



IBM Spectrum Scale

**IBMs Elastic Storage Server
in
Synch and Share environments**



- ESS in a sync and share solution stack
- Spectrum Scale and ESS
- Performance & Integration (outlook)



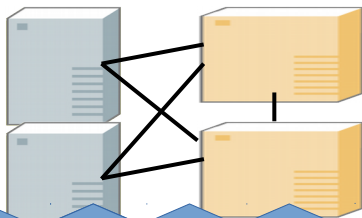
HPC data



streaming data



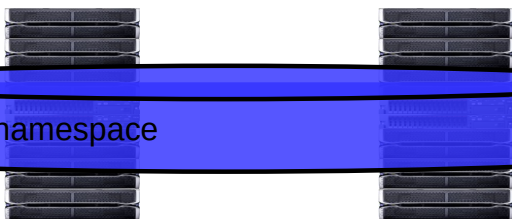
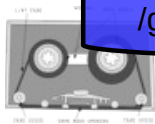
access network



I N T E G R A T I O N

10 / 40 / 100 Gbit IB / RDMA

/global/big/namespace



Clients, desktops ... etc..

- scales massive parallel
- 1000's of end points

application layer/ middle ware

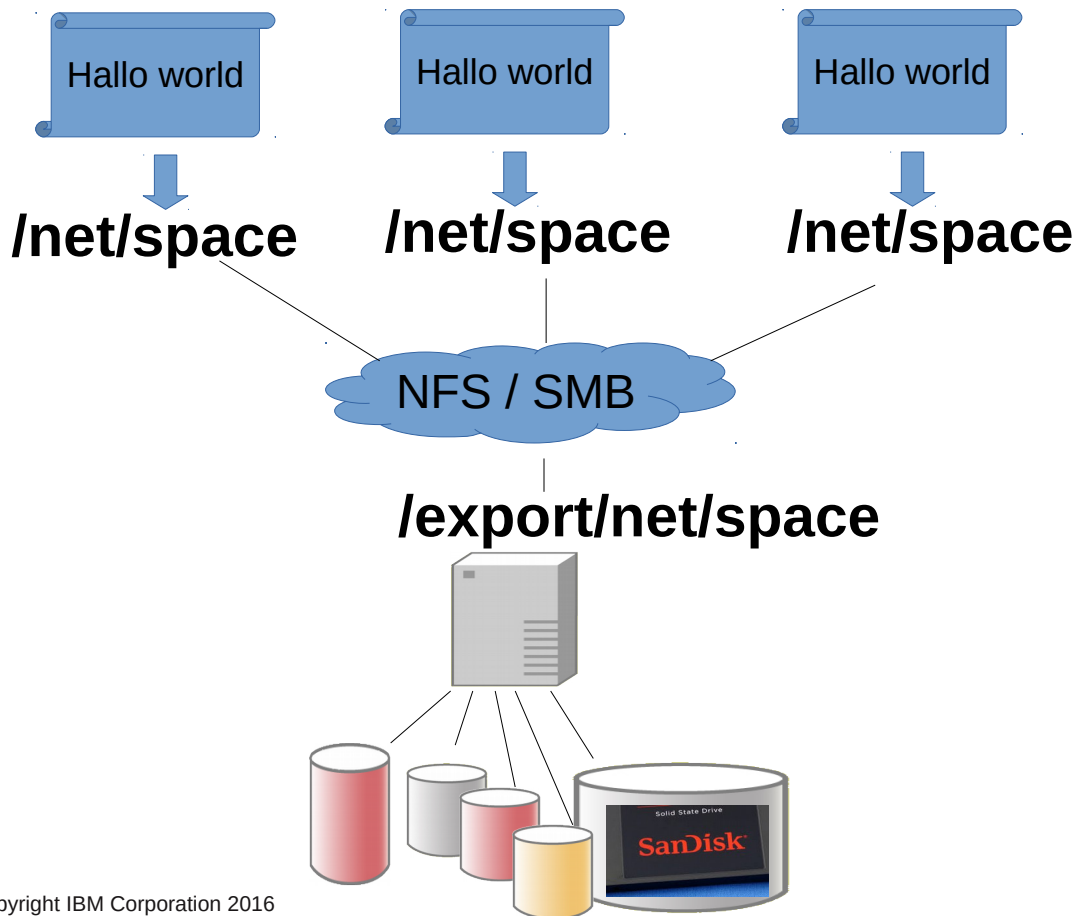
- data bases
- web front ends

Back end / storage

- parallel vs. shared file system
- underlying infrastructure
- scale-able, robustness & redundancy
- performance

- ESS in a sync and share solution stack
- Spectrum Scale and ESS
- Performance & Integration (outlook)

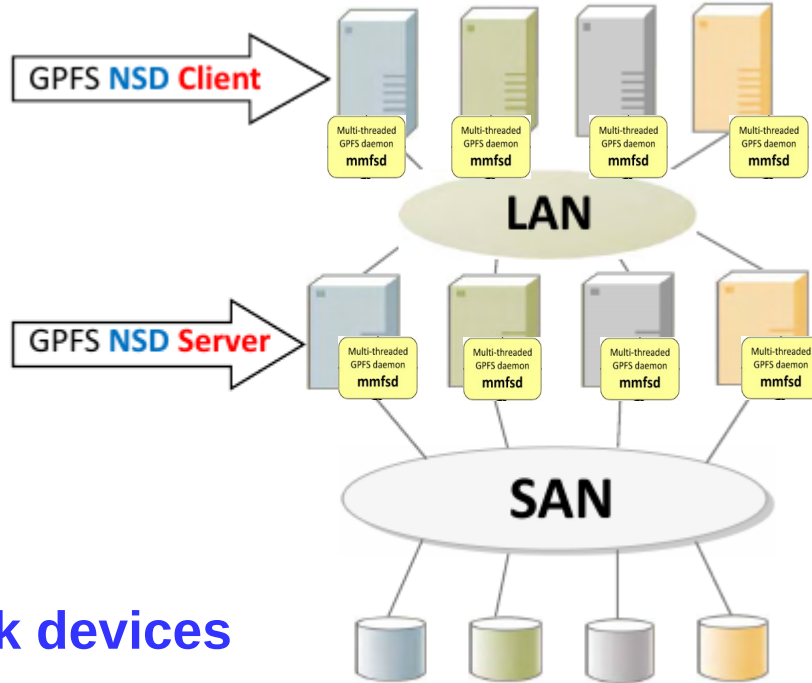
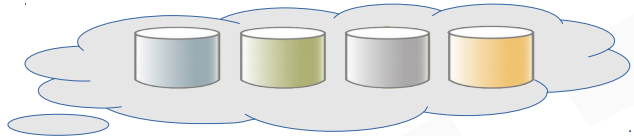




Limits:

- protocol stack
- capacity
- availability
- bandwidth
- latency

key feature of Spectrum Scale:
Network Shared Disk

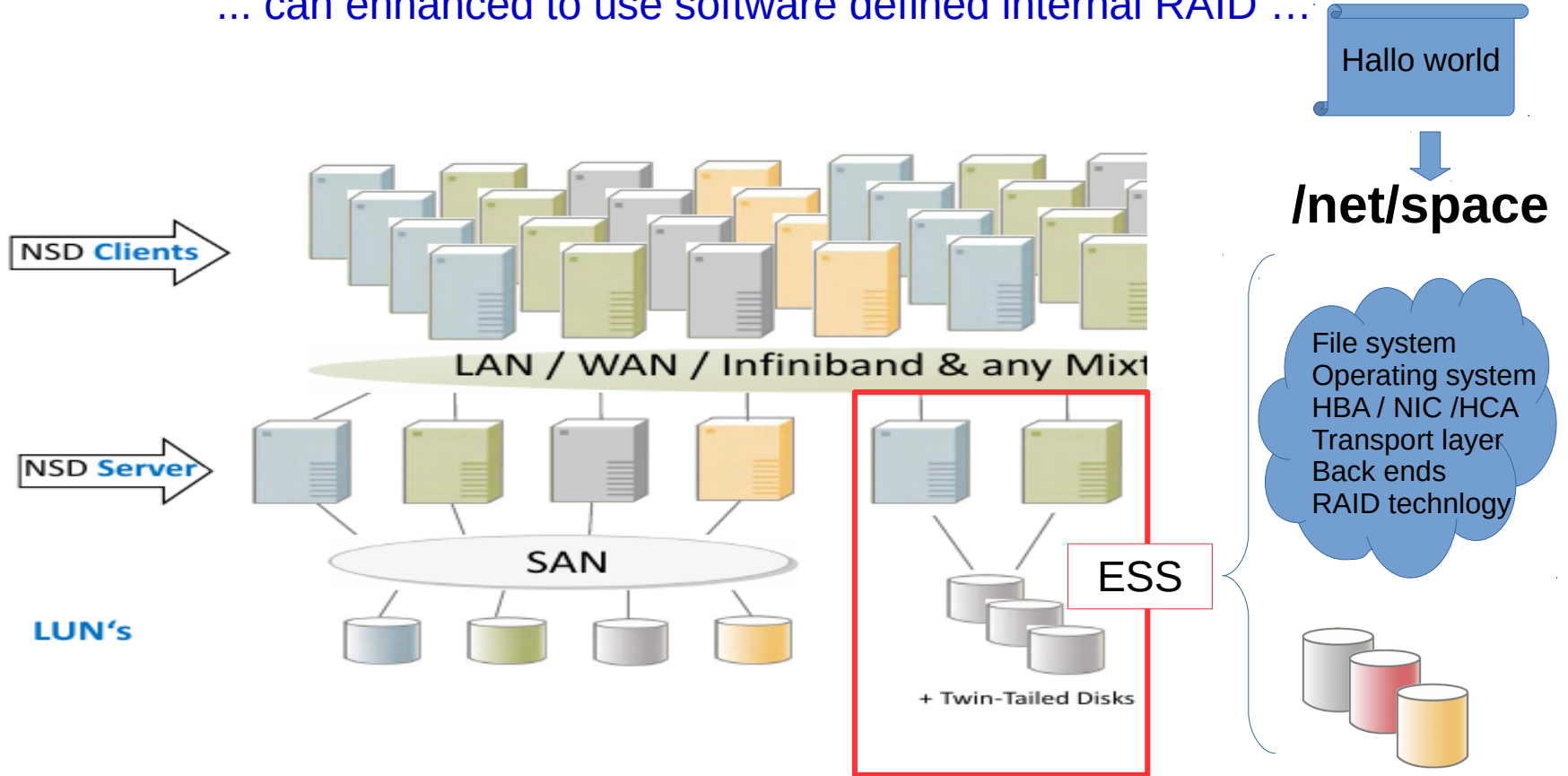


SpectrumScale / GPFS

- code available for
 - AIX, Linux x86_64, Linux ppc64
 - windows, zLinux
- POSIX compliant
- various additional features
- disks are shared for parallel access
- has its own high efficient protocol
 RPCs for “disk access”

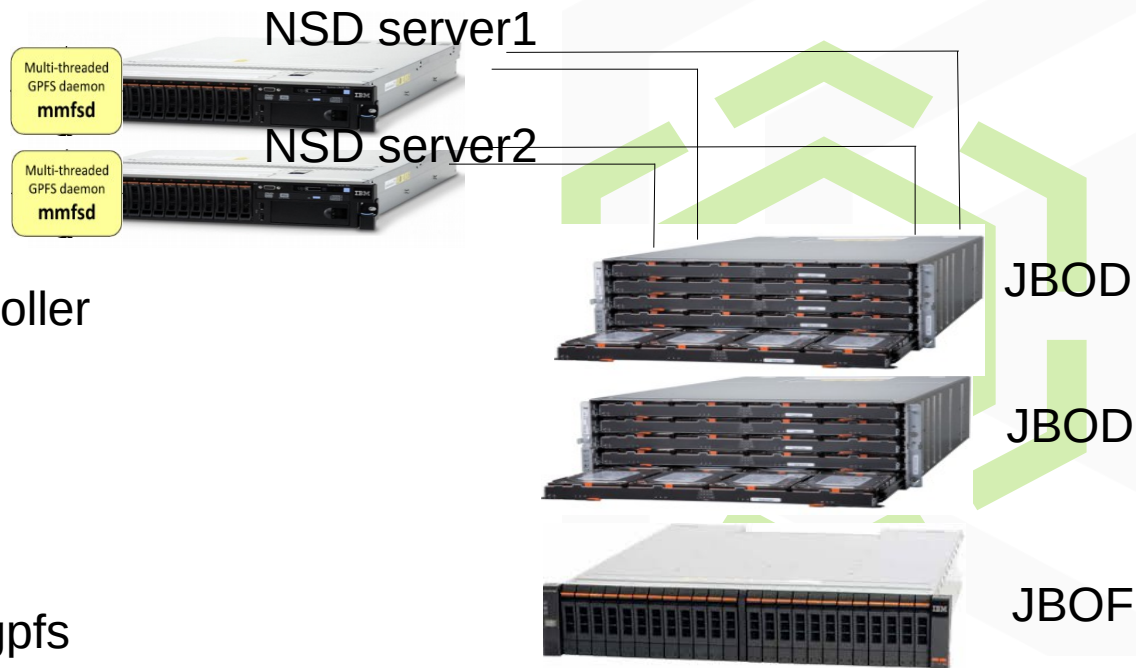
block devices

... can enhanced to use software defined internal RAID ...

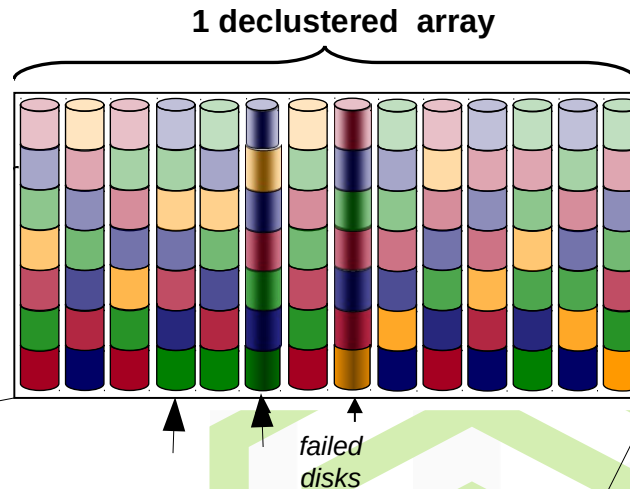


What is an ESS ?

- 2 nodes, connected to the same set of disks
- SAS connectivity
- JBODs , JBOFs
- no classical storage controller
- spectrum Scale software (gdfs.gnr.xxx.rpm)
- provides NSDs ready to use storage for gdfs



- no long running degraded RAID5
- rebuild process runs in background
“with low noise”
- critical data rebuilt less than 5 minutes



8+3p erasure coding within a 58 DDM and 3 faults:

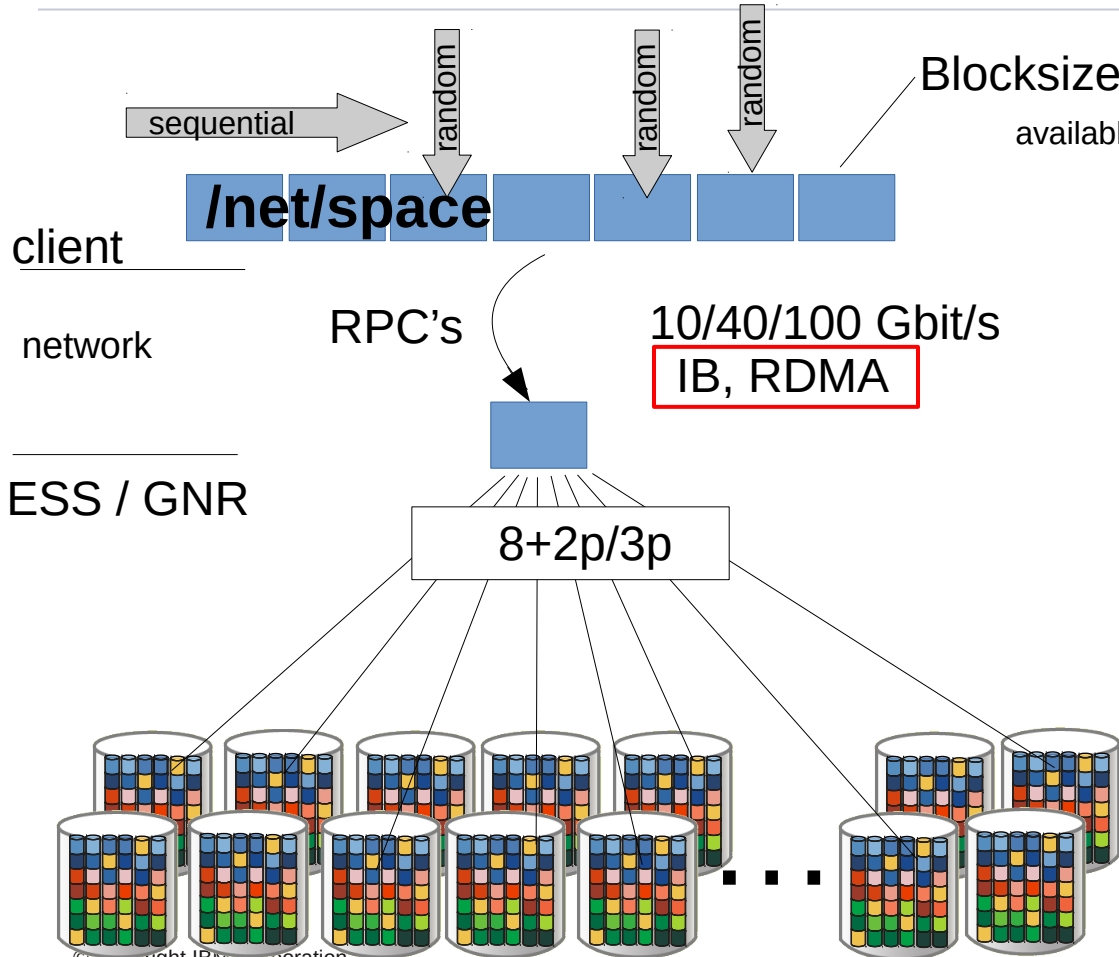
- Less than 1 % of all stripes affected by concurrent failure of 3 disks **critical**
- 3.3 % of all stripes affected by concurrent failure of 2 disks **not critical / background**
- 19 % of all stripes affected by concurrent failure of 1 disk **not critical / background**

* Example assumes

Fraction of stripes with 1 failure = $M/N = 11/58 = 19\%$

Fraction of stripes with 2 failure = $M/N * (M-1)/(N-1) = 11/58 * 10/57 = 3.33\%$

Fraction of stripes with 3 failures = $M/N * (M-1)/(N-1) * (M-2)/(N-2) = 11/58 * 10/57 * 9/56 = 0.53\%$



perfect hand over of workload

- workload prediction is done at the file system
- access is completely random (from the DDM's point of view)
- small fractions of a block can be also access directly
- end 2 end chksum protected



"Twin Tailed" JBOD
Disk Enclosure
X TB Drives

Capacity


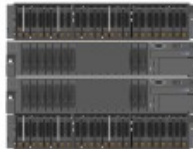


←

| | | |
|--|--|---|
| <p>Model GL2: 2 Enclosures, 12U 116 NL-SAS, 2 SSD</p>  | <p>Model GL4: 4 Enclosures, 20U 232 NL-SAS, 2 SSD</p>  | <p>Model GL6: 6 Enclosures, 28U 348 NL-SAS, 2 SSD</p>  |
|--|--|---|

→

Speed

←

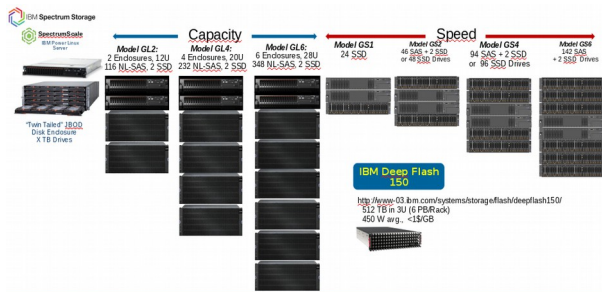
| | | | |
|---|---|---|--|
| <p>Model GS1 24 SSD</p>  | <p>Model GS2 46 SAS + 2 SSD or 48 SSD Drives</p>  | <p>Model GS4 94 SAS + 2 SSD or 96 SSD Drives</p>  | <p>Model GS6 142 SAS + 2 SSD Drives</p>  |
|---|---|---|--|

→

IBM Deep Flash 150

<http://www-03.ibm.com/systems/storage/flash/deepflash150/>
512 TB in 3U (6 PB/Rack)
450 W avg., <1\$/GB





- **Uses standard, inexpensive, JBOD disk drives**

- Erasure Code in software
- Disk Hospital for sick, slow, and failing disks

- **Sustained Data Availability = Data Durability**

- Critical Rebuilds = Days Minutes
- 8+2 or 8+3 Erasure Coding, mix/match with 3-way & 4-way mirrors

- **Minimal performance impact for rebuilds**

- Very few rebuilds require “critical rebuild”- finishes in minutes!

- **Data reliability and fault tolerance**

- End to end checksum
- No single point of failure,

- **Building block approach**

- Scales endless capacity,IOPS,bandwidth

NSD server1

Multi-threaded
GPFS daemon
mmfsd



NSD server2

Multi-threaded
GPFS daemon
mmfsd



JBOD



JBOD

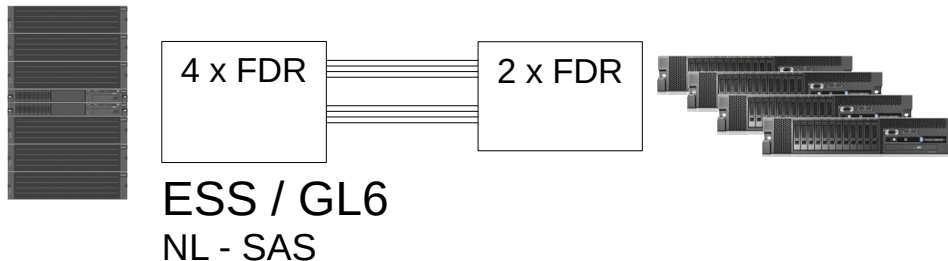


JBOF



- ESS in a sync and share solution stack
- Spectrum Scