From CephFS to Spectrum Scale

Sean Crosby

Research Computing Services

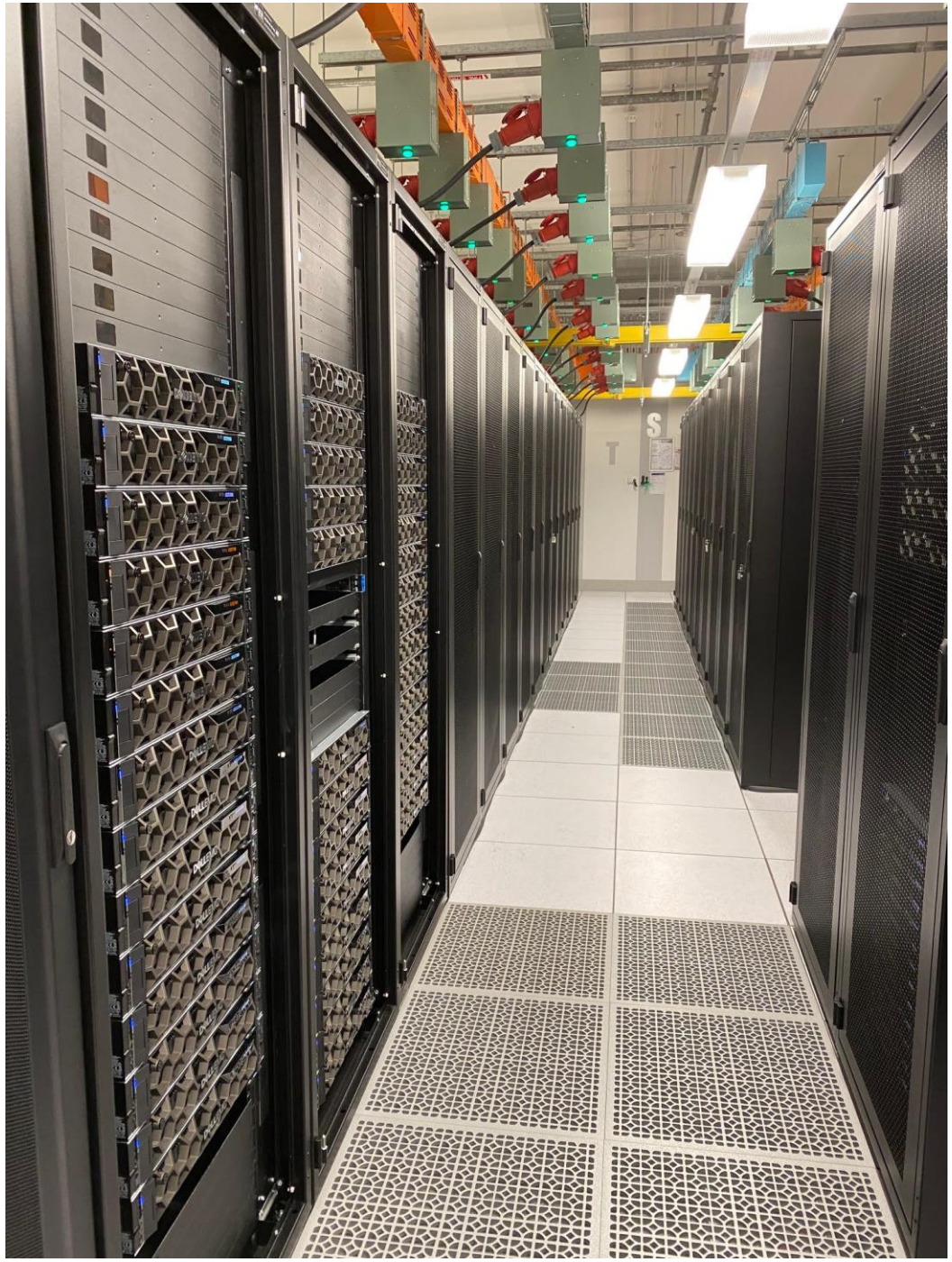
University of Melbourne

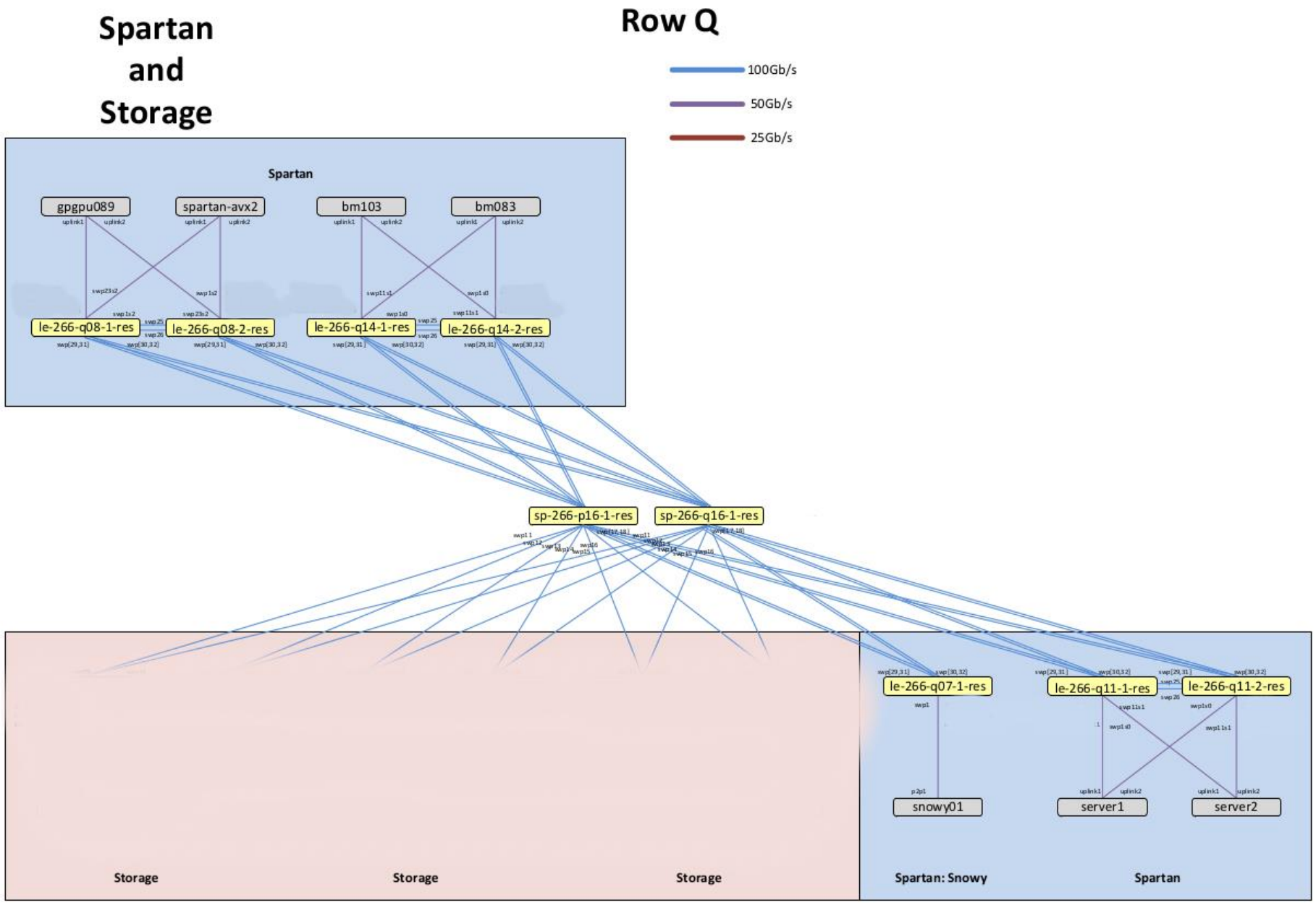
Our HPC site

- Spartan is our HPC system, a catchall HPC service for all researchers at the University
- Started in 2015 as a cloud/physical hybrid – majority of cores came from spare cycles on the NeCTAR Research Cloud. Physical nodes were purchased by research groups at the Uni for dedicated use.
- Filesystem was Netapp NFS
