Mixing storage systems in Spectrum Scale

Migrations and pools stories

Luis Bolinches

Disclaimers

- This is a personal view, opinions are my own, not IBM.
- I am usually wrong, which makes me learn a lot along the way
- Please ask whenever during the presentation, first one is free

What I would like to talk about in this session

- What to look after when migrating multiple storage subsystems (MSS) in the same filesystem:
  - In the storage subsystems
  - In the Spectrum Scale filesystem
- MSS can happen when using multiple pools or doing storage migrations
- Reasons why I think Quality of Service (QoS) rocks
- Have some moderate fun

What I would not like to talk about in this session*

- Filesystems migrations
  - Filesystem migrations to 5.0.0
  - AFM
- Protocol Nodes
- GUI
- Subblocks on 5.0.0
- ESS

* But I guess we will end up talking about it anyway

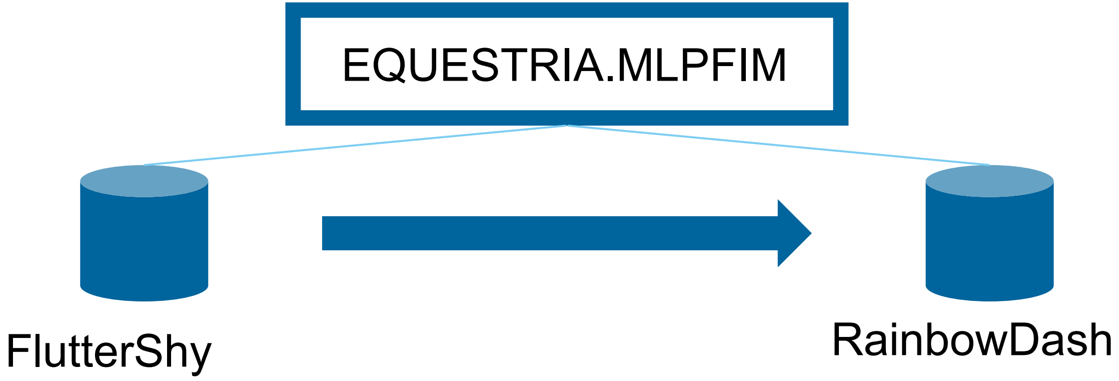
Lingo matters

- The blocksize is the largest size IO that Spectrum Scale will issue to the storage device
- A subblock is the smallest allocation of a file in a Spectrum Scale filesystem
  - 1/32 before 5.0.0 (My personal ”before 5.0.0” starts on 3.2)
  - Changes on 5.0.0
- Sector is the smallest IO request size it can issued to the storage device
- RAID stripe is the amount of data on a segment size x number of usable capacity data disks
- RAID segment size is the amount of data written to a single disk within a RAID stripe
- RAID Full Stripe Write: A mystical figure?

* Probably a Pegasus

The replacement case - overview

- 1 Spectrum Scale cluster (EQUESTRIA.MLPFIM)
- 3 Linux on Power 8 node cluster with shared storage (canterlot, ponyville and crystalempire)
- 1 filesystem created on 3.5 and migrated over the time to 5.0 (EQUESTRIA)
- Only one SAN type of storage used so far since day 1 (FlutterShy)
- New storage acquired based on multi level cell to replace old storage (RainbowDash)



The replacement case – cluster and filesystem

- Created on 4.2.3 with blocksize of 64KB
- Filesystem is not 4K aligned
- Only one internal pool (system)
- Dedicated NSD for metadata
- No metadata nor metadata replication
- QoS is not enabled
- FlutterShy is using 512 sector size
- FlutterShy using RAID10 and 16 MB extents
- RainbowDash configured to provide 512 bytes vdisks
- RainbowDash allows to create vdisks on 512 or 4096 bytes as the filesystem is not 4K aligned is not possible to add any NSD of 4096 bytes sector size

The replacement case – The migration steps

- Add the NSD from RainbowDash to the EQUESTRIA filesystem
- Suspend the disks from FlutterShy
- Enable QoS for maintenance
- Delete NSD FlutterShy disks

The replacement case – mmlsfs EQUESTRIA / mmlsqos EQUESTRIA

```
[root@canterlot migration_case]# mmlsfs EQUESTRIA -f -i -B --subblocks-per-full-block --is4KAligned
flag                value                    description
------------------- ------------------------ -----------------------------------
 --subblocks-per-full-block 32               Number of subblocks per full block
 --is4KAligned      No                       is4KAligned?
 -f                 2048                     Minimum fragment (subblock) size in bytes
 -i                 4096                     Inode size in bytes
 -B                 65536                    Block size
```

```
[root@canterlot ~]# mmlsqos EQUESTRIA
QOS config::             disabled
QOS status::             throttling inactive, monitoring inactive
```

The replacement case – mmdf EQUESTRIA

```
[root@canterlot ~]# mmdf EQUESTRIA
disk                    disk size  failure holds    holds              free KB             free KB
name                        in KB    group metadata data        in full blocks        in fragments
--------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: system (Maximum disk size allowed is 4.0 TB)
EQUESTRIA_MD001       33554432       10 Yes      No          32416896 ( 97%)           100 ( 0%)
EQUESTRIA_MD002       33554432       10 Yes      No          32416576 ( 97%)            60 ( 0%)
EQUESTRIA_MD003       33554432       10 Yes      No          32416832 ( 97%)           100 ( 0%)
EQUESTRIA_FlutterShy001     536870912       10 No       Yes         536805184 (100%)            60 ( 0%)
EQUESTRIA_FlutterShy002     536870912       10 No       Yes         536805184 (100%)            60 ( 0%)
EQUESTRIA_FlutterShy003     536870912       10 No       Yes         536805184 (100%)            60 ( 0%)
                                                 -------------                      -------------------- ---
----------------
(pool total)       1711276032                            1707665856 (100%)           440 ( 0%)
```

The replacement case – Adding RainbowDash

```
[root@canterlot migration_case]# mmlsdisk EQUESTRIA
disk         driver   sector     failure holds    holds                            storage
name         type       size       group metadata data  status        availability pool
------------ -------- ------ ----------- -------- ----- ------------- ------------ ------------
EQUESTRIA_MD001 nsd         512          10 Yes      No    ready         up           system
EQUESTRIA_MD002 nsd         512          10 Yes      No    ready         up           system
EQUESTRIA_MD003 nsd         512          10 Yes      No    ready         up           system
EQUESTRIA_FlutterShy001 nsd         512          10 No       Yes   ready         up           system
EQUESTRIA_FlutterShy002 nsd         512          10 No       Yes   ready         up           system
EQUESTRIA_FlutterShy003 nsd         512          10 No       Yes   ready         up           system
EQUESTRIA_RainbowDash_MD001 nsd         512          11 Yes      No    ready         up           system
EQUESTRIA_RainbowDash_MD002 nsd         512          11 Yes      No    ready         up           system
EQUESTRIA_RainbowDash_MD003 nsd         512          11 Yes      No    ready         up           system
EQUESTRIA_RainbowDash001 nsd         512          11 No       Yes   ready         up           system
EQUESTRIA_RainbowDash002 nsd         512          11 No       Yes   ready         up           system
EQUESTRIA_RainbowDash003 nsd         512          11 No       Yes   ready         up           system
```

The replacement case – Suspending FlutterShy

```
[root@canterlot migration_case]# mmchdisk EQUESTRIA suspend -F FS5K

[root@canterlot migration_case]# mmlsdisk EQUESTRIA
disk         driver   sector     failure holds    holds                            storage
name         type       size       group metadata data  status        availability pool
------------ -------- ------ ----------- -------- ----- ------------- ------------ ------------
EQUESTRIA_MD001 nsd         512          10 Yes      No    to be emptied up           system
EQUESTRIA_MD002 nsd         512          10 Yes      No    to be emptied up           system
EQUESTRIA_MD003 nsd         512          10 Yes      No    to be emptied up           system
EQUESTRIA_FlutterShy001 nsd         512          10 No       Yes   to be emptied up           system
EQUESTRIA_FlutterShy002 nsd         512          10 No       Yes   to be emptied up           system
EQUESTRIA_FlutterShy003 nsd         512          10 No       Yes   to be emptied up           system
EQUESTRIA_RainbowDash_MD001 nsd         512          11 Yes      No    ready         up           system
EQUESTRIA_RainbowDash_MD002 nsd         512          11 Yes      No    ready         up           system
EQUESTRIA_RainbowDash_MD003 nsd         512          11 Yes      No    ready         up           system
EQUESTRIA_RainbowDash001 nsd         512          11 No       Yes   ready         up           system
EQUESTRIA_RainbowDash002 nsd         512          11 No       Yes   ready         up           system
EQUESTRIA_RainbowDash003 nsd         512          11 No       Yes   ready         up           system
```

The replacement case – QoS

```
[root@canterlot EQUESTRIA]# mmchqos EQUESTRIA --enable
QOS configuration has been installed and broadcast to all nodes.

[root@canterlot EQUESTRIA]# mmlsqos EQUESTRIA
QOS config::     enabled
QOS values::     pool=system,other=inf,maintenance/all_local=inf
QOS status::     throttling active, monitoring active


[root@canterlot EQUESTRIA]# mmchqos EQUESTRIA --enable pool=system,maintenance=300IOPS,other=unlimited
QOS configuration has been installed and broadcast to all nodes.


[root@canterlot EQUESTRIA]# mmlsqos EQUESTRIA
QOS config::     enabled -- pool=system,other=inf,maintenance/all_local=300Iops
QOS status::     throttling active, monitoring active
```

The qualifier /all_local after maintenance indicates that the maintenance IOPS are applied to all the files systems owned by the cluster. This value is the default for the maintenance class.

The replacement case – QoS

```
[root@canterlot ~]# mmlsqos EQUESTRIA --seconds 10
QOS config::              enabled -- pool=system,other=inf,maintenance/all_local=300Iops
QOS status::              throttling active, monitoring active

#### for pool system
10:25:35  misc iops=33.2 ioql=0.017513 qsdl=5.95e-06 et=5
10:25:35 other iops=3076 ioql=51.192 qsdl=0.0003862 et=5
10:25:35 maint iops=300 ioql=0.31915 qsdl=46.877 et=5
10:25:40  misc iops=20.6 ioql=0.01032 qsdl=3.9615e-06 et=5
10:25:40 other iops=3299 ioql=55.527 qsdl=0.00038394 et=5
10:25:40 maint iops=300 ioql=0.24989 qsdl=47.739 et=5
```

IOQL is the average number of I/O requests in the class that are pending for reasons other than being queued by QoS

QSDL is the average number of I/O requests in the class that are queued by Qo

ET is the intetval in seconds during the measurament is made

The replacement case – QoS

- I particularly find fine-grained QoS output extremely helpful to understand what is going on in a filesystem. Not the only way to do this
- To enable with PID and 10 seconds of information in memory of the nodes

```
mmchqos EQUESTRIA --fine-stats 10 --pid-stats yes
```

- You can see the node doing the operation, the type of operation (R/W), the class, if it is sector, subblock, less than full block, full block operation, PID, …
- The output is script readable

```
Time, Class, Node, Iops, TotSctrs, Pool, Pid, RW, SctrI, AvgTm, SsTm, MinTm, MaxTm, AvgQd, SsQd
1524033666,misc,10.10.17.113,239,81184,system,14680078,R,2047,0.003057,0.000572,0.000718,0.013203,0.000000,0.000000
1524033666,misc,10.10.17.113,2,16,system,14680072,W,16,0.002058,0.000000,0.002046,0.002070,0.000001,0.000000
1524033666,misc,10.10.17.113,1,336,system,14680072,W,2047,0.006226,0.000000,0.006226,0.006226,0.000000,0.000000
1524033666,misc,10.10.17.113,1,64,system,13369351,W,2047,0.002510,0.000000,0.002510,0.002510,0.000000,0.000000
1524033666,maintenance,10.10.17.113,3,32,system,7,R,16,0.002468,0.000021,0.000542,0.006171,0.000002,0.000000
1524033666,maintenance,10.10.17.113,2,2024,system,7,R,2047,0.020098,0.000040,0.015647,0.024549,0.000001,0.000000
1524033666,maintenance,10.10.17.113,21,43008,system,7,R,2048,0.015679,0.000249,0.011681,0.027030,0.000001,0.000000
1524033666,maintenance,10.10.17.113,9,72,system,7,W,16,0.001176,0.000007,0.000271,0.002474,0.000002,0.000000
1524033666,maintenance,10.10.17.113,1,1960,system,7,W,2047,0.005770,0.000000,0.005770,0.005770,0.000003,0.000000
1524033666,maintenance,10.10.17.113,21,43008,system,7,W,2048,0.002386,0.000018,0.002083,0.006312,0.000001,0.000000
1524033667,maintenance,10.10.17.113,56,114688,system,7,R,2048,0.014973,0.000352,0.012016,0.026745,0.000523,0.000106
```

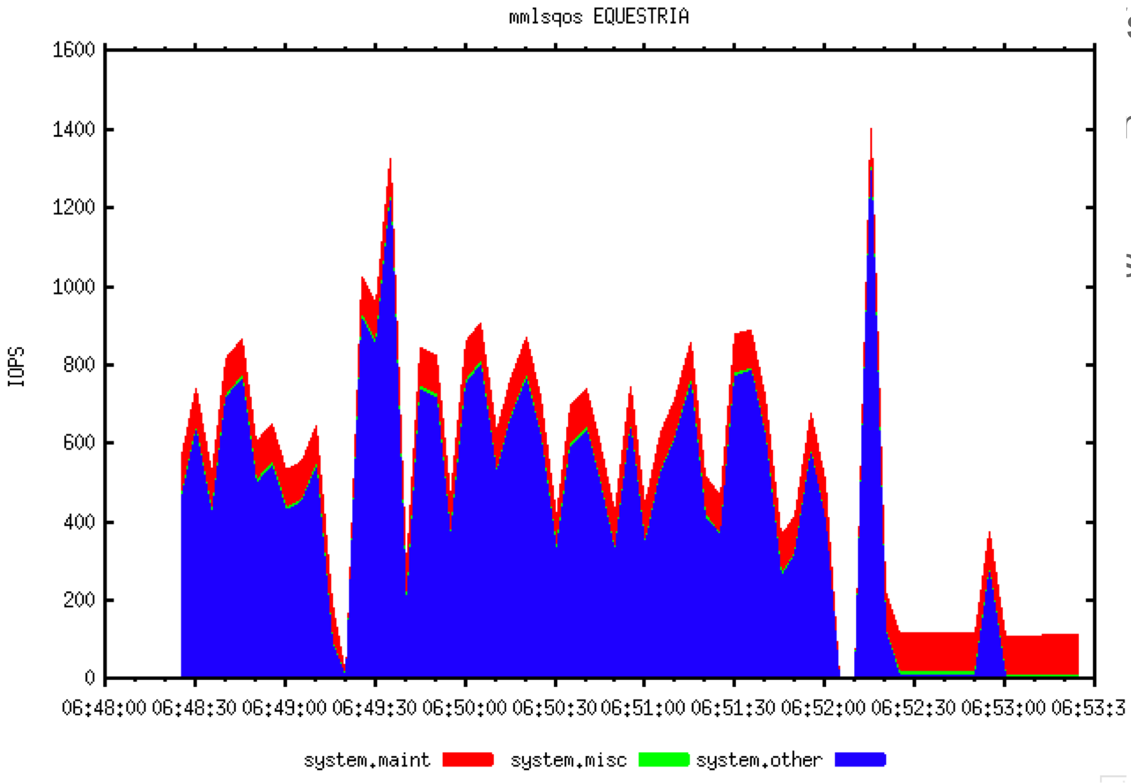
The replacement case – QoS

- We
  qosp
- To u
  cpan
  yum
- Let's
  qosp

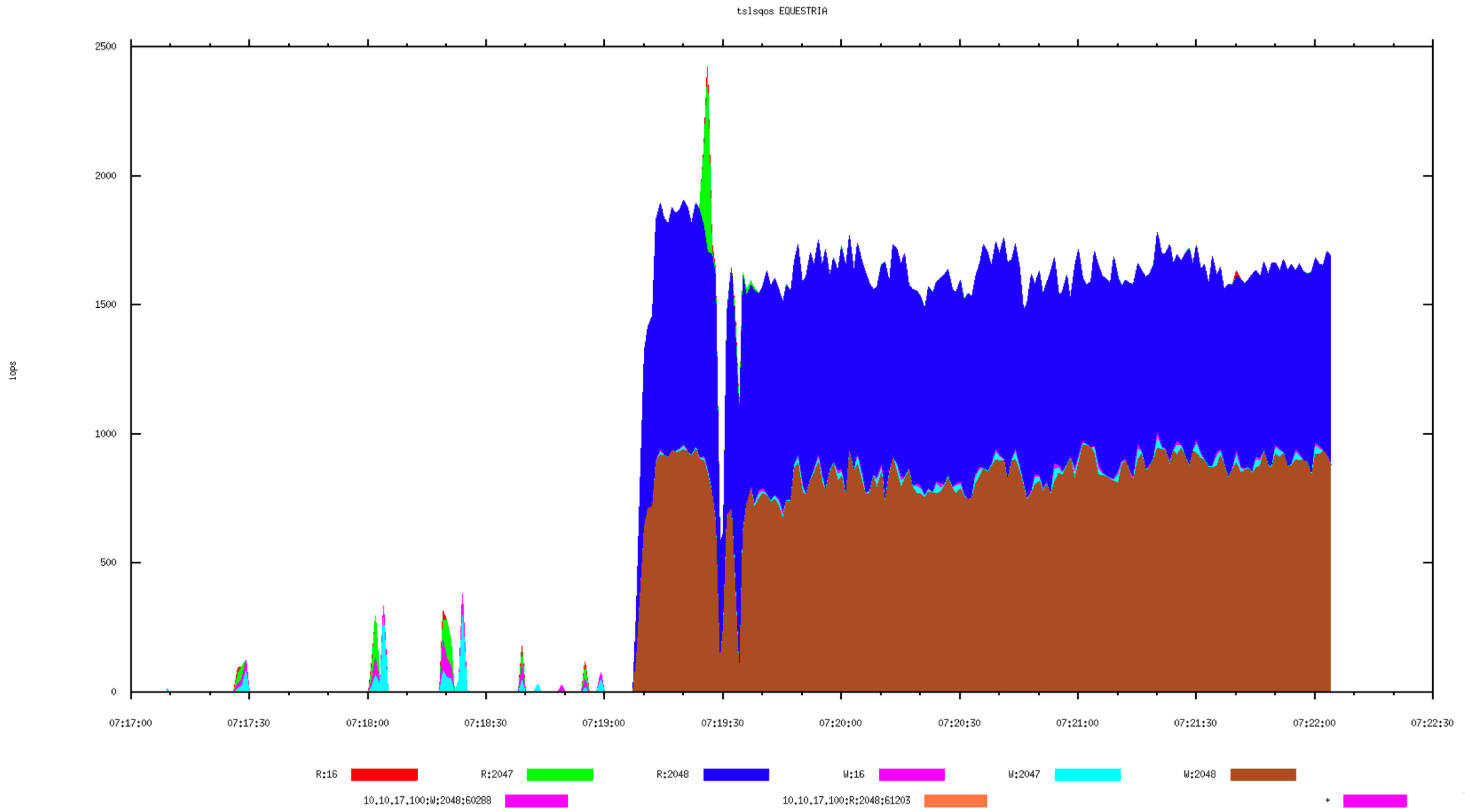
s/samples/charts

my base OS install

see only last 5 minutes



The replacement case – QoS



The replacement case – Bonnie++

- “**Bonnie++** is a free file system benchmarking tool for Unix-like operating systems, developed by Russell Coker. Bonnie++ is a benchmark suite that is aimed at performing a number of simple tests of hard drive and file system performance. ” Wikipedia

- The defaults were used here, so it creates only one file of 64GB on which performs its tests. That has a clear effect on the writes as Spectrum Scale does group writes for few seconds

- This is not the smartest way to test a workload I just used to create repeated ”noise” during the migration

- I find iozone more interesting for standalone clients, ior for multiple clients. It has been easier for me to compare results at customers done with neutral tools than the great gpfsperf

The replacement case – Removing FlutterShy

	Bonnie++	Removing disks
QoS	B min	62*D minutes
No QoS	6*B min	D minutes

	FlutterShy 5K	RainbowDash 900
Bonnie++ Write	W	4*W
Bonnie++ Read	R	35*R

The replacement case – cluster and filesystem

- Created on 5.0.1 with blocksize of 1MB
- Fylesystem is 4K aligned
- FlutterShy 5K is using 512 sector size
- FlutterShy 5K Gen 1 using RAID5 (8+P+S) and 64 MB extents
- Only one internal pool (system)
- Dedicated NSDs for metadata
- No metadata nor metadata replication

The replacement case – Creating a cluster on with intall toolkit on 5.0*

```
yum -y install ksh perl gcc kernel-devel imake compat-libstdc++ gcc-c++ redhat-lsb ntp net-tools
/usr/lpp/mmfs/5.0.0.1/installer/spectrumscale setup -s 10.10.17.10
/usr/lpp/mmfs/5.0.0.1/installer/spectrumscale node add -q -m -a -n canterlot
/usr/lpp/mmfs/5.0.0.1/installer/spectrumscale node add -q -m -a -n ponyville
/usr/lpp/mmfs/5.0.0.1/installer/spectrumscale node add -q -m -a -n crystalempire
/usr/lpp/mmfs/5.0.0.1/installer/spectrumscale config gpfs -c EQUESTRIA.MLPFIM -p randomio
/usr/lpp/mmfs/5.0.0.1/installer/spectrumscale fileauditlogging disable
/usr/lpp/mmfs/5.0.0.1/installer/spectrumscale callhome disable
/usr/lpp/mmfs/5.0.0.1/installer/spectrumscale config gpfs --ephemeral_port_range 60000-60010
/usr/lpp/mmfs/5.0.0.1/installer/spectrumscale config ntp -e on -s
0.centos.pool.ntp.org,1.centos.pool.ntp.org,2.centos.pool.ntp.org,3.centos.pool.ntp.org
/usr/lpp/mmfs/5.0.0.1/installer/spectrumscale install -pr
/usr/lpp/mmfs/5.0.0.1/installer/spectrumscale install
```

* Before for Protocols “edition” too

The replacement case – Creating a cluster on with intall toolkit on 5.0*

```
clusterName EQUESTRIA.MLPFIM
clusterId 13032611071871502107
autoload yes
profile gpfsProtocolRandomIO
dmapiFileHandleSize 32
minReleaseLevel 5.0.0.0
ccrEnabled yes
cipherList AUTHONLY
maxblocksize 16M
[cesNodes]
maxMBpS 5000
numaMemoryInterleave yes
enforceFilesetQuotaOnRoot yes
workerThreads 512
[common]
tscCmdPortRange 60000-60010
adminMode central
```

* Before for Protocols “edition” too

The replacement case – 1MB filesystem

```
[root@canterlot ~]# mmlsfs EQUESTRIA -f -i -B --subblocks-per-full-block --is4KAligned
flag                value                    description
------------------- ------------------------ -----------------------------------
 --subblocks-per-full-block 128              Number of subblocks per full block
 --is4KAligned      Yes                      is4KAligned?
 -f                 8192                     Minimum fragment (subblock) size in bytes
 -i                 4096                     Inode size in bytes
 -B                 1048576                  Block size


[root@canterlot ~]# mmlsqos EQUESTRIA
QOS config::             disabled
QOS status::             throttling inactive, monitoring inactive
```

The replacement case – Removing FlutterShy 1MB

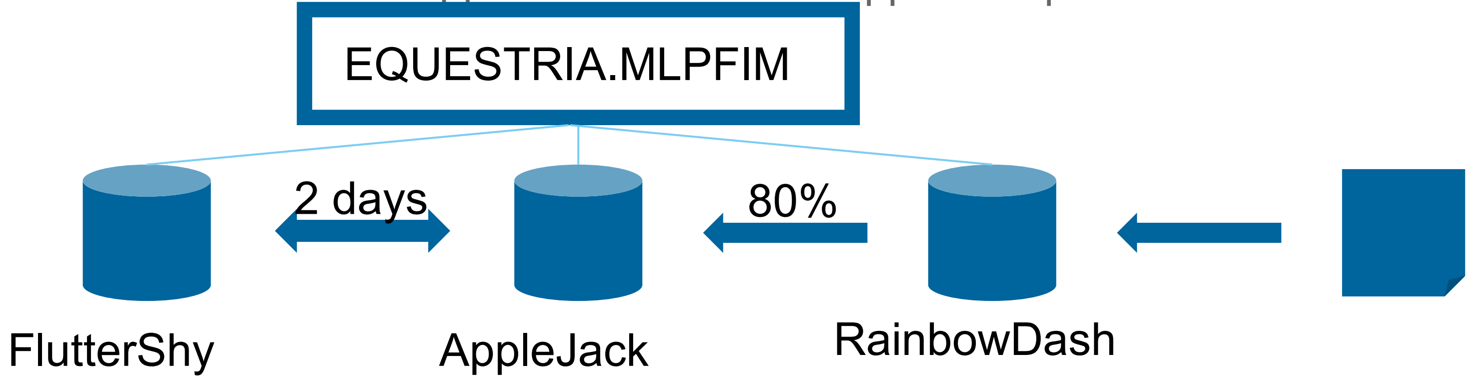
	Bonnie++ 1MB	Bonnie++ 64KB	Removing disks 1MB	Removing disks 64KB
QoS	0.8*B min	B min	42*D minutes	62*D minutes
No QoS	4.8*B min	6.2*B min	0.6*D minutes	D minutes
	FlutterShy 5K 1MB	FlutterShy 5K 64KB	RainbowDash 900 1MB	RainbowDash 900 64KB
Bonnie++ Write	W	W´	24*W	4*W´
Bonnie++ Read	R	R´	47*R	39*R´

The replacement case – Take aways

- Not noticeable change on performance on 4K sector vs 512 when already aligned
- Full stripe write did not have much impact in **these** storages on throughput and **this** test
  - Bonnie++ used one file only!
  - Sequential runs of bonnie++
- There was a x3 latency impact on latency when not full stripe write aligned
- mmdeldisk is robust (Please share your horror stories)
- But as far as source disks are suspended you can remove source disks one by one

The multi pool case – Introduction

- 1 Spectrum Scale cluster (EQUESTRIA.MLPFIM)
- 3 Linux node cluster with shared storage (canterlot, ponyville and crystalempire)
- 1 filesystem created on 3.5 and migrated over the time to 5.0 (EQUESTRIA)
- Blocksizes: Metadata 256KB, Data 1MB
- Dedicated disks from RainbowDash for metadata on system pool
- Dedicated disks from RainbowDash for data on RainbowDash pool
- Dedicated disks from FlutterShy for data on FlutterShy pool
- Dedicated disks from AppleJack for data on AppleJack pool



The multi pool case – Details on storage

- Move between pools is not much different from migration case
- RainbowDash provides 4K sector disks
- AppleJack (1MB RAID stripe)
- FlutterShy RAID 5 (8+P+S)

The multi pool case – Policies

```
[root@canterlot pools_case]# mmaddcallback MIGRATION --command /usr/lpp/mmfs/bin/mmstartpolicy --event
lowDiskSpace,noDiskSpace --parms "%eventName %fsName"
mmaddcallback: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

```
[root@canterlot pools_case]# mmlspolicy EQUESTRIA -L
RULE 'clean_RainbowDash' MIGRATE FROM POOL 'RainbowDash' THRESHOLD(80,40)
WEIGHT(KB_ALLOCATED)
TO POOL 'AppleJack'
RULE 'clean_AppleJack'
MIGRATE FROM POOL 'AppleJack'
WEIGHT(KB_ALLOCATED)
TO POOL 'FlutterShy'
WHERE (CURRENT_TIMESTAMP - ACCESS_TIME > INTERVAL '2' DAYS)
RULE 'upgrade_to_AppleJack'
MIGRATE FROM POOL 'FlutterShy'
WEIGHT(KB_ALLOCATED)
TO POOL 'AppleJack'
WHERE (CURRENT_TIMESTAMP - ACCESS_TIME < INTERVAL '2' DAYS)
RULE 'default' SET POOL 'RainbowDash'
```

The multi pool case – Policies

- The move between RainbowDash 900 and AppleJack IV would happen automatically when RainbowDash pool reaches 80% utilization or higher.

- The move between AppleJack IV and FlutterShy 5K back and forward would be a batch process that we can control when it happens

- So in addition to the same factors we need to look on the migration case we need to look into the speed of migration between pools. This is particularly important for the move between RainbowDash and AppleJack pools as if does not happen faster than the ingest of data it could overrun the pool

The multi pool case – QoS

- If we do not put any QoS things happen in best effort to all the operations happening to the filesystem (workload, move between pools, restripes, …)
- In our case we want to prioritize the move away from RainbowDash 900. But we also want to give some margin to the workload. The answer, as usual, it depends
- The move between AppleJack VI and FlutterShy 5K is a batch job that is admin triggered
- It is a very good idea to set QoS in this case as it was on the migration case
- For this example I am going to give migration away from RainbowDash 900 more IOPS than migration from/to the other pools.
- As my most of my users are localized in similar timezones I a going to trigger the AppleJack VI and FlutterShy 5K migrations off peak
- QoS for AppleJack VI and FlutterShy 5K pools would have different values during off peaks

The multi pool case – QoS

```
[root@canterlot ~]# mmchqos EQUESTRIA --enable pool=system,maintenance=3000IOPS,other=unlimited
pool=RainbowDash,maintenance=2000IOPS,other=unlimited pool=AppleJack maintenance=1000IOPS,other=unlimited
pool=*,maintenance=500IOPS,other=unlimited
Adjusted QOS Class specification:
pool=RainbowDash,other=inf,maintenance/all_local=2000Iops:pool=system,other=inf,maintenance/all_local=3000Iops
:pool=*,other=inf,maintenance/all_local=500Iops
QOS configuration has been installed and broadcast to all nodes.

[root@canterlot ~]# mmlsqos EQUESTRIA
QOS config::             enabled --
pool=RainbowDash,other=inf,maintenance/all_local=1000Iops:pool=system,other=inf,maintenance/all_local=3000Iops
:pool=*,other=inf,maintenance/all_local=500Iops
QOS values::
pool=system,other=inf,maintenance/all_local=3000Iops:pool=AppleJack,other=inf,maintenance/all_local=500Iops:po
ol=FlutterShy,other=inf,maintenance/all_local=500Iops:pool=RainbowDash,other=inf,maintenance/all_local=1000Iop
s
QOS status::             throttling active, monitoring active
```

The qualifier /all_local after maintenance indicates that the maintenance IOPS are applied to all the files systems owned by the cluster. This value is the default for the maintenance class.

The multi pool case – QoS

- QoS fine-grained with one pool can also separate per pool metrics
- That includes nodes, block, sectors, R/W, …

The multi pool case – Detail, details

- In Spectrum Scale you can currently use up to 16MB blocksize in a filesystem
- However some storages do not take this very well. Im Linux you need to look into /sys/block/DEVICE/queue in particular max_sectors_kb and max_hw_sectors_kb
- max_sectors_kb configured maximum IO size (Linux default 512 KiB)
- max_hw_sectors_kb supported maximum IO size (HW driver specific)
- For most of the cases you want to go **at least** the filesystem blocksize with the max_sectors_kb. You can give a try to align it with max_hw_sectors_kb
- For some drivers/devices optimal_io_size has a non zero value use that or multiples of that only

The multi pool case – Detail, details

- If your blocksize > max_sectors_kb more IO operations are needed to read/write a block
- If your blocksize > max_hw_sectors_kb a get ready for a bumpy ride with some storages
  - With AppleJack IV when using metadata was painful: hanging, errors on dmesg, unusable
  - With FlutterShy 5K nothing happened during my tests (8MB / 16MB for data and metadata)
- Complete Fair Scheduler (CFQ) is the default IO scheduler in Linux, give a try deadline and noop. Or even better ask the storage manufacturer
- CFQ allows grouping IO requests before sending it to the device (quantum), price is latency
- nr_requests default is 128, give a try to 256, queue_depth marks the pace

The multi pool case – Detail, details

- /sys/block/DEVICE/device/queue_depth Linux default is 16 … danger area. Our documentation states that for SATA/SAS drives use 32. Check with the storage vendor for centralized storage.
- My very personal rule of thumb to start with is to match it with the data disks of a RAID5/6 on FlutterShy (8 in this case). For AppleJack and RainbowDash whatever the manufaturer recommends
- Caching (SBC) [ca] mode page:

```
  WCE        0  [cha: n, def:  0, sav:  0]  Write cache enable
  RCD        0  [cha: n, def:  0, sav:  0]  Read cache disable
```

Overall take aways

- Understand the underlaying storage (no news here, right?)
- Reallife™ small writes are really bad for RAID5/RAID6 performance
    - Some storages try to group writes, impact on latency
    - An entry write pool might easy up that burden for that particular workload
- Ask the manufacturer, check the drives maximum transfer IO
- Linux IO schedulers matter, try to have fun
- Linux device queue matters, more does not mean faster, try to have fun
- Dream scenario of only one workload? Tune for it.
- Limited workload types? Divide and conquer with filesystems an option
- Try QoS even if only for information gathering, adapt QoS

Questions?

Thanks a lot for being here, I hope that you enjoyed it

References

- LINUX IO performance tuning for IBM System Storage, V1.4
- Operating system configuration and tuning
- mmlsqos command