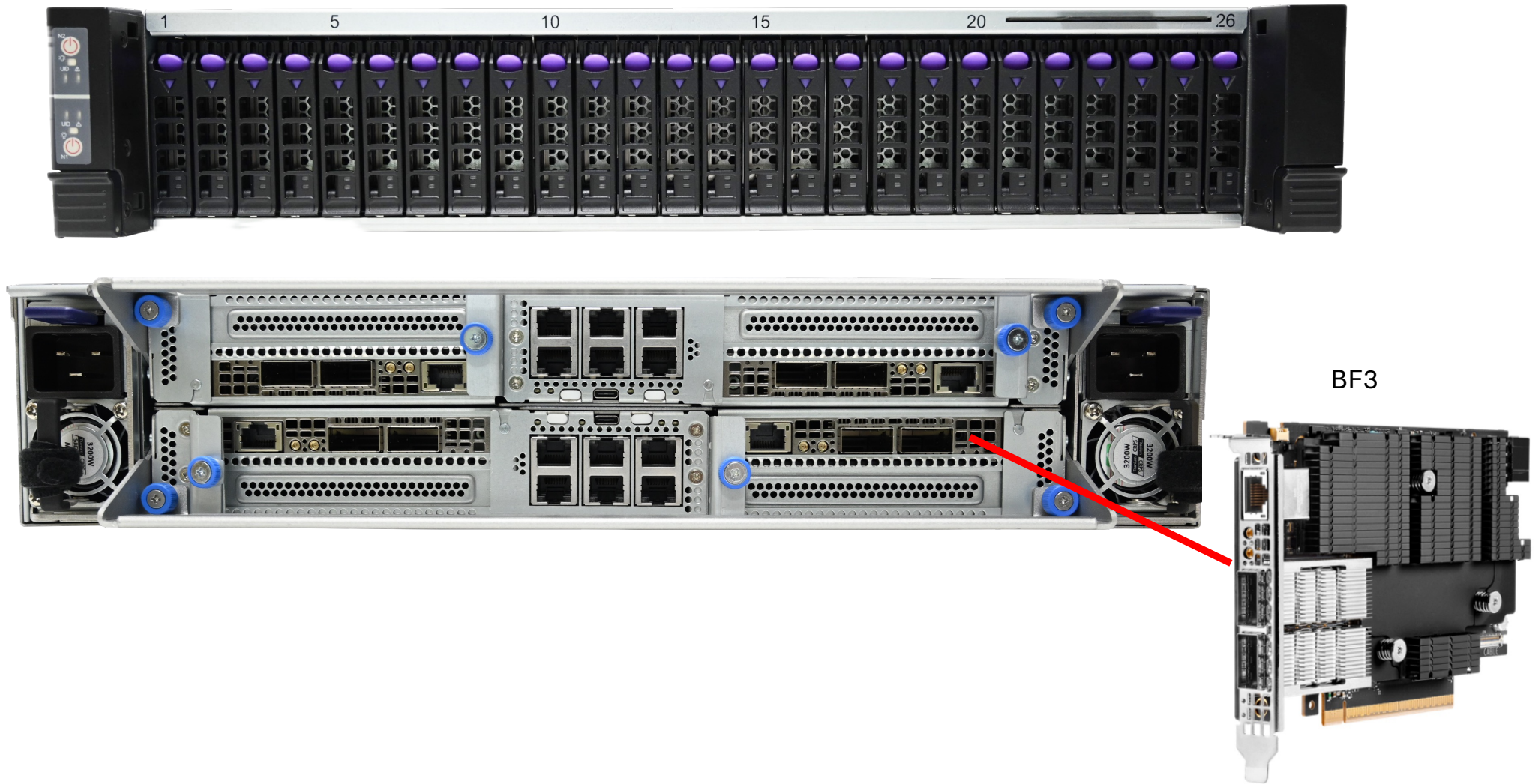


- Combining best of the both worlds
 - Tiered Flash storage
 - TLC + QLC in same enclosure
 - Extending QLC to JBOF for deep Flash Storage
 - 1 to 4 JBOF – up to 104 x 122 TB
 - Each JBOF supports 26 QLC and up to 4 JBOFs
- TLC drives in the JBOF in 7.0.1 is 4Q2026

2U26 JBOF Enclosure Details

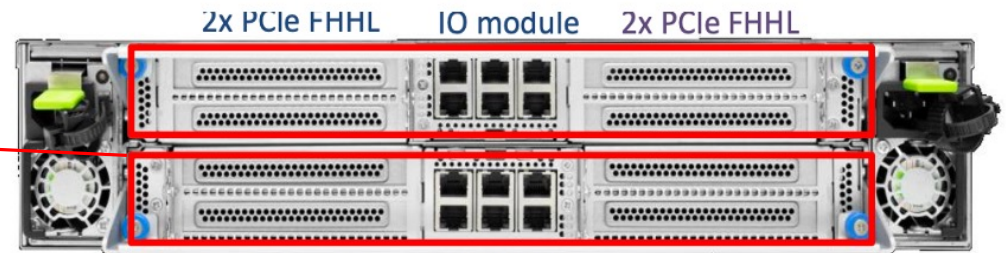
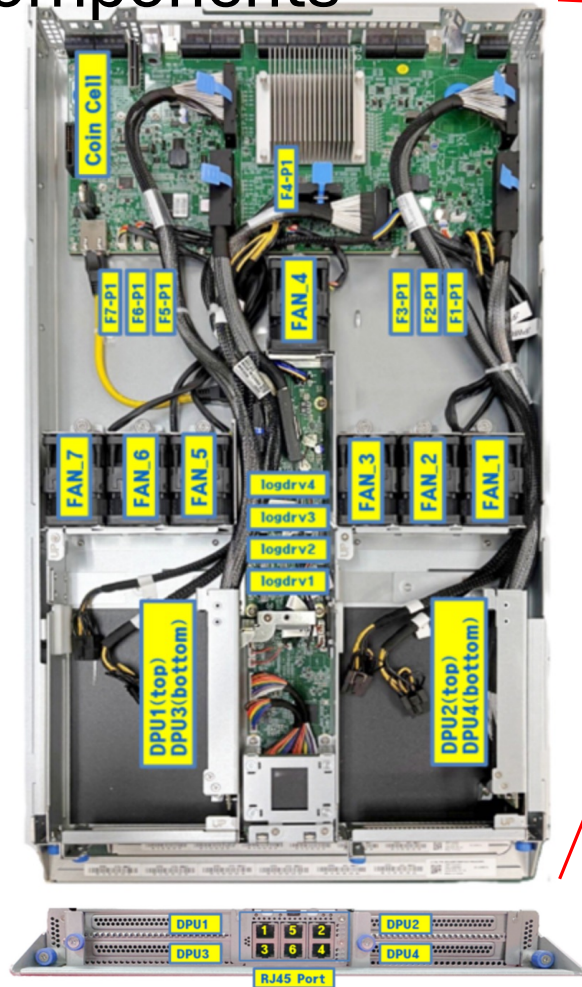


5149-F26 JBOF Enclosure FRU – Front and Rear View

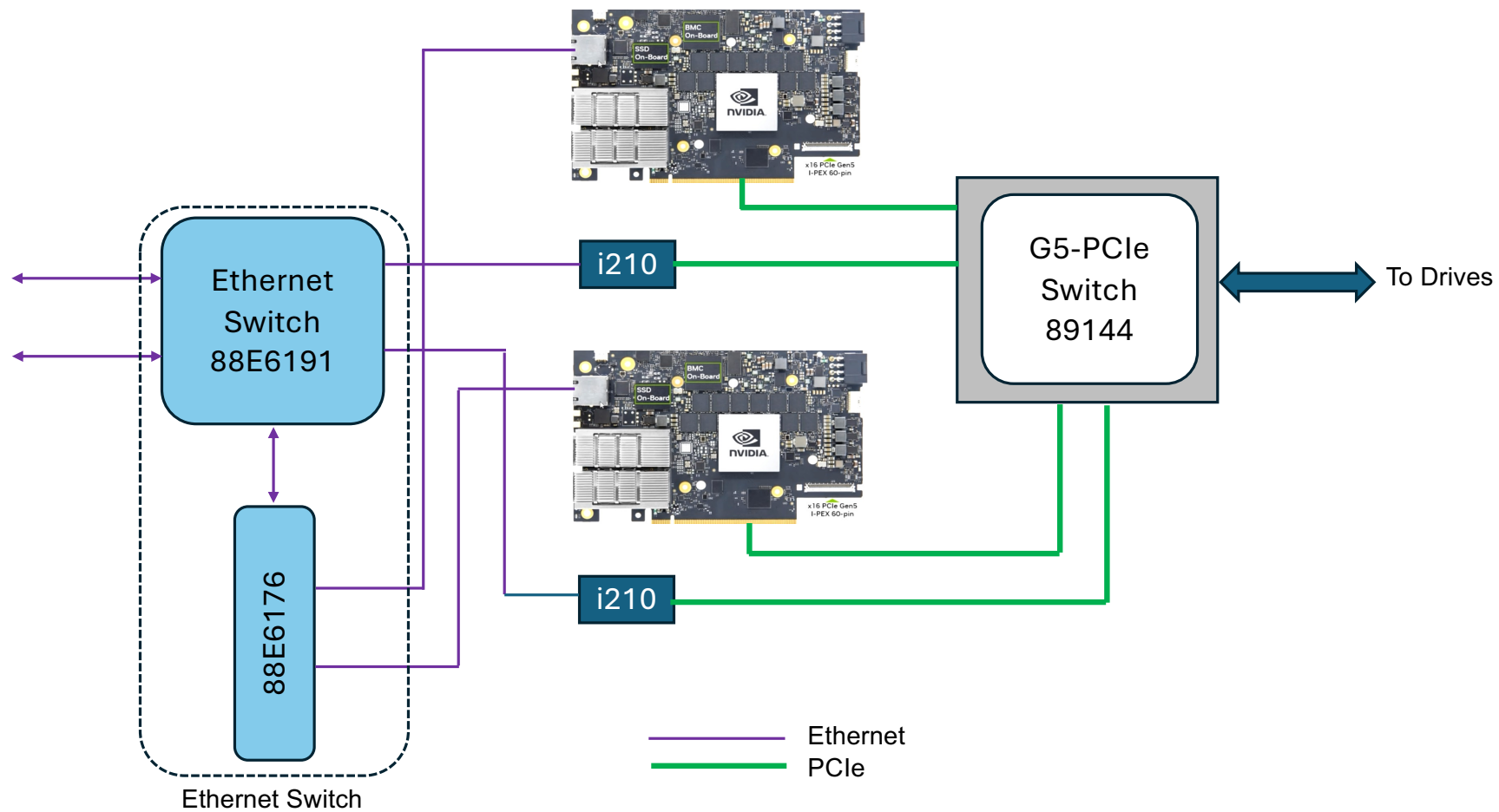


2U26 JBOF Canister

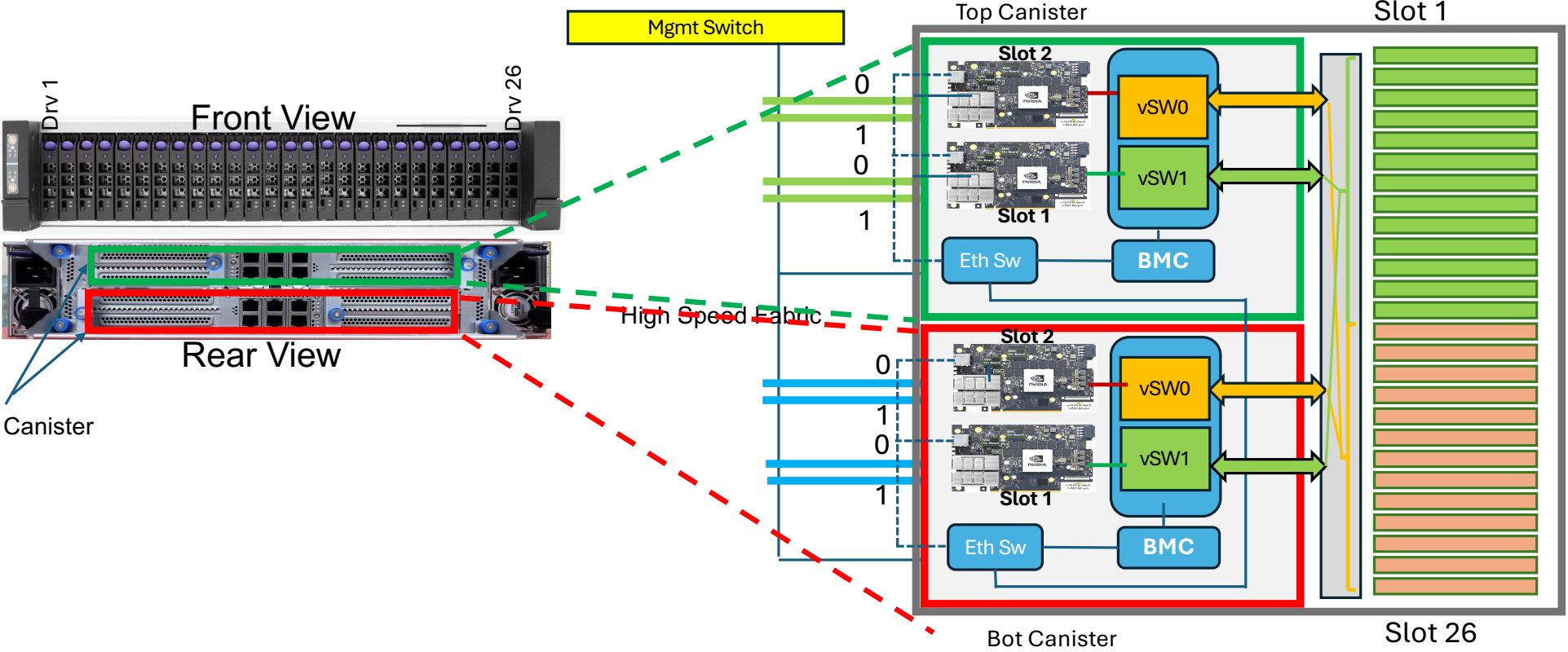
Key components

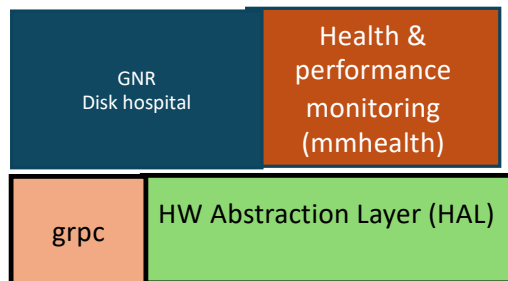


- 2x PCIe FHHL
- IO module
- 2x PCIe FHHL
- 2x PCIe FHHL
- IO module
- 2x PCIe FHHL
- Controller Canister B
- Enclosure Chassis
- Canister chassis[1 .. 2]
 - M.2 log drive [1 .. 4]
 - BlueField3 [1 .. 4]
 - Pigtail cable
 - Transceiver [1 .. 2]
 - Highspeed cable
- Rail kit
- Drive assembly [1 .. 26]
 - Drive
 - Drive carrier
- Power Adapter Assembly [1 .. 2]
 - Power Supply
 - Cable retainer
 - Power cable



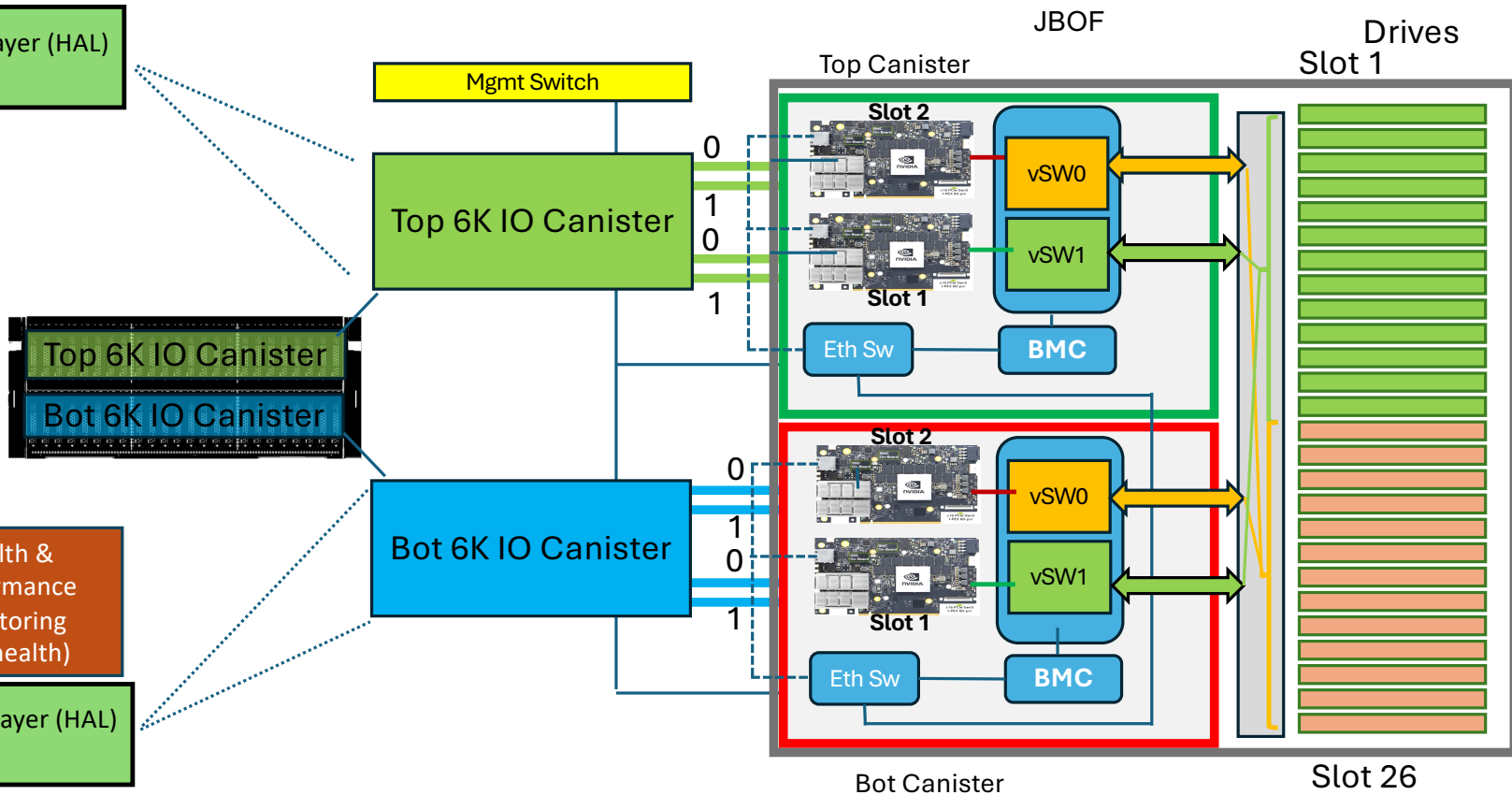
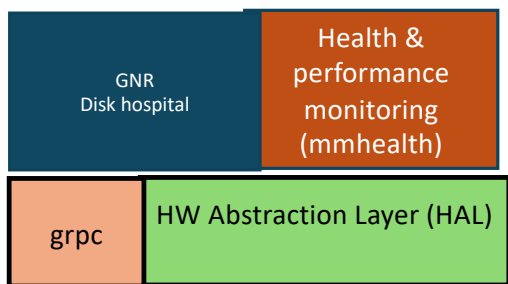
JBOF



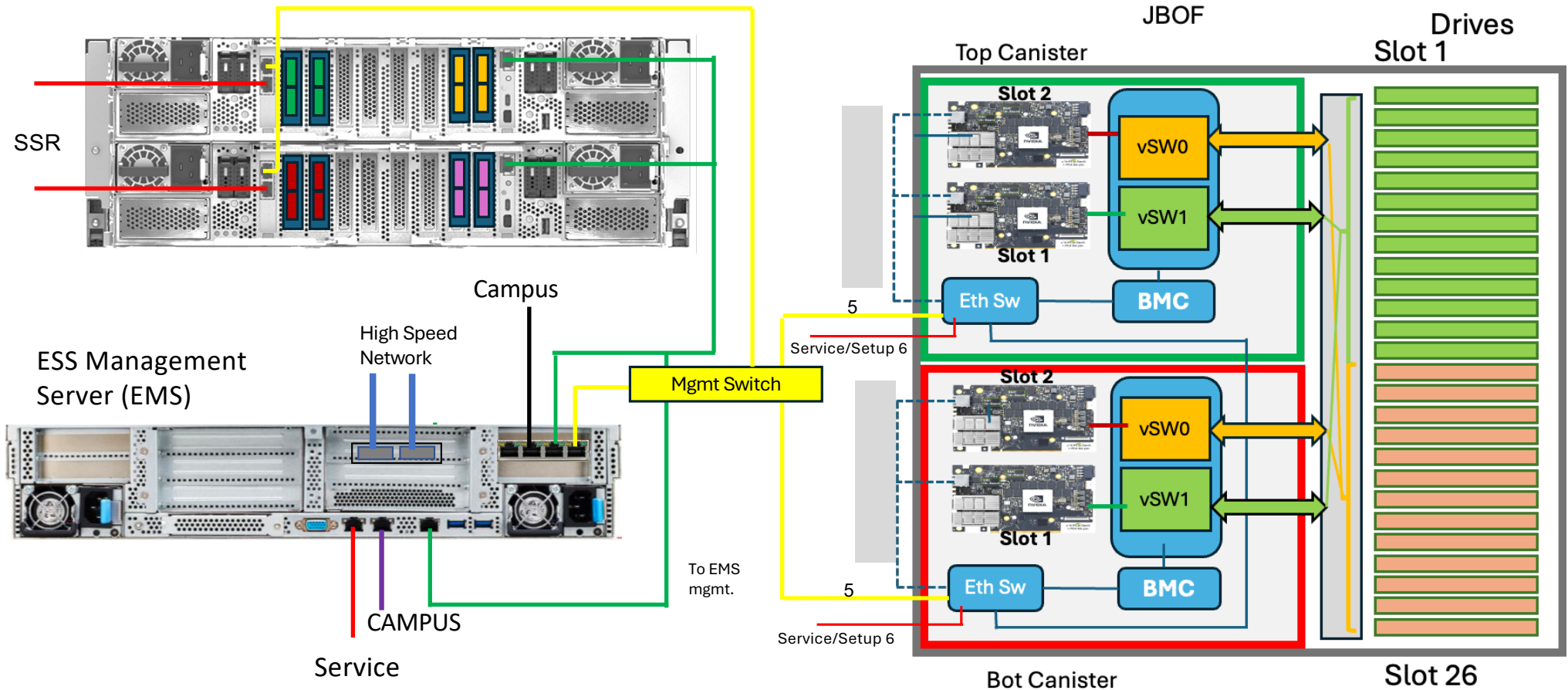


gRPC = admin and
out of band mgmt.

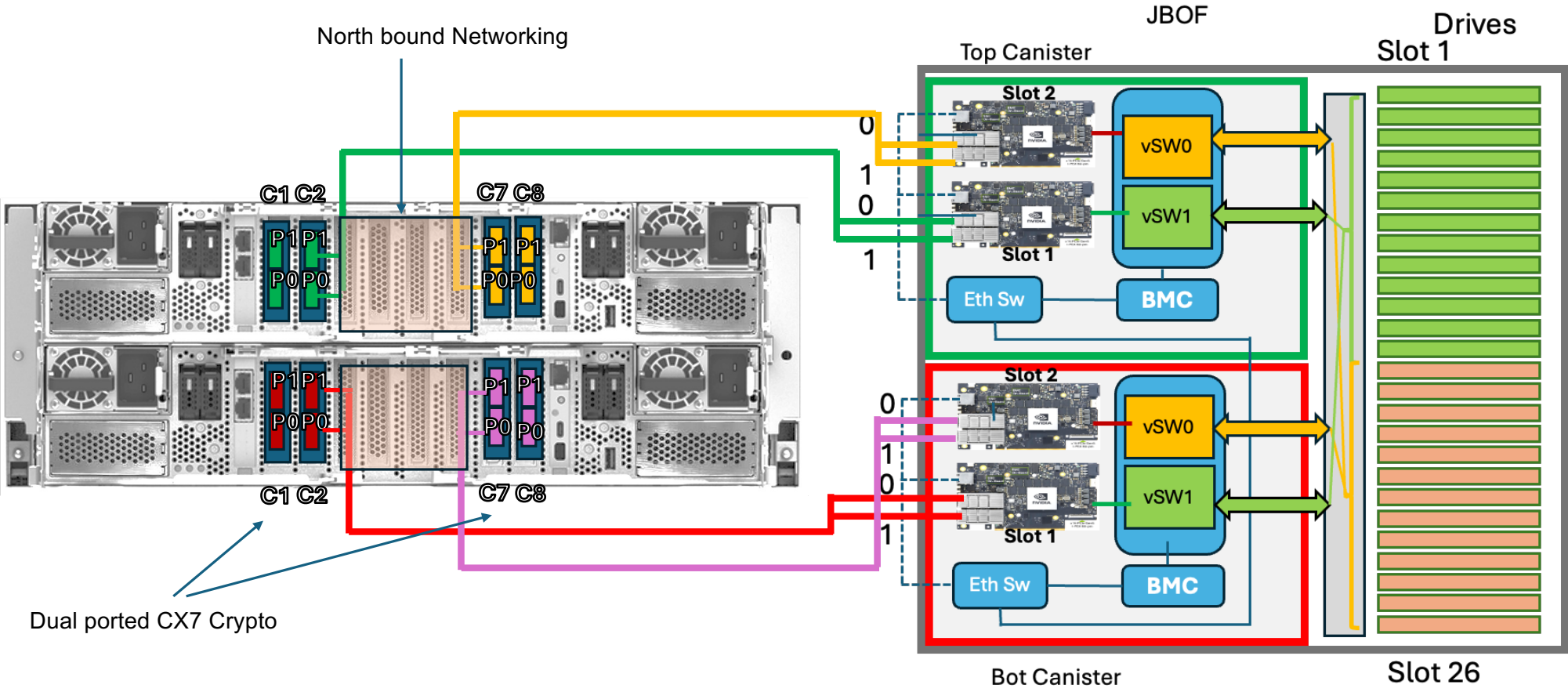
HAL = Hardware
management layer
End point HW mgmt.
and control.



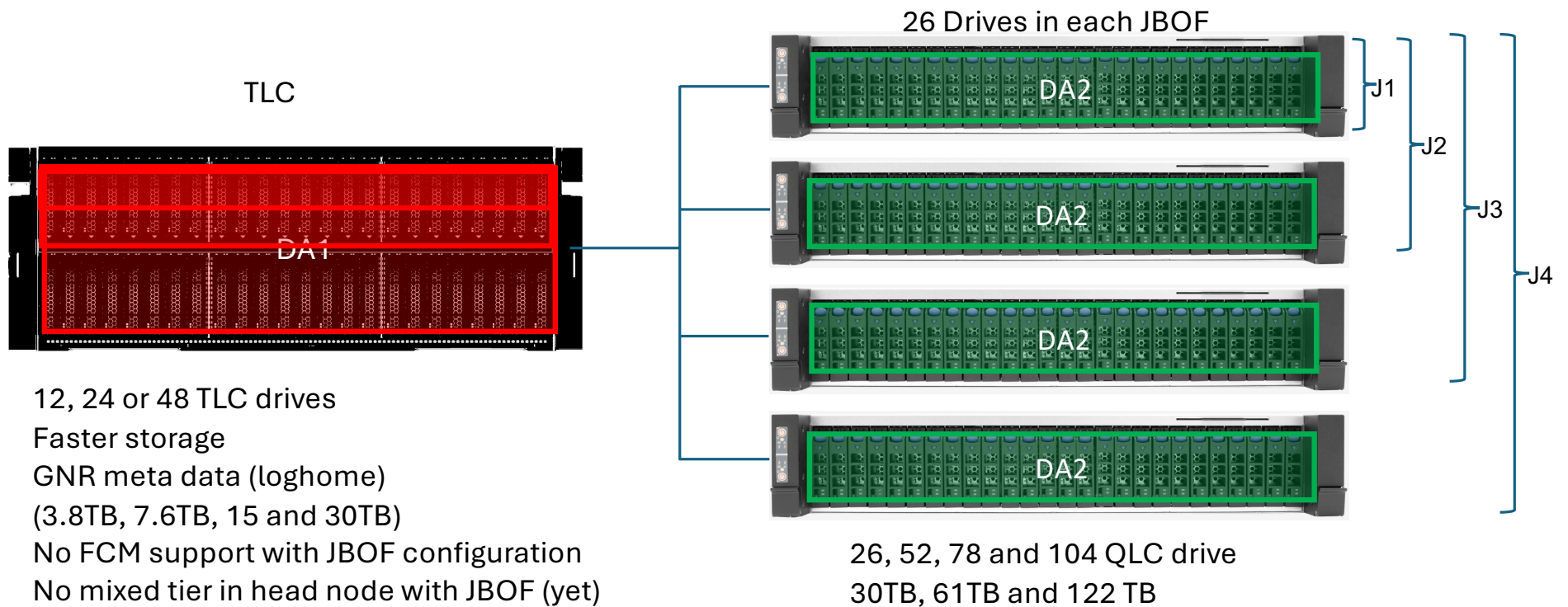
2U26 – BMC and management networking

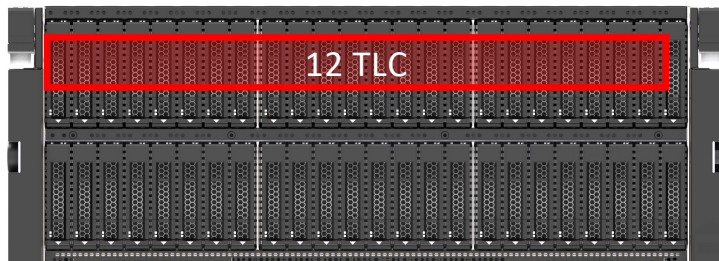


High speed network attachment and drive mapping

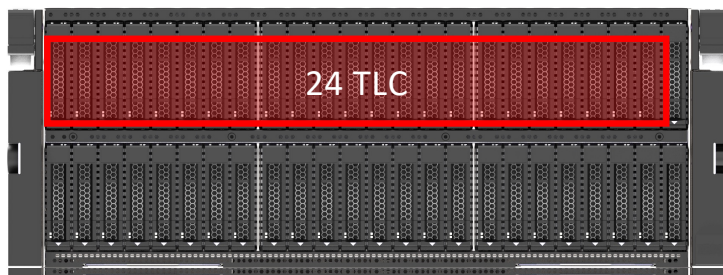


Extreme deep capacity with QLC while maintain TLC performance

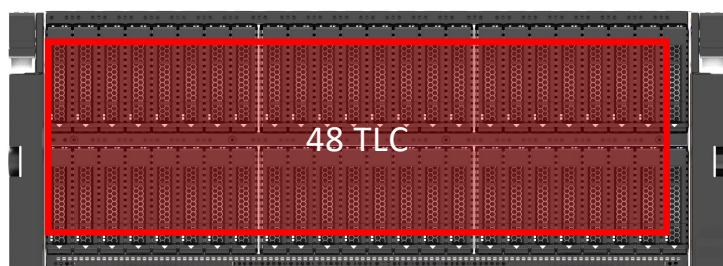




OR



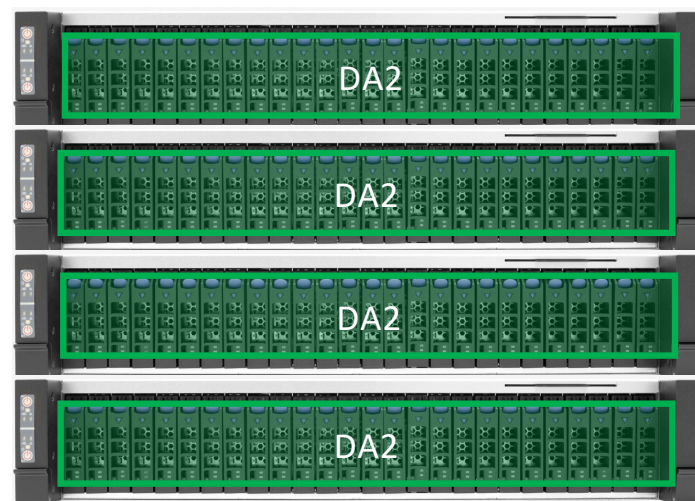
OR



Coming

MES

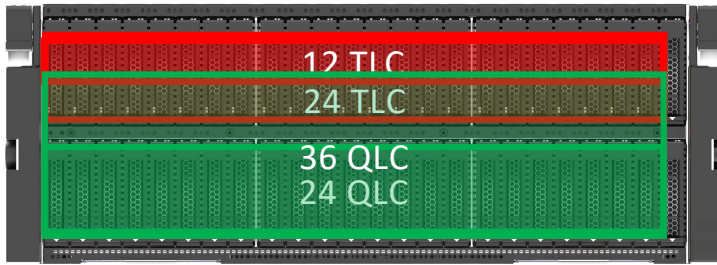
26 Drives in each JBOF



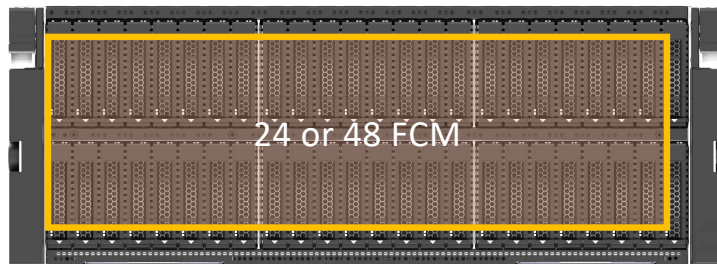
26, 52, 78 and 104 QLC drive
30, 61TB and 122 TB (30 coming soon)

1, 2, 3 or 4 JBOF can be added

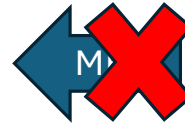
Mixed Tier



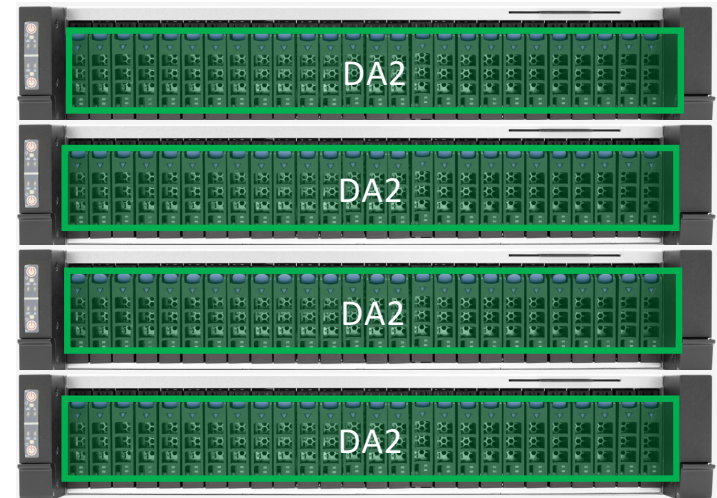
FCM

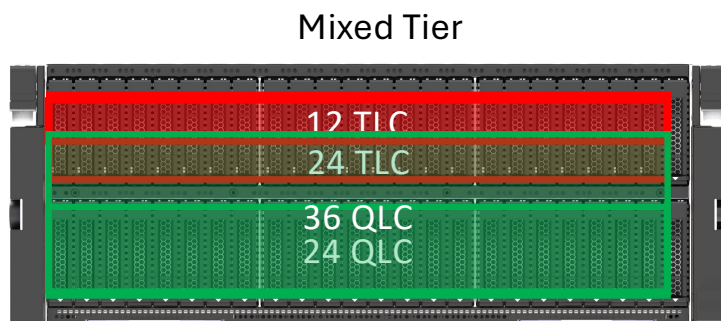


Coming

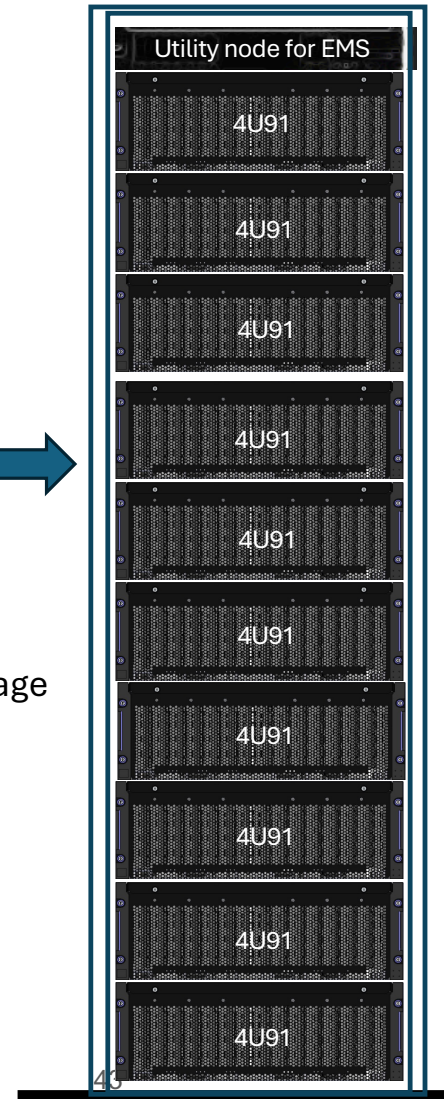


26 Drives in each JBOF

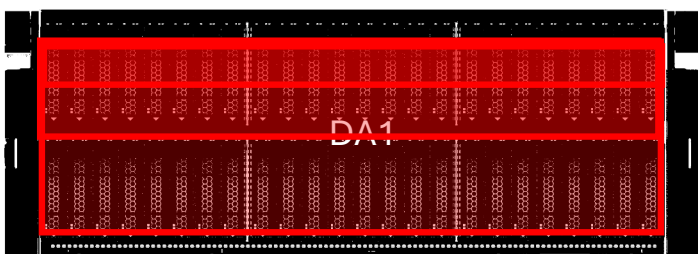




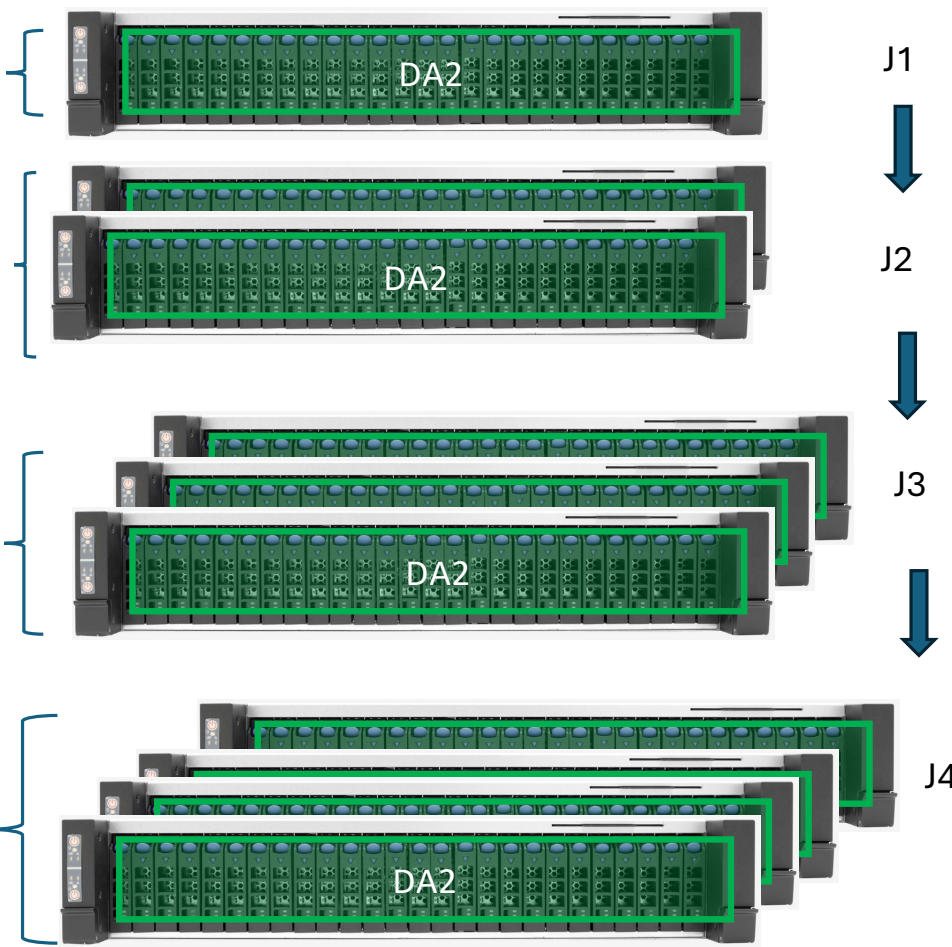
HDD Tier to Tiered TLC/QLC storage



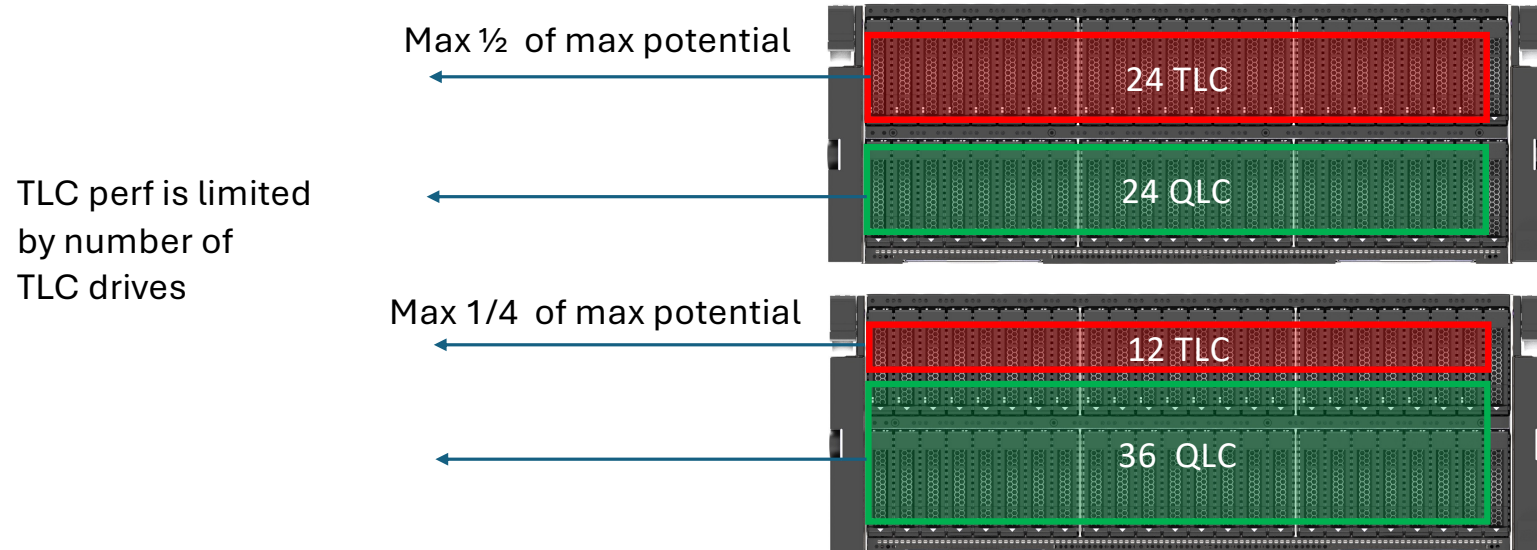
MES of JBOFs (coming)



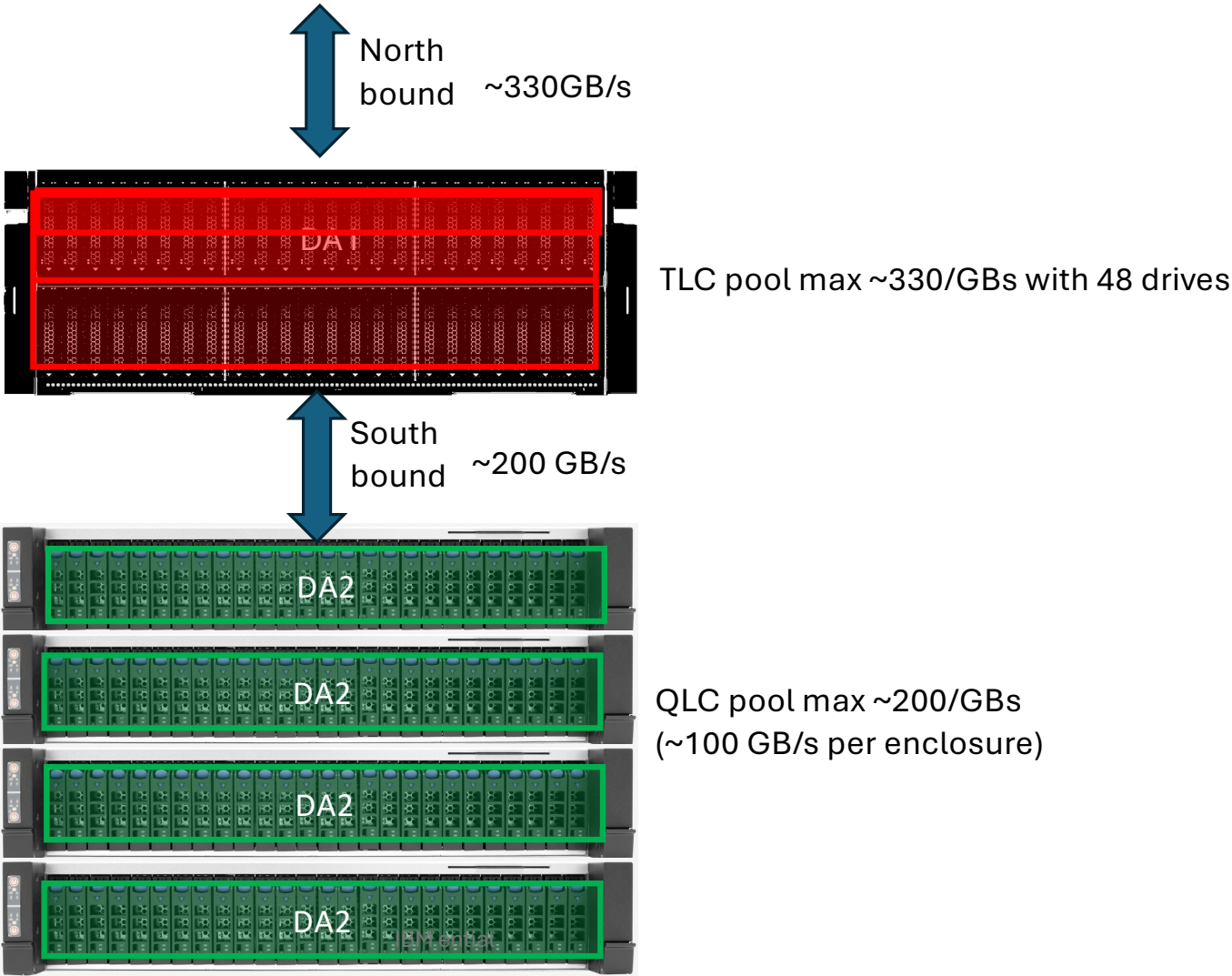
26 Drives in each JBOF

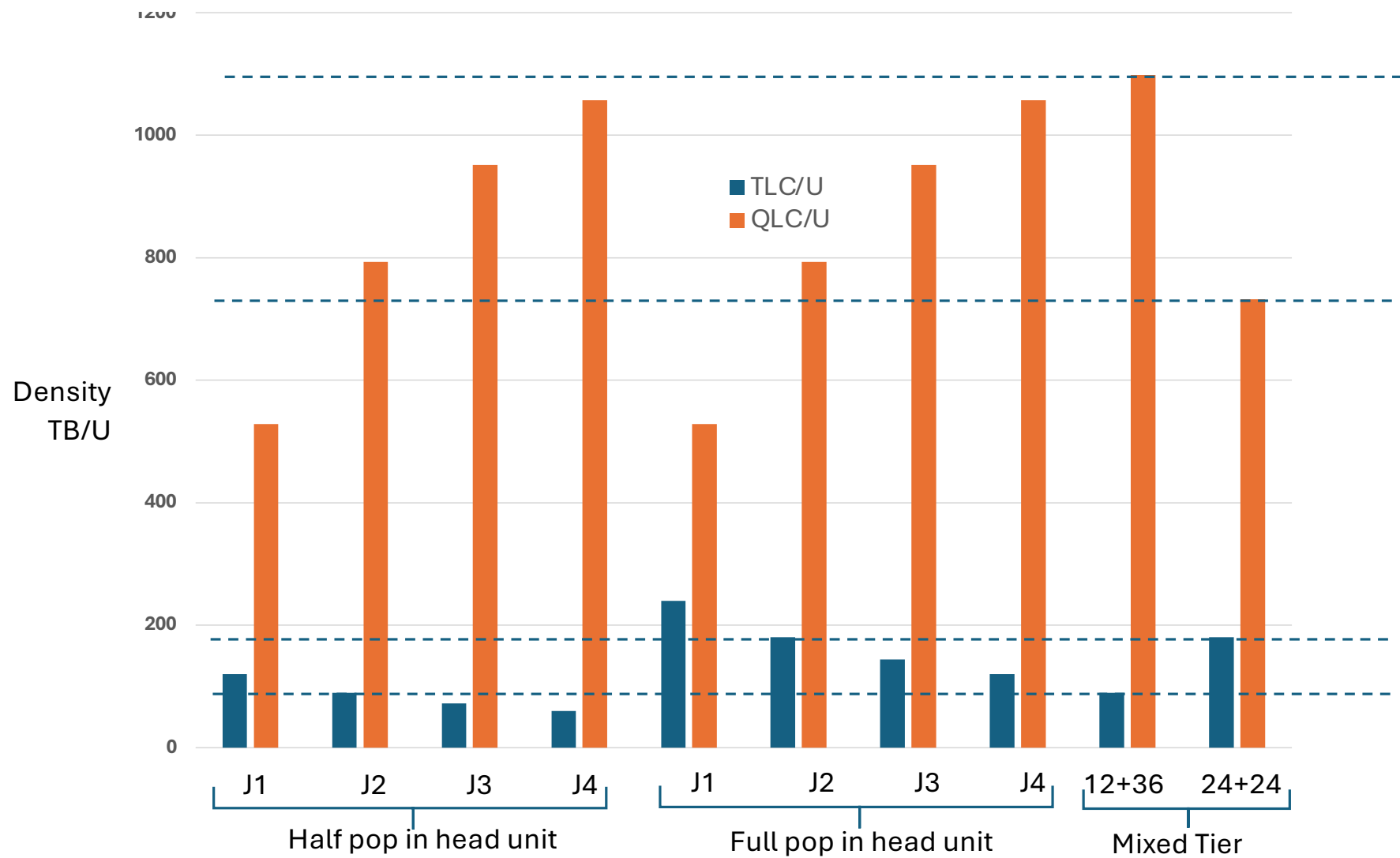


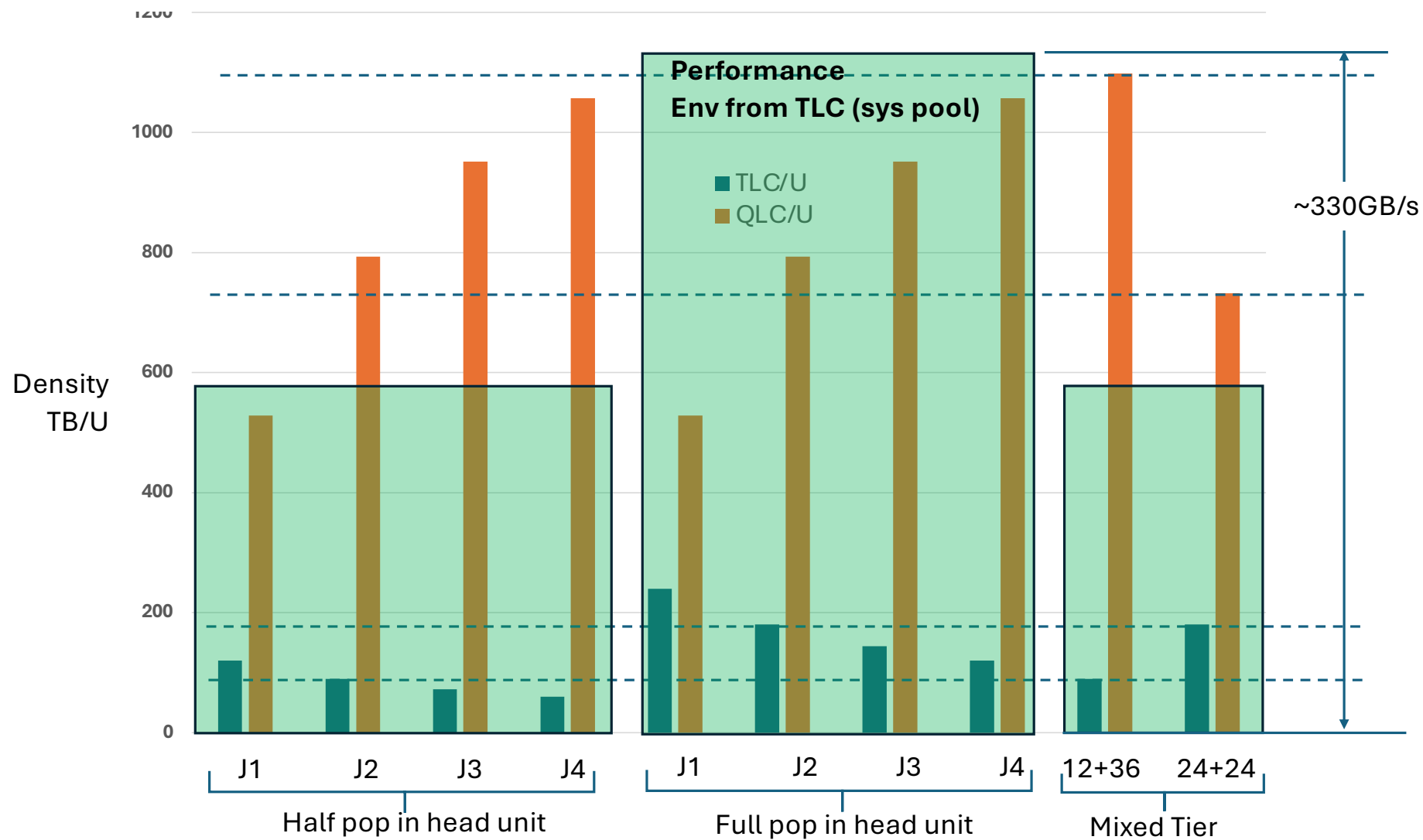
Performance Characteristics of Tiered Configuration

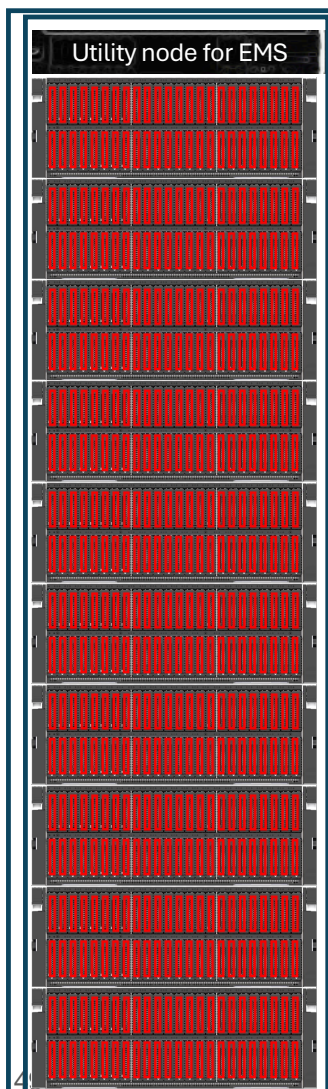


Performance Characteristics of JBOF Configuration









TLC Performance

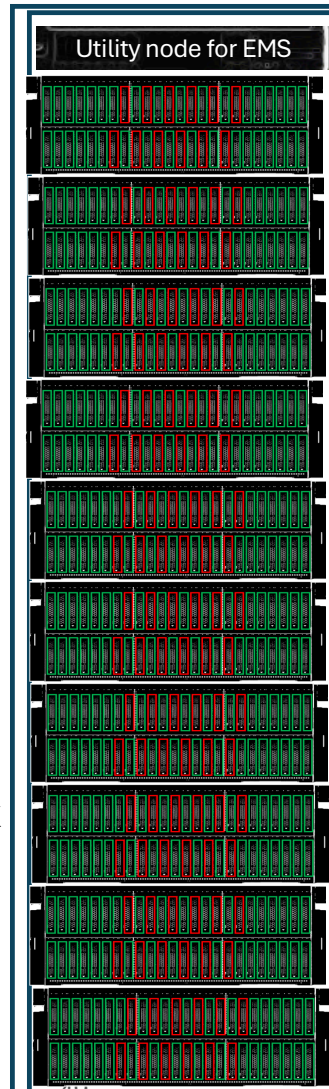
**3200 GB/s
From TLC**

Per BB

TLC
720 TB TLC
24 x 30TB

1440 TB TLC
48 x 30TB

**Max per rack
14.4PB**



Mixed TLC/QLC Tier

**800GB/s
From TLC**

**Deep
Capacity
per Building
Block**

**24TLC (30TB)
24 QLC (122TB)**
720 TB TLC
2928 TB QLC

**12TLC (30TB)
36 QLC (122TB)**
360 TB TLC
4392 TB QLC

**Max per rack
47.5PB**



Flash Hybrid with QLC in JBOF

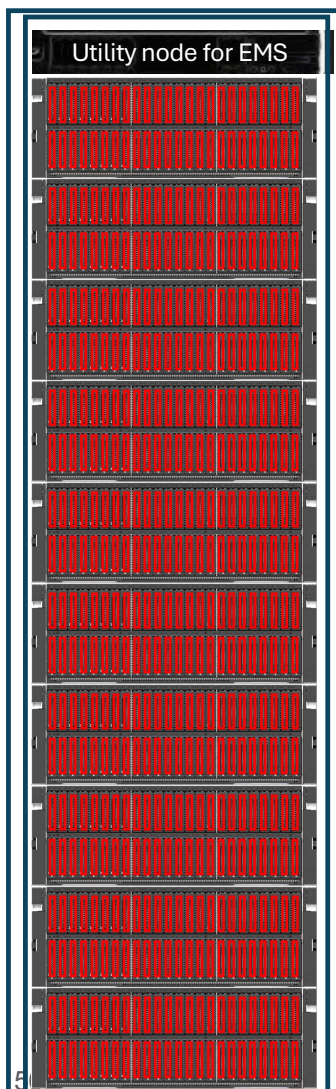
**960GB/s
from TLC**

**Extreme Deep
Capacity per
Building Block while
maintaining TLC
performance**

TLC
360 TB TLC 12 x 30TB
720 TB TLC 24 x 30TB
1440 TB TLC 48 x 30TB

QLC
3172 TB – J1 26 x 122TB
6344 TB – J2 52 x 122TB
9516 TB – J3 78 x 122 TB
12688 TB – J4 104 x 122TB

**Max per rack
42.3PB**



**TLC
Performance**

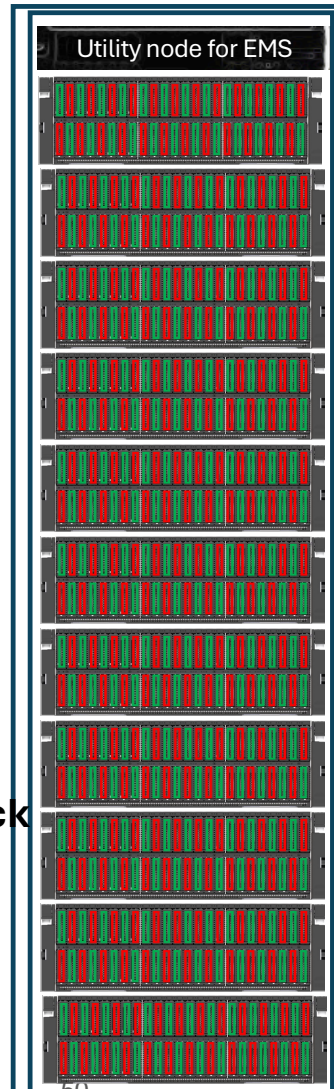
**3200 GB/s
From TLC**

Per BB

**TLC
720 TB TLC
24 x 30TB**

**1440 TB TLC
48 x 30TB**

**Max per rack
14.4PB**



**TLC/QLC Tier
Performance and
capacity**

**1600GB/s
From TLC**

**Deep
Capacity
per Building
Block**

**24TLC (30TB)
24 QLC (122TB)
720 TB TLC
2928 TB QLC**

**Max per rack
36.4 PB**



**Flash Hybrid
with QLC in JBOF**

**1600GB/s
from TLC**

**Extreme Deep
Capacity per
Building Block while
maintaining TLC
performance**

TLC

**360 TB TLC 12 x 30TB
720 TB TLC 24 x 30TB
1440 TB TLC 48 x 30TB**

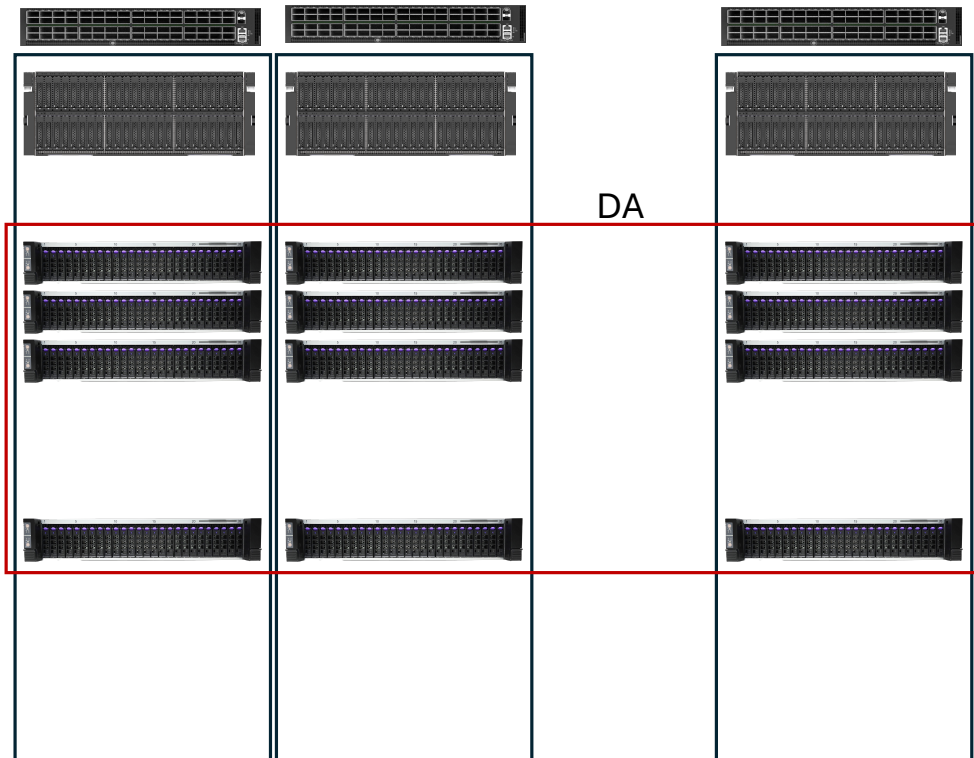
QLC

**3172 TB – J1 26 x 122TB
6344 TB – J2 52 x 122TB
9516 TB – J3 78 x 122 TB
12688 TB – J4 104 x 122TB**

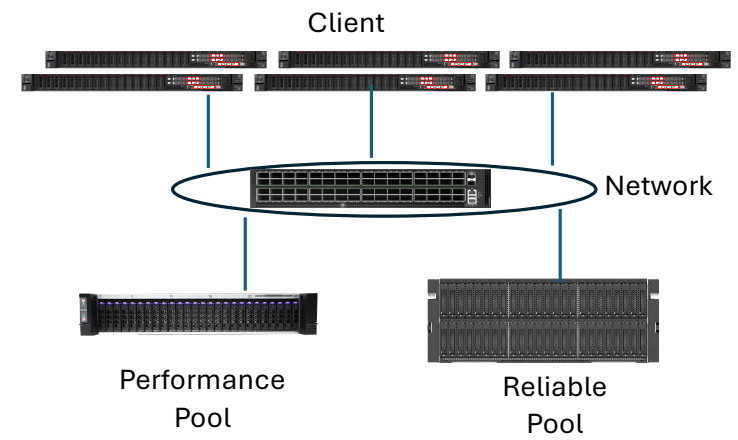
**Max per rack
41.9 PB**

Enabled with switched... and coming soon

Rack Level Redundancy

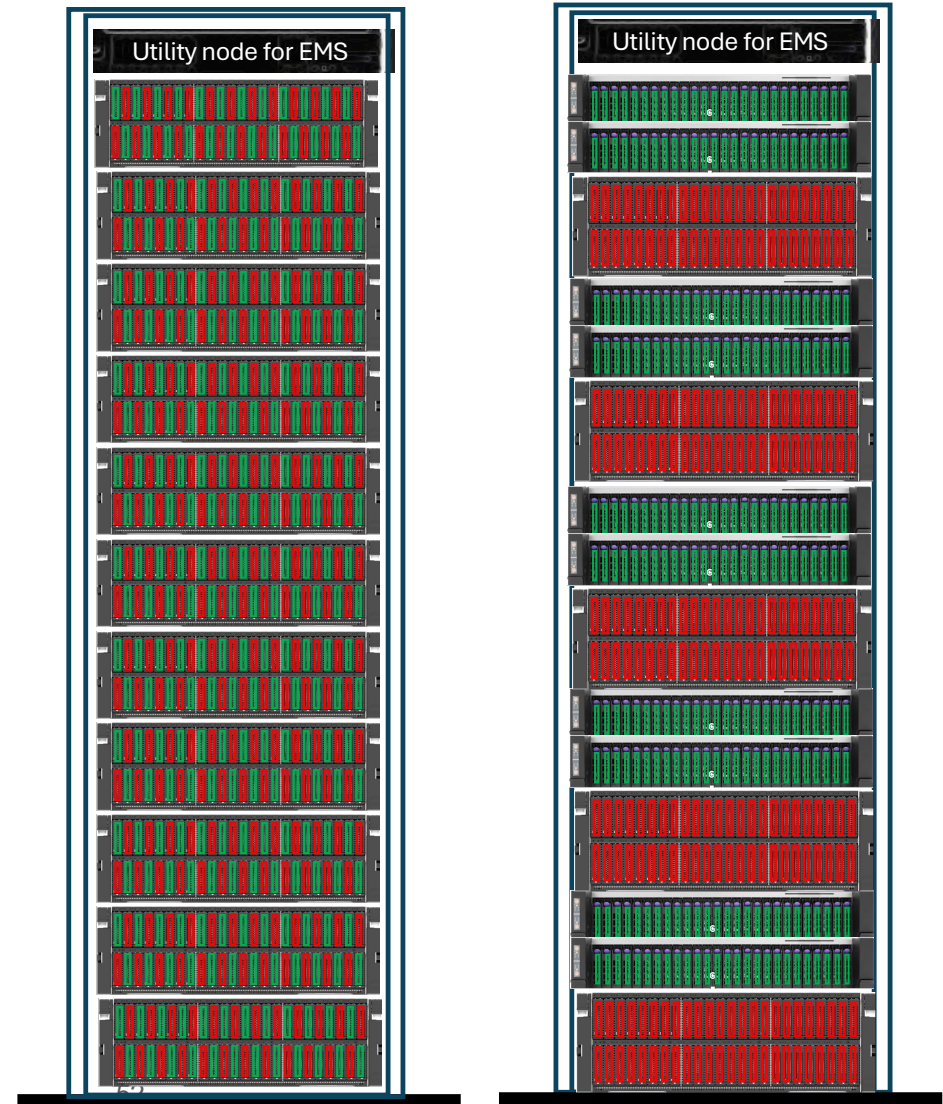


Data Acceleration Tier with JBOF



Extreme Deep Capacity with performance

- Deeper Flash Capacity by increasing number of QLC in Scale System 6000 head node – flexible mixed tier
- Provide Extreme deep Flash Capacity of a building block by JBOF attachment





Impact of Replacing bmSem Linux Kernel Semaphore with Mutex



Performance improvements were observed on 4K and 8K random DIO write workloads, particularly at high thread counts, for single client and multi client configurations



The mutex changes appear to reduce contention under high concurrency workloads, improving Random DIO write scalability

Workload command:

```
mpiexec -f <node_file> -n <total_mpi_processes> gpfsperf-mpi write rand /gpfs/fs1/gpfsperf/gfilelarge -r <io_size>K -n 999999G -dio -millis 180000 -th <threads_per_process>
```

Workload Characteristics

- Random DIO write workload
- 4KiB and 8KiB transfer sizes
- Runtime: 180 seconds
- Comparison between 6.0.0 GA and 6.0.1 daily build



Rogelio Rivera – rriverag@mx1.ibm.com
Mara Miranda - maramb@mx1.ibm.com

Mutex (struct mutex)

For performance boost, mutex provides better results specially for small size IO writes (e.g. 4k, 8k, ..) in contested environment because of:

- Optimistic spinning based on MCS SpinLock which avoid cache bouncing.
- No counting overhead as compared to semaphores.
- Handles thread priority inversion better.
- In certain edge cases, might also lead to CPU starvation where kernel is not build with preemption enabled.

Mutex (struct mutex)...

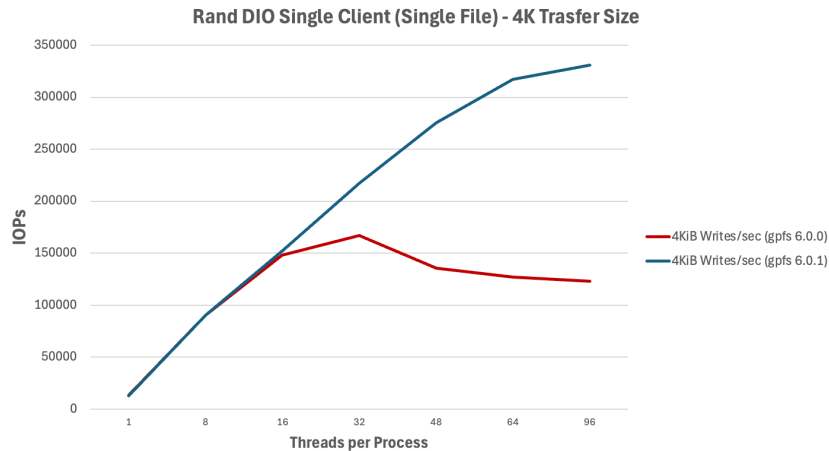
- On latest RedHat versions preemption is enabled.
- GPFS exploit mutexes for performance gains on 6.0.1.0 release.
- On latest OS (RHEL9) gpfs uses mutex for synchronization by default.
- For older kernels (RHEL8) not enabled by default.
- 6.0.1.0 release also provides ability to switch between implementations for comparison.

CONFIG_PREEMPTION

- Latest kernels specially build for x86 arch. Have preemption enabled.
- Which allow even kernel threads to get preempted after their scheduler time slice is expired.
- This avoid cpu hogging.
- Usual points of preemption are:
 - Lock release
 - New checks of `might_sleep()` added

Impact of Replacing bmSem Linux Kernel Semaphore with Mutex

SINGLE CLIENT Configuration



When comparing GPFS 6.0.0.0 with GPFS 6.0.1.0, 4K and 8K random DIO write workloads in a single client configuration show the following improvement:



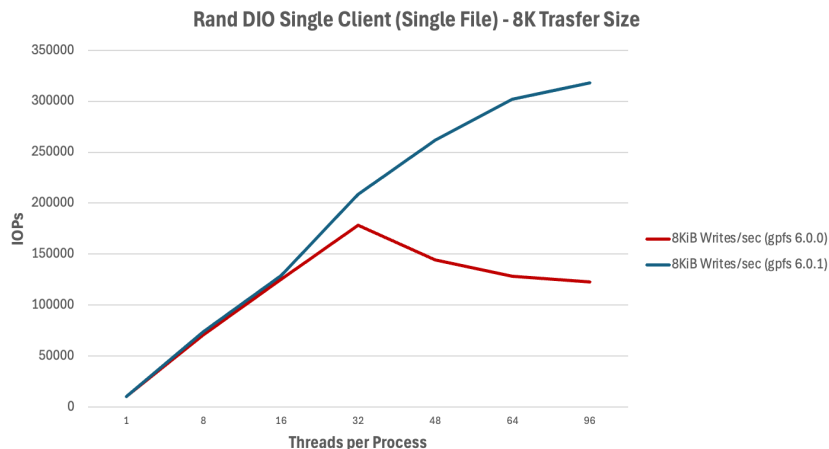
At low thread counts the performance improvement is minimal



Starting at 32 threads per node performance improvements are of 30% and over

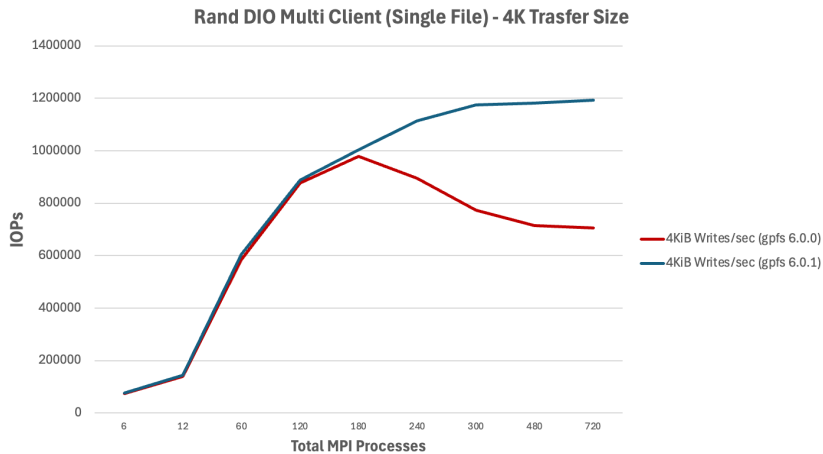


Performance continues to scales more efficiently as thread count increases, up to 170%



Impact of Replacing bmSem Linux Kernel Semaphore with Mutex

MULTI CLIENT Configuration



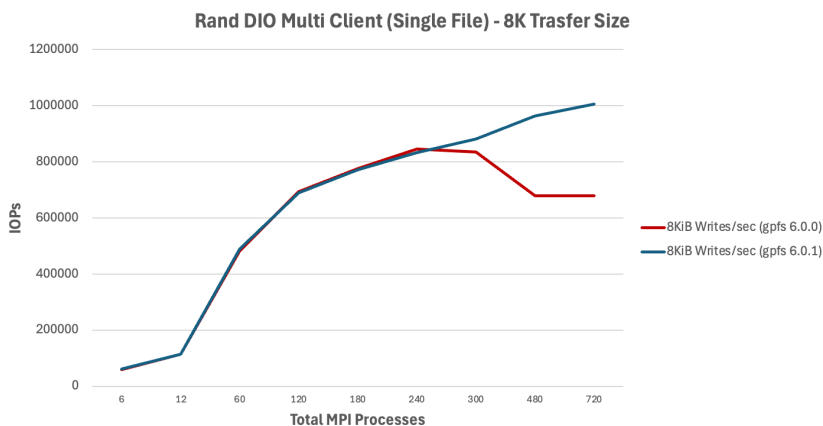
When comparing GPFS 6.0.0.0 with GPFS 6.0.1.0, 4K and 8K random DIO write workloads in multiple client configuration show the following:



The improvements show as aggregate, with 6 clients the improvements start at 180 processes



At higher process count the improvement is ~65-75%



Recommendations for Resiliency: Tuning to Address Short-timeouts in IBM Storage Scale

Recommendations for Cluster Stability

- The Storage Scale development team has been investigating node expel events observed in large-scale deployments, particularly in cases where TCP connections between nodes cannot be established within expected time limits.
- While additional product improvements are under development, the following configuration and tuning recommendations can help reduce expels related to TCP connection establishment (additional details are provided on the upcoming slides):
 - [mmchconfig](#) `proactiveReconnect=no -i`
In 5.2.2.0, **proactiveReconnect** is disabled for all new clusters but existing clusters require running mmchconfig for this change to take effect
 - [mmchconfig](#) `idleSocketTimeout=0 -l`
To prevent connected sockets from being disconnected due to hitting a threshold of idle activity – now the default for this tuning option as of 5.1.9.0
 - [mmchconfig](#) `allToAllConnection=all` # means to connect to both local and remote cluster mmfsd's
To ensure that nodes establish connections across the cluster, shortly after joining, rather than on-demand as needed
 - Consider making sysctl changes to implement ARP tuning, with the goal being to minimize ARP activity on your cluster.

The most important parameters are:

gc_thresh1, gc_thresh2, gc_thresh3, gc_stale_time, gc_interval, mcast_solicit, and ucast_solicit

Details on the “**ARP Tuning Options to Consider to Improve Network Stability**” slides that follow, taken primarily from the 2024 Network Topics presentation given at the 2024 NYC User group:

<https://www.spectrumscaleug.org/wp-content/uploads/2025/04/SSUG24NYC11-Storage-Scale-Network-Topics-NYC.pdf>

- Consider taking steps to address potential congestion in your switch fabric, such as enabling Explicit Congestion Notification (ECN), if all switches and adapters support ECN (more details and suggestions on the “**Causes for Expels and Recommendations**” chart)

Details on Node Requested Expels – Shorter Timeout Flows

Note that there are potentially shorter timeouts that can lead to node requested expels before hitting the 300 second timeout that occurs when communicating with destinations that have connections already established. For this reason, consider tuning the system to minimize reconnects (described later).

If a node needs to (1) reconnect to, or (2) send an RPC message to another node which it has no established connection to, a shorter timeout can be hit in the code paths that establish new connections. Two examples of these shorter timeout cases, for TCP connections, are as follows:

1. When a new TCP connection must be established to another host, if ARP resolution is not working and a valid ARP entry is not present for the destination IP, a "No Route To Host"* error will occur establishing the connection, by default after 3 seconds of ARP retries (controlled by the applicable `sysctl mcast_solicit` parameter). Storage Scale will retry establishing the connection, but the RPC could timeout in as little as 6 seconds if ARP is not working and the default tuning is in effect.
2. If ARP is working but TCP connections cannot be established, an RPC timeout can, with default TCP tuning, occur in about two minutes (Storage Scale will retry the TCP connection if it times out on connect). This ~60 second timeout covers more than the TCP connection timeout, as Storage Scale needs to run additional (TLS) authentication code in gskit after the connection is established.

* Note that "No route to host" (EHOSTUNREACH) errors can be seen in other flows and one possible source of this error can be a firewall misconfiguration that blocks Scale messages.

Details on Node Requested Expels – Shorter Timeout Example 1

- Example of (1 on previous **Shorter Timeout Flows** slide) timeout establishing new TCP connection when ARP doesn't resolve (no ARP entry causes "No route to host", EHOSTUNREACH, error):

```
2024-10-05_19:13:47.152+0200: [N] Connecting to 10.200.42.204 fscs-sr650v2-4
<c0n21>:[0]
```

```
2024-10-05_19:13:50.251+0200: [N] Close connection to 10.200.42.204 fscs-
sr650v2-4 <c0n21>:[0] (No route to host)
```

```
2024-10-05_19:13:50.251+0200: [N] Retry connection to 10.200.42.204 fscs-
sr650v2-4 <c0n21>:[0]
```

```
2024-10-05_19:13:53.323+0200: [N] Close connection to 10.200.42.204 fscs-
sr650v2-4 <c0n21>:[0] (No route to host)
```

```
[...] >> Network connectivity established at this point (ARP works at 19:13:54 but
the client has already decided it will send an expel request at 19:15:05 [...])
```

```
2024-10-05_19:15:05.620+0200: [N] Request sent to 9.152.187.70 (fscs-sr650-49)
to expel 9.152.185.166 (fscs-sr650v2-4) from cluster GNRocks.fscs-ess3500-1-a
```

- Connectivity is down for only 6.1 seconds (from 19:13:47.152 - 19:13:53.251) before the client makes the decision to send a request for fscs-sr650v2-4 be expelled, but it delays sending the expel request for fscs-sr650v2-4 to the cluster manager to give the cluster manager time to discover if there is a network issue impacting fscs-sr650v2-4, which might prevent it from renewing its lease.
- The total time taken from introducing the network connectivity failure to the time that the expel request is sent to the cluster manager is 1 minute + 18 seconds (19:15:05 - 19:13:47).

Details on Node Requested Expels – Shorter Timeout Example 2

- Example of (2 on previous **Shorter Timeout Flows** slide) timeout establishing new TCP connection when we have a valid ARP entry for the node being connected to (fscs-sr650v2-4):

```
2024-10-05_17:21:11.732+0200: [N] Connecting to 10.200.42.204 fscs-sr650v2-4
<c0n21>:[0]
```

```
2024-10-05_17:22:15.241+0200: [N] Close connection to 10.200.42.204 fscs-
sr650v2-4 <c0n21>:[0] (Connection timed out)
```

```
2024-10-05_17:22:15.242+0200: [N] Retry connection to 10.200.42.204 fscs-
sr650v2-4 <c0n21>:[0]
```

```
2024-10-05_17:23:18.775+0200: [N] Close connection to 10.200.42.204 fscs-
sr650v2-4 <c0n21>:[0] (Connection timed out)
```

```
2024-10-05_17:23:30.200+0200: [N] Expel data is not collected on any node. It
was collected recently at 2024-10-05_16:54:34+0200.
```

```
2024-10-05_17:23:30.200+0200: [N] Request sent to 9.152.187.70 (fscs-sr650-49)
to expel 9.152.185.166 (fscs-sr650v2-4) from cluster GNRocks.fscs-ess3500-1-a
```

- Connectivity is down for 2 minutes and 7 seconds (from 17:21:11.732 – 17:23:18.775) before the client makes the decision to send a request for fscs-sr650v2-4 be expelled (the expel request is sent 12 seconds later).
- The total time taken from introducing the network connectivity failure to the time that the expel request is sent to the cluster manager is 2 minutes and 19 seconds (17:23:30 - 17:21:11).

Proactive Reconnect and RTO/Expel Timelines

T1 - After ~8 seconds RTO is 6,496 ms

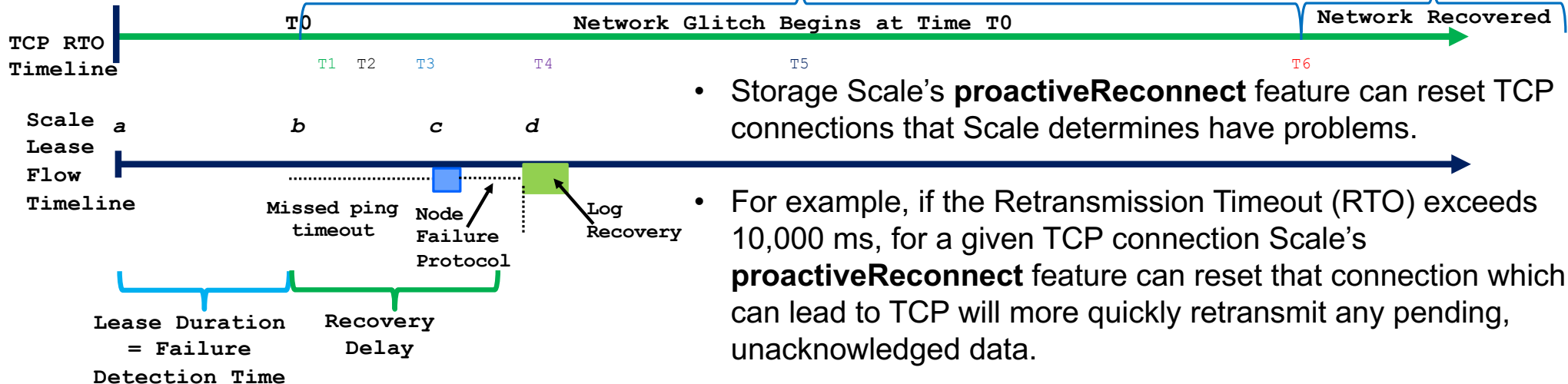
T2 - After ~14 seconds RTO is 12,992 ms

T3 - After ~27 seconds RTO is 25,984 ms

T4 - After ~53 seconds RTO is 51,968 ms

T5 - After ~105 seconds RTO is 103,936 ms

T6 - After ~202 seconds RTO is 120,000 ms



- TCP uses an exponential backoff algorithm to control retransmission timing (RTO= Retransmission Timeout) in response to potential network congestion.
- After each retransmission, TCP doubles the time interval before it attempts to retransmit again, effectively slowing down the rate of retransmission attempts to avoid overwhelming the network.

- Storage Scale's **proactiveReconnect** feature can reset TCP connections that Scale determines have problems.

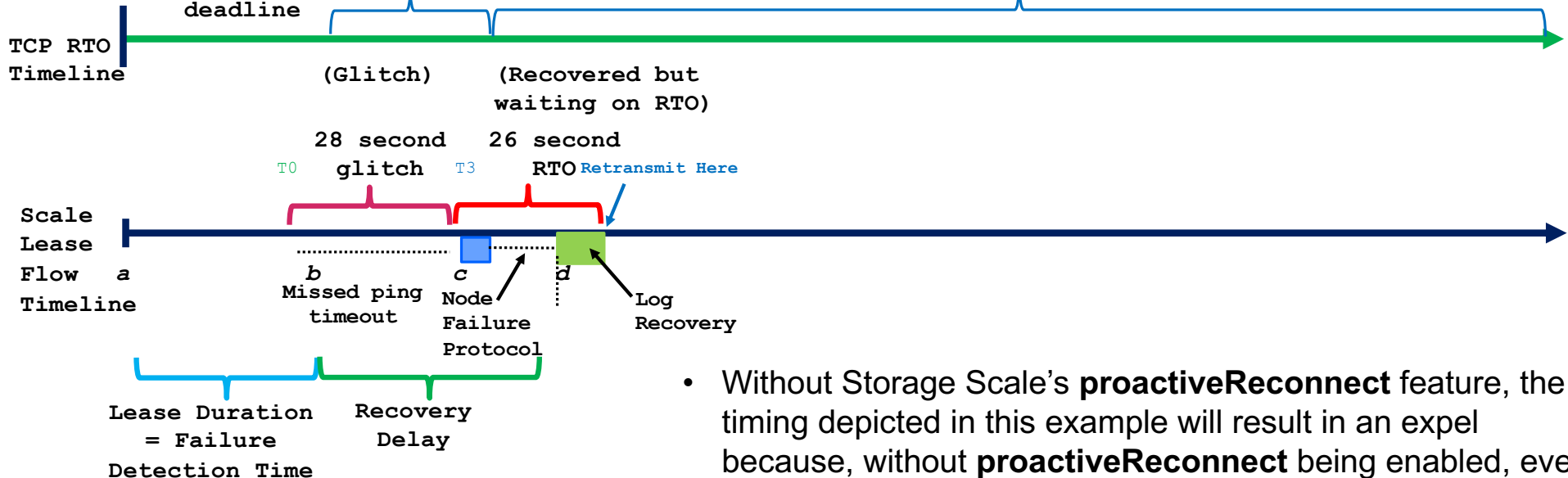
- For example, if the Retransmission Timeout (RTO) exceeds 10,000 ms, for a given TCP connection Scale's **proactiveReconnect** feature can reset that connection which can lead to TCP will more quickly retransmit any pending, unacknowledged data.

Proactive Reconnect and RTO/Expel Timelines

T3 - After ~27 seconds RTO is 25,984 ms

Network Glitch Begins at
Time T0 -> 1 second
before the lease renewal
deadline

Network Recovered



- This example shows a case in which **proactiveReconnect can help**. .the network glitch occurs about 1 second before the lease expires and the network recovers after about 28 seconds, at which point the RTO is almost 26 seconds.

- Without Storage Scale's **proactiveReconnect** feature, the timing depicted in this example will result in an expel because, without **proactiveReconnect** being enabled, even though the network recovers before the expel (c – Node Failure Protocol), any pending data for the connection won't be retransmitted until the 26 second RTO expires

Avoiding the “Shorter Timeout” Node Requested Expels

- As shown on the previous slides, short interruptions to network connectivity can lead to node requested expels if the network issues coincide with windows in which new network connections need to be established, and the long-term plan is to address this issue via design changes
- Until we have code changes to address the shorter timeouts in the connection establishment path, we can try to minimize how often nodes will establish new connections as follows:
 - Disable **proactiveReconnect** (e.g., [mmchconfig](#) `proactiveReconnect=no -i`) to prevent Storage Scale from reconnecting established connections (in 5.2.2.0, **proactiveReconnect** is disabled for all new clusters)
 - Disable idle sockets from being disconnected (`“mmchconfig idleSocketTimeout=0 -i”`, which becomes the default for this tuning option in 5.1.9.0)
 - Consider enabling the **allToAllConnection** option to force Storage Scale to establish connections to other nodes shortly after mmfsd comes up (this makes connection failures less likely to happen in a cluster that is in steady-state production, with all nodes already running jobs):
[mmchconfig](#) **allToAllConnection=all** # means to connect to both local and remote cluster mmfsd's
(See: <https://www.spectrumscaleug.org/wp-content/uploads/2020/09/004-spectrum-scale-performance-update.pdf>)

Causes for Expels and Recommendations

- After eliminating packet loss related to hardware issues and tuning/configuration settings, another issue to check is if there are resource issues on the nodes that may impact performance.
- If the nodes are short on memory, this can impact performance – check for out-of-memory (OOM) errors that may be logged in /var/log/messages and monitor memory use while workloads are running.
- Also look for CPU contention that may impact performance – Storage Scale will log ‘critical thread’ monitoring’ issues when threads are blocked for excessive amounts of time and Linux will log ‘stuck CPU’ warnings’ when threads are blocked for more than 120 seconds.
- On quorum nodes, local disk performance is important as slow I/O times to writes to the file /var/mmfs/gen/LastLeaseRequestSent may result in the quorum nodes being expelled.
- As previously noted, in some cases, the [mmchconfig](#) option **proactiveReconnect** may cause issues in that, if the network experiences stability issues, **proactiveReconnect** can end up accelerating the timeline for expels and exposing nodes to shorter expel timeouts that occur during the establishment of TCP/IP connections, so consider minimizing the establishment of new connections as recommended on a previous slide
 - The up-side of **proactiveReconnect** is that it can (1) avoid timing holes as described on the **Proactive Reconnect and RTO/Expel Timelines** slide, and (2) work-around the ‘Socket Lockup’ exposure seen on older Linux levels in which TCP connections could hang as a result of Linux kernel issues (e.g. 2.6.x kernel exposures related to how TIME_WAIT and FIN_WAIT were handled during heavy retransmission)
- Also consider implementing ARP tuning, as broadcast operations can introduce more stress on the network. . . in general larger systems should consider some form of ARP tuning – traditional guidance has been to implement such tuning on systems larger than 1000 nodes, but we’ve also seen good success resolving network issues by implementing ARP tuning on much smaller clusters (even systems as large as 128 nodes)

Causes for Expels and Recommendations

- Another concern to consider is the potential for overloading the network, leading to drops on switches and adapters, particularly when very large TCP/IP windows are used and/or many to few communication patterns occur.
- Some techniques that can address congestion and overloading of a network fabric:
 1. Enabling Explicit Congestion Notification (ECN), if all switches and adapters support ECN, allowing senders to be notified of congestion without dropping packets. (This option is strongly recommended.)
 2. Using smaller TCP window sizes, which reduces the total data that can be in flight for any TCP connection.
 3. Enabling more modern congestion control algorithms, such as Google's BBR (Bottleneck Bandwidth and Round-trip propagation time, available in 4.9 kernel levels by setting: "**sysctl -w net.ipv4.tcp_congestion_control=bbr**").
 4. Enabling an alternative congestion control solution such as L4S (Low Latency Low Loss Scalable Throughput, which requires a recent kernel level, enabling ECN, and a compatible congestion control algorithm, e.g., "**sysctl -w net.ipv4.tcp_congestion_control=dctcp**").
 5. Consider Active Queue Management (AQM) scheme options, such as Proportional Integral controller Enhanced (PIE), that can be enabled to better manage queue lengths in routers and switches, allowing more proactive congestion notification and dropping of packets under congestion conditions.

ARP Tuning Options to Consider to Improve Network Stability

gc_thresh1: INTEGER Minimum number of entries to keep. Garbage collector will not purge entries if there are fewer than this number. Default: 128

gc_thresh2: INTEGER Threshold when garbage collector becomes more aggressive about purging entries. Entries older than 5 seconds will be cleared when over this number. Default: 512

gc_thresh3: INTEGER Maximum number of non-PERMANENT neighbor entries allowed. Increase this when using large numbers of interfaces and when communicating with large numbers of directly-connected peers. Default: 1024

gc_stale_time: Determines how often to check for stale neighbor entries. When a neighbor entry is considered stale, it is resolved again before sending data to it. Default: 60 (seconds)

gc_interval: How frequently the garbage collector for neighbor entries should attempt to run. Default: 30 (seconds)

base_reachable_time_ms: Once a neighbor has been found, the entry is considered to be valid for at least a random value between $base_reachable_time/2$ and $3*base_reachable_time/2$. An entry's validity will be extended if it receives positive feedback from higher level protocols. Default: 30000 (ms) (since since Linux 2.6.12 - *base_reachable_time*, which was defined in seconds, is obsolete)

mcast_solicit: The maximum number of attempts to resolve an address by multicast/broadcast before marking the entry as unreachable. Default: 3

ucast_solicit: When a neighbor entry becomes stale or questionable, Linux may first try to confirm it by sending unicast probes to the known MAC/link-layer address. The *ucast_solicit* option sets the number of those unicast attempts. If they fail, the kernel can proceed to multicast/broadcast-style neighbor discovery depending on the state and related parameters such as *mcast_solicit*.

Configuration Options to Avoid Network Related Expels

Here are the ARP tuning parameters that we found resolved stability issues on some of our largest deployments (the Coral systems):

```
net.ipv4.neigh.ib0.gc_thresh1=30000
net.ipv4.neigh.ib0.gc_thresh2=32000
net.ipv4.neigh.ib0.gc_thresh3=32768
net.ipv4.neigh.ib0.gc_stale_time=2000000
net.ipv4.neigh.ib0.gc_interval=2000000
net.ipv4.neigh.ib0.base_reachable_time_ms=120000
net.ipv4.neigh.ib0.mcast_solicit=18
```

Other large systems in the field have seen improvements applying slightly modified settings:

```
net.ipv4.neigh.ib0.gc_thresh1=30000
net.ipv4.neigh.ib0.gc_thresh2=32000
net.ipv4.neigh.ib0.gc_thresh3=32768
net.ipv4.neigh.ib0.gc_stale_time=864000
net.ipv4.neigh.ib0.gc_interval=864000
net.ipv4.neigh.ib0.mcast_solicit=9
```

`net.ipv4.neigh.default.base_reachable_time_ms` has not been tuned in these modified tunings because we to allow ARP entries to grow stale in a timely manner so we can detect if an entry is no longer valid (e.g. due to a LID change)

Added for 2026 London User group:

For some customers experiencing expels due to hitting the "shorter-timeouts" related to RPC-timeout related node requested expels, IBM has suggested increasing the **ucast_solicit** sysctl parameter to 9.

ucast_solicit can extend the reconfirmation period for an existing neighbour entry whose MAC address is already known. This is different from `mcast_solicit`, which controls how many ARP probes are sent while resolving an unknown neighbour.

Storage Scale Education Links

March 2026 Storage Scale Performance Debugging and Tuning Class



<https://ibm.box.com/s/k37git8ev6ipsajvhlcvxj8ukqo1n0o3>

Sept. 2024 Storage Scale Network Topics Presentation



<https://www.spectrumscaleug.org/wp-content/uploads/2025/04/SSUG24NYC11-Storage-Scale-Network-Topics-NYC.pdf>

Thanks!