- Moved to CephFS for 3 reasons
  - Running out of space and maintenance on Netapp
  - Cloud team had experience with Ceph as object store – FS can't be too hard?
  - Uni won a LIEF grant for 77 GPGPU nodes

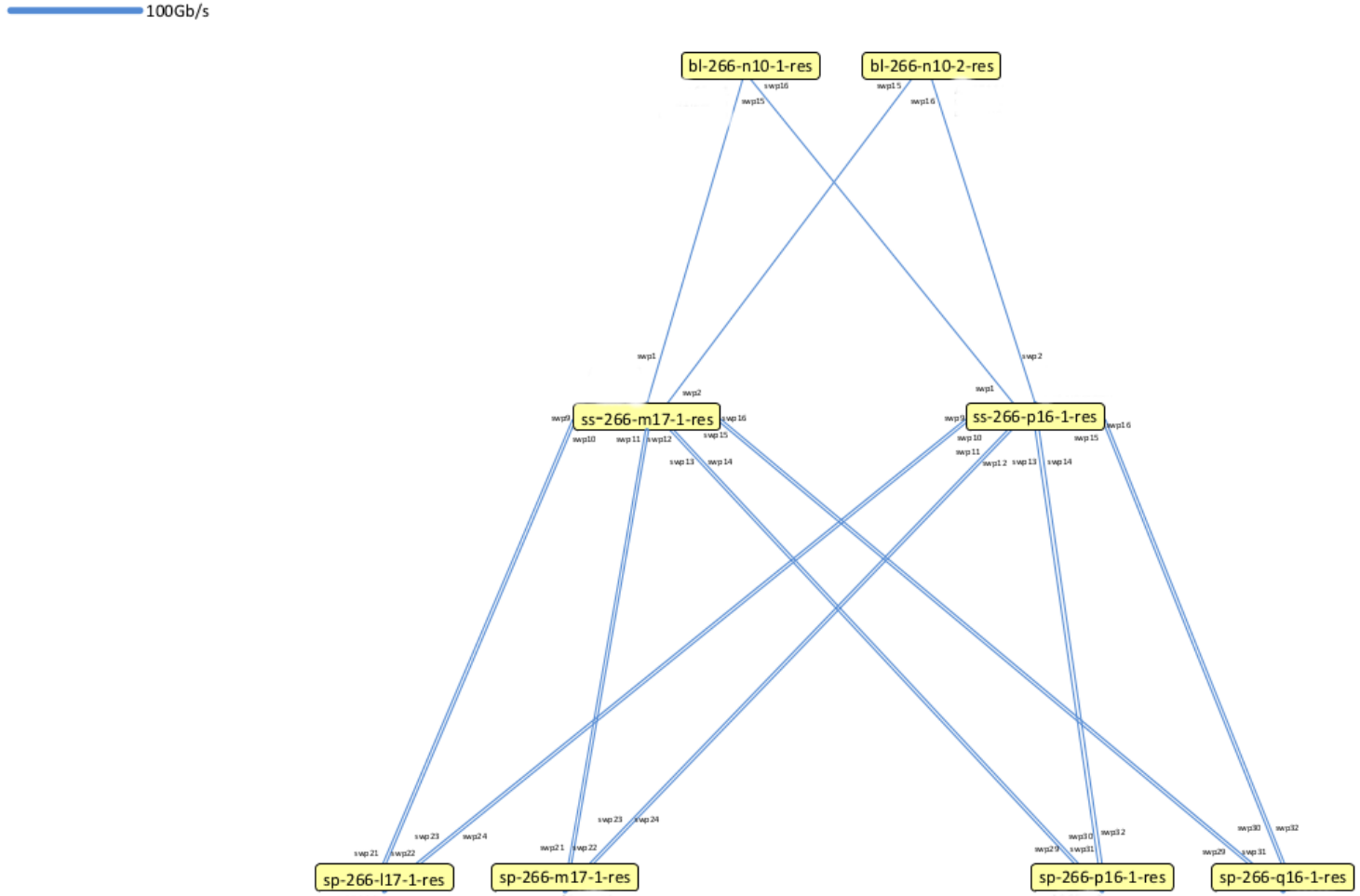
Our HPC site

- 90 GPU nodes (360 P100/V100 GPUs) – Dell C4130 – Single connected 100Gb
- 100 CPU nodes (24/32/72 core) – Dell R840 – Dual connected LACP 50Gb
- Mellanox SN2700/SN3700 leafs in superspine/spine/leaf configuration running Cumulus 4.1




Spartan and Storage
Row Q
100Gb/s
50Gb/s
25Gb/s
Spartan
gpgpu089
spartan-avx2
bm103
bm083
le-266-q08-1-res
le-266-q08-2-res
le-266-q14-1-res
le-266-q14-2-res
sp-266-p16-1-res
sp-266-q16-1-res
le-266-q07-1-res
le-266-q11-1-res
le-266-q11-2-res
snowy01
server1
server2
Storage
Storage
Storage
Spartan: Snowy
Spartan

SuperSpine Layer

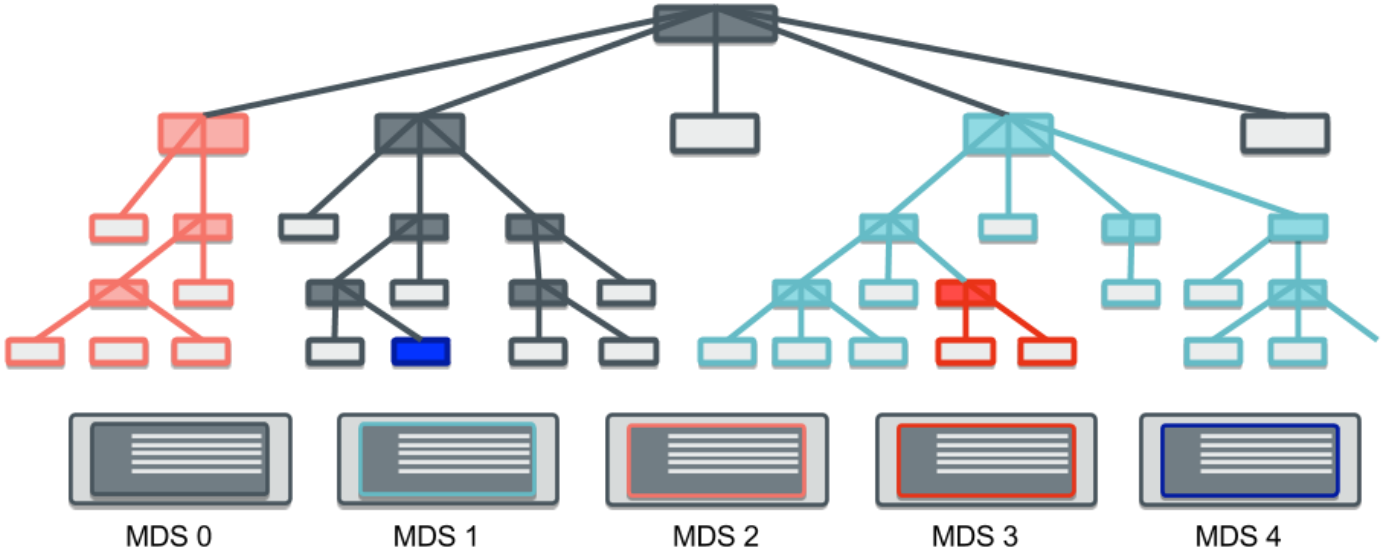


CephFS

- Filesystem on top of Ceph object storage
- Ceph has monitors and OSDs
  - Monitors keep track of state of cluster and quorum
  - OSDs are the object storage devices
    - Normally 1 OSD per physical drive
    - Data stored in either replication (typically 3), or erasure coding (typically 4+2)
    - When OSD is unavailable, system will replicate data on that OSD to free OSDs to recover system
  - Very stable
- CephFS adds metadata servers to provide the filesystem
  - Store metadata in either same pool as data or dedicated
  - Grants/revokes capabilities (caps) on metadata and data of inodes, and locks on inodes
  - Single threaded
  - Was “not supported”/experimental until a few years ago

CephFS

- Scaling CephFS
  - Single threaded means many inode/metadata updates can be slow
  - Multiple metadata servers
    - Active/backup – won't help with speed, but is helpful for availability
    - Multi active – split directory tree between active members



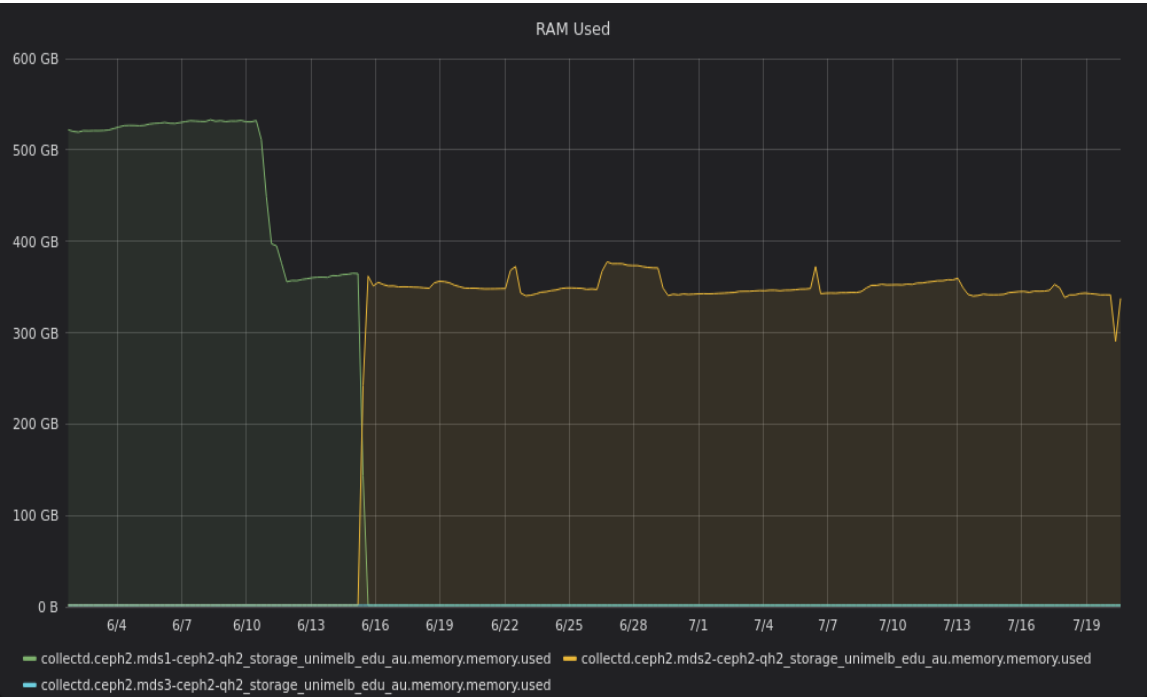
Storage issues with Spartan

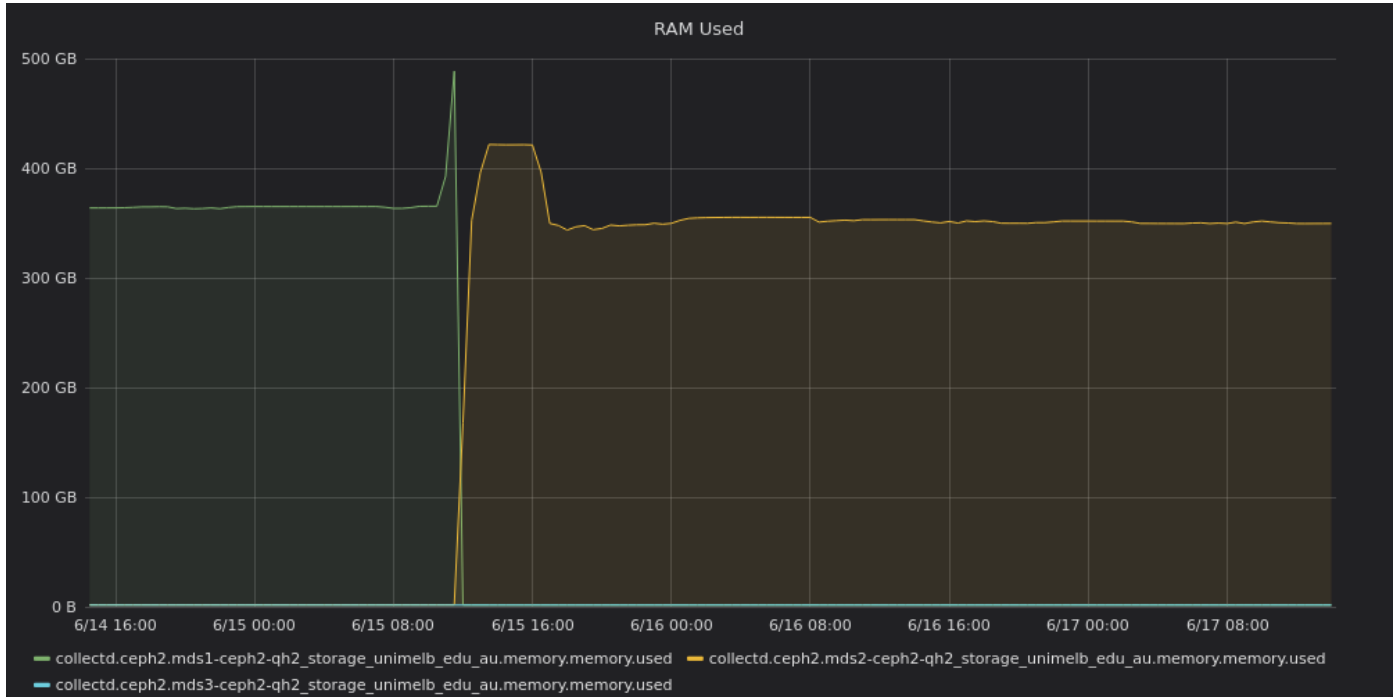
- July 2018
  - Monitors couldn’t contact a few OSD hosts, so replication recovery began
  - More OSDs started to not be contactable, in a rolling fashion. Monitors were so loaded with recovery calculations they crashed.
  - Monitors brought back up, but filesystem still not accessible
  - Online guide to filesystem recovery run which brought cluster back online
  - 2 days later same network problem occurred, and same steps run. Cluster back online.
  - 2 days later MDS crashed with inode uniqueness error. Files were trying to be written with same inode number as existing files. Thought nothing of it – started MDS. Cluster back online.

Storage issues with Spartan

- July 2018
  - MDS crashed again with inode uniqueness problem
  - Rechecked online guide. One of the commands suggested to run (inode session table zap) should only ever be run in certain circumstances.
  - Did a full filesystem scan (3 days) and then filesystem back up and stable
- Memory pressure
  - Our MDS servers had 512GB RAM. MDS memory usage due to capabilities given to nodes. MDS when getting low in RAM should request caps to be released, freeing RAM. Memory usage normally around 460GB, with spikes every few weeks causing MDS to crash, and either starting again and picking up where it was before, or standby MDS taking over. Normally periods of 10mins where IO was stuck and MDS was recovering.

Storage issues with Spartan





Storage issues with Spartan

- GPGPU workload causing MDS slowness
  - From `ceph health detail`, IO ops on Spartan would be around 4-5k ops/sec.
  - A few datasets (clothing1M in particular) when run on multiple nodes, would cause IO ops/sec to spike at 90-100k ops/s, causing metadata slowness, and users would see simple interactive ops (ls, chown etc) hang
- Lack of monitoring
  - Simple ability to check which nodes were causing highest Ceph load (either ops/s, or bandwidth), and breakdown between multiple pools (we had 2 – one 10K SAS, and another Sandisk flash) was lacking with CephFS
  - Would have made tracking which jobs were causing the most CephFS issues so much easier
- Different mount method produces different functionality
  - CephFS offers two ways of mounting the filesystem – kernel or FUSE client
  - FUSE client – supports latest functionality, supports quotas,mmap(), but is much slower, and suffers from memory pressures as well
  - Kernel client – fastest, no quota support (in version we were using), needed newer kernel than stock EL7 kernel for most functionality (until Redhat bought Ceph and then they backported to stock EL7 kernel), no mmap() support
  - So either run in fastest mode and have no quotas, or slowest mode and have quotas. So we ran FUSE on login nodes, and kernel on worker nodes
  - Users running work on login node would cause the OOM killer to kill ceph-fuse, stopping filesystem on that node

Time for a new FS

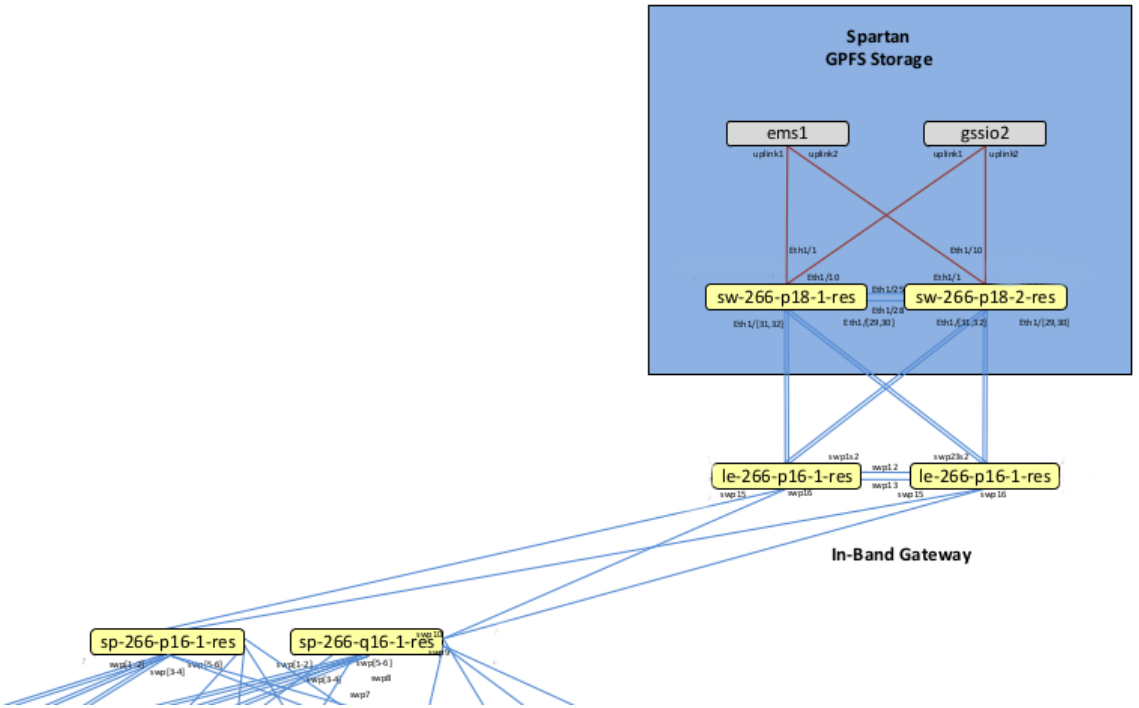
- Reliability, reliability, reliability
- RoCE used by jobs – why not have storage use it as well?
- Quota enforcement everywhere
- Currently at 5k CPU cores – will probably reach 10k CPU cores in next few years - need to guarantee minimum of 4MB/s throughput for all CPU cores.
- Snapshot support
- Single monitoring pane – IO throughput, quotas, node IO, system health
- 2PB spinning and 500TB flash, with ability to add more if required
- Reliability

Time for a new FS

- Responses included Lustre, BeeGFS, WekaIO, Spectrum Scale
- Based on price, requirements and references, Spectrum Scale from Advent One/IBM was chosen
- GH14s, EMS node, 3 protocol nodes and 3 ESS 3000
- Single point of contact for hardware and software support, as well as no capacity license was a huge factor for us
- Proceeded to POC

Spectrum Scale POC

- GH14s installed in our datacentre and configured by IBM
- Both IO nodes connected with 6x100Gb QSFP28
- Originally running 5.0.4-1
- Aim was compatibility with environment, RoCE, performance and functional tests
- Functional – GUI, mmap(), quotas, most common apps on CephFS
- Performance – IO500 10 node 160thread


Spartan
GPFS Storage
ems1
gssio2
sw-266-p18-1-res
sw-266-p18-2-res
le-266-p16-1-res
le-266-p16-1-res
In-Band Gateway
sp-266-p16-1-res
sp-266-q16-1-res


33
31
29
27
25
20
15
13
11
0
1
IBMR1


30
34
32
IBMR2

RoCE

- RDMA over converged ethernet
- Uses explicit congestion notification (ECN) and priority flow control (PFC) to allow Infiniband verbs to be carried over ethernet in a lossless fashion
- Have been using it for 2 years so far
- OpenMPI – openib BTL, rdmacm, UCX PML

```
traffic.cos_2.priority_source.dscp = [48]
traffic.cos_3.priority_source.dscp = [26]

traffic.priority_group_list = [control, service, bulk]
priority_group.control.cos_list = [2]
priority_group.service.cos_list = [3]
priority_group.bulk.cos_list = [0,1,4,5,6,7]
priority_group.control.weight = 0
priority_group.service.weight = 16
priority_group.bulk.weight = 16

ecn_red.port_group_list = [ROCE_ECN]

ecn_red.ROCE_ECN.port_set = ...
ecn_red.ROCE_ECN.cos_list = [3]
ecn_red.ROCE_ECN.ecn_enable = true
ecn_red.ROCE_ECN.red_enable = true

pfc.port_group_list = [ROCE_PFC]

pfc.ROCE_PFC.port_set = ...
pfc.ROCE_PFC.cos_list = [3]
pfc.ROCE_PFC.port_buffer_bytes = 70000
```

le-266-p06-1-res

spartan-gpgpu047	spartan-gpgpu048	:	6352.926928 MB/sec	1.4055 usec	
spartan-gpgpu049	spartan-gpgpu050	:	6357.761073 MB/sec	1.367001 usec	
spartan-gpgpu051	spartan-gpgpu052	:	6354.948283 MB/sec	1.3605 usec	
spartan-gpgpu053	spartan-gpgpu054	:	6352.022526 MB/sec	1.3915 usec	
spartan-gpgpu055	spartan-gpgpu056	:	6352.503554 MB/sec	1.385 usec	Avg Lat 1.3738077692 usec
spartan-gpgpu057	spartan-gpgpu058	:	6355.217911 MB/sec	1.3925 usec	Max Lat 1.4055 usec
spartan-gpgpu059	spartan-gpgpu060	:	6358.898474 MB/sec	1.357 usec	Min Lat 1.3455 usec
spartan-gpgpu061	spartan-gpgpu062	:	6359.01418 MB/sec	1.3635 usec	Avg Trans rate 6355.6768387 MB/sec
spartan-gpgpu063	spartan-gpgpu064	:	6358.01169 MB/sec	1.386499 usec	Max Trans rate 6359.01418 MB/sec
spartan-gpgpu065	spartan-gpgpu066	:	6355.911299 MB/sec	1.3595 usec	Min Trans rate 6351.21456 MB/sec
spartan-gpgpu067	spartan-gpgpu068	:	6351.21456 MB/sec	1.3455 usec	
spartan-gpgpu069	spartan-gpgpu070	:	6356.50853 MB/sec	1.387501 usec	
spartan-gpgpu047	spartan-gpgpu071	:	6358.859895 MB/sec	1.358 usec	

le-266-q14-1-res

spartan-bm083	spartan-bm084	:	5805.329354 MB/sec	1.6505 usec	
spartan-bm085	spartan-bm086	:	5816.358397 MB/sec	1.606 usec	
spartan-bm087	spartan-bm088	:	5815.342314 MB/sec	1.631501 usec	
spartan-bm089	spartan-bm090	:	5812.31225 MB/sec	1.613 usec	
spartan-bm091	spartan-bm092	:	5806.743109 MB/sec	1.580978 usec	Avg Lat 1.5947422727 usec
spartan-bm093	spartan-bm094	:	5812.134978 MB/sec	1.613458 usec	Max Lat 1.6505 usec
spartan-bm095	spartan-bm096	:	5811.878085 MB/sec	1.565029 usec	Min Lat 1.565029 usec
spartan-bm097	spartan-bm098	:	5815.89159 MB/sec	1.59 usec	Avg Trans rate 5810.1659998 MB/sec
spartan-bm099	spartan-bm100	:	5806.759955 MB/sec	1.570035 usec	Max Trans rate 5820.459723 MB/sec
spartan-bm101	spartan-bm102	:	5809.396576 MB/sec	1.570501 usec	Min Trans rate 5768.299235 MB/sec
spartan-bm103	spartan-bm104	:	5811.394362 MB/sec	1.602981 usec	
spartan-bm105	spartan-bm106	:	5820.459723 MB/sec	1.580978 usec	
spartan-bm107	spartan-bm108	:	5810.23413 MB/sec	1.601467 usec	
spartan-bm109	spartan-bm110	:	5810.267862 MB/sec	1.597509 usec	
spartan-bm111	spartan-bm112	:	5813.247195 MB/sec	1.574459 usec	
spartan-bm113	spartan-bm114	:	5768.299235 MB/sec	1.618988 usec	
spartan-bm115	spartan-bm116	:	5805.972055 MB/sec	1.589477 usec	
spartan-bm117	spartan-bm118	:	5806.26024 MB/sec	1.59099 usec	
spartan-bm119	spartan-bm120	:	5817.809295 MB/sec	1.576496 usec	
spartan-bm121	spartan-bm122	:	5819.359784 MB/sec	1.605484 usec	
spartan-bm123	spartan-bm124	:	5814.133006 MB/sec	1.583001 usec	
spartan-bm124	spartan-bm125	:	5814.068501 MB/sec	1.571498 usec	

spartan-gpgpu023 spartan-gpgpu024 : 6235.214381 MB/sec 2.3965 usec

RoCE – IO500 and GPFS

- To start with, GPFS was not working with RoCE due to vlan for GPFS not being native (Dale's talk)

```
[RESULT-invalid] BW   phase 1            ior_easy_write              15.640 GB/s : time 199.82 seconds
[RESULT] BW   phase 2            ior_hard_write               0.075 GB/s : time 2340.00 seconds
[RESULT] BW   phase 3             ior_easy_read              24.648 GB/s : time 126.78 seconds
[RESULT] BW   phase 4             ior_hard_read               1.245 GB/s : time 140.71 seconds
[RESULT-invalid] IOPS phase 1         mdtest_easy_write              68.010 kiops : time  42.65 seconds
[RESULT] IOPS phase 2          mdtest_hard_write              12.649 kiops : time 374.80 seconds
[RESULT] IOPS phase 3                       find              58.420 kiops : time 115.50 seconds
[RESULT] IOPS phase 4          mdtest_easy_stat              60.136 kiops : time  35.21 seconds
[RESULT] IOPS phase 5          mdtest_hard_stat              52.927 kiops : time  90.55 seconds
[RESULT] IOPS phase 6        mdtest_easy_delete              31.949 kiops : time  69.23 seconds
[RESULT] IOPS phase 7          mdtest_hard_read              40.132 kiops : time 119.28 seconds
[RESULT] IOPS phase 8        mdtest_hard_delete              10.861 kiops : time 446.11 seconds
```

RoCE – IO500 and GPFS

- Enabling RoCE

```
[RESULT-invalid] BW   phase 1            ior_easy_write              17.914 GB/s : time 174.44
seconds
[RESULT] BW   phase 2            ior_hard_write               0.249 GB/s : time 702.60 seconds
[RESULT] BW   phase 3             ior_easy_read              28.577 GB/s : time 109.35 seconds
[RESULT] BW   phase 4             ior_hard_read               2.740 GB/s : time  63.91 seconds
[RESULT-invalid] IOPS phase 1         mdtest_easy_write             127.714 kiops : time  17.51
seconds
[RESULT] IOPS phase 2          mdtest_hard_write              13.989 kiops : time 382.92 seconds
[RESULT] IOPS phase 3                       find             103.780 kiops : time  70.76 seconds
[RESULT] IOPS phase 4          mdtest_easy_stat             139.283 kiops : time  15.59 seconds
[RESULT] IOPS phase 5          mdtest_hard_stat             138.733 kiops : time  40.20 seconds
[RESULT] IOPS phase 6        mdtest_easy_delete              69.651 kiops : time  38.51 seconds
[RESULT] IOPS phase 7          mdtest_hard_read             131.654 kiops : time  41.64 seconds
[RESULT] IOPS phase 8        mdtest_hard_delete              17.000 kiops : time 324.98 seconds
```

- All results are faster, but especially mdtest
- Latency is much lower when RoCE enabled (approx. 1.6us)

GPFS

```
[RESULT-invalid] BW   phase 1            ior_easy_write              17.914 GB/s : time 174.44
seconds
[RESULT] BW   phase 2            ior_hard_write               0.249 GB/s : time 702.60 seconds
[RESULT] BW   phase 3            ior_easy_read               28.577 GB/s : time 109.35 seconds
[RESULT] BW   phase 4            ior_hard_read                2.740 GB/s : time  63.91 seconds
[RESULT-invalid] IOPS phase 1         mdtest_easy_write             127.714 kiops : time  17.51
seconds
[RESULT] IOPS phase 2         mdtest_hard_write              13.989 kiops : time 382.92 seconds
[RESULT] IOPS phase 3                      find              103.780 kiops : time  70.76 seconds
[RESULT] IOPS phase 4         mdtest_easy_stat              139.283 kiops : time  15.59 seconds
[RESULT] IOPS phase 5         mdtest_hard_stat              138.733 kiops : time  40.20 seconds
[RESULT] IOPS phase 6       mdtest_easy_delete               69.651 kiops : time  38.51 seconds
[RESULT] IOPS phase 7         mdtest_hard_read              131.654 kiops : time  41.64 seconds
[RESULT] IOPS phase 8       mdtest_hard_delete               17.000 kiops : time 324.98 seconds
```

CephFS

```
[RESULT] BW   phase 1         ior_easy_write          6.283 GB/s : time 461.65 seconds
[RESULT] BW   phase 2         ior_hard_write          0.156 GB/s : time 1122.71 seconds
[RESULT] BW   phase 3         ior_easy_read          41.064 GB/s : time  70.63 seconds
[RESULT] BW   phase 4         ior_hard_read           0.359 GB/s : time 487.15 seconds
[RESULT-invalid] IOPS phase 1       mdtest_easy_write          6.872 kiops : time 292.20
seconds
[RESULT] IOPS phase 2       mdtest_hard_write          6.762 kiops : time 318.28 seconds
[RESULT] IOPS phase 3                    find        131.290 kiops : time  31.58 seconds
[RESULT] IOPS phase 4       mdtest_easy_stat         18.632 kiops : time 108.67 seconds
[RESULT] IOPS phase 5       mdtest_hard_stat         15.759 kiops : time 137.41 seconds
[RESULT] IOPS phase 6     mdtest_easy_delete          4.451 kiops : time 450.88 seconds
[RESULT] IOPS phase 7       mdtest_hard_read          3.958 kiops : time 543.12 seconds
[RESULT] IOPS phase 8     mdtest_hard_delete          4.577 kiops : time 470.17 seconds
```

Road to production

- Data migration
  - CephFS had 1.2PB data, and 1250 top level directories
  - 3 simultaneous rsync running on 8 nodes, starting from Monday every week for 6 weeks
  - Average of 1.5 days to get into sync
  - Any more rsyncs caused CephFS MDS to OOM – kind of validates our need to move to new FS
  - No major issues seen
- Flash tier
  - 150TB flash tier put as default pool in front of SAS pool
- Go live
  - 3 day maintenance window – OS, OFED and Cumulus update
  - Unmount CephFS everywhere, add Spectrum Scale
  - Finish data migration
  - Finished on time, and a full 1 day of additional testing

Road to production

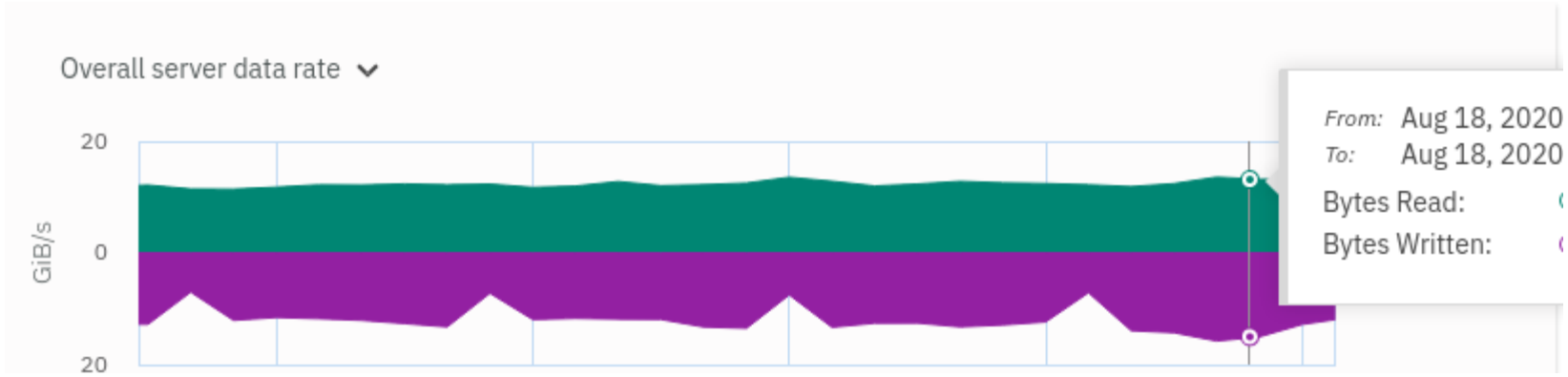
- Go live
  - Within 15 mins of opening login nodes to users they started crashing and rebooting
  - /var/crash showed segfault in setacl GPFS routine

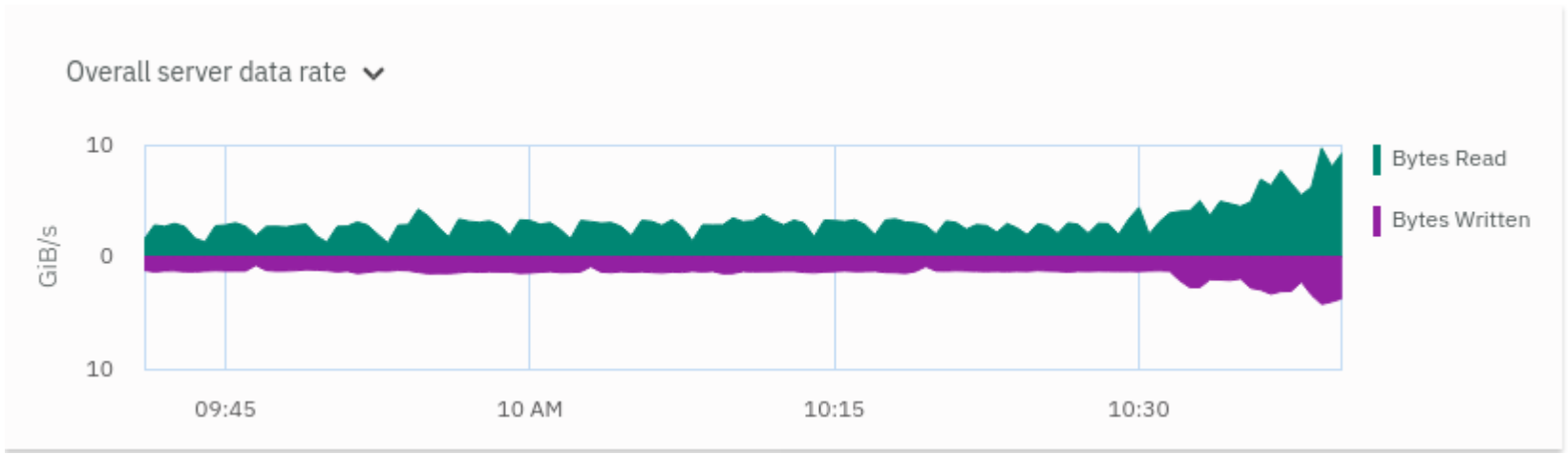
```
 1642.006978] Call Trace:
[ 1642.009437]  [<ffffffff97ab7742>] ? posix_acl_from_xattr+0x82/0x190
[ 1642.015722]  [<ffffffffc114cc9d>] gpfs_set_posix_acl+0xad/0x330 [mmfslinux]
[ 1642.022699]  [<ffffffffc114e055>] ? gpfs_get_posix_acl+0x1a5/0x330 [mmfslinux]
[ 1642.029931]  [<ffffffff979ee5e9>] ? do_read_fault.isra.63+0x139/0x1b0
[ 1642.036388]  [<ffffffffc114d064>] gpfs_i_setxattr+0x144/0x610 [mmfslinux]
[ 1642.043190]  [<ffffffff97b8c252>] ? radix_tree_lookup_slot+0x22/0x50
[ 1642.049551]  [<ffffffff979c0e9d>] ? filemap_fault+0x17d/0x420
[ 1642.055310]  [<ffffffffc114e34b>] ? gpfs_i_getxattr+0x16b/0x6c0 [mmfslinux]
[ 1642.062310]  [<ffffffffc07a9374>] ? xfs_iunlock+0x114/0x120 [xfs]
[ 1642.068433]  [<ffffffffc079a99e>] ? __xfs_filemap_fault+0x8e/0x1d0 [xfs]
```

  - Never occurred in 2 months of testing with the same kernel/OFED/gpfs packages – of course users trigger it ☺
  - Were running 5.0.4-3, fixed in 5.0.4-4

3 weeks in

- Users asked for feedback
  - We know I/O should be MUCH better on login node – only 1 commented
  - Can give out more quota, which is what users like
- Love the GUI
  - My 3x daily monitoring page
  - Acts like Nagios too – get emails sometimes from GPFS before Nagios picks it up
- Spectrum Discover
  - We have SD doing scans, and will become useful, especially to identify users with the same dataset that can be put into shared area
- Policy engine
  - Used to find core.XXXX files and delete





spartan-gpgpu086-hs.ss.stor...	✓ Healthy	15%	3.18	38.8%	0 bytes/s	961 MiB/s
spartan-gpgpu080-hs.ss.stor...	✓ Healthy	15.6%	3.86	70.5%	1.97 MiB/s	896 MiB/s

scrosby@unimelb.edu.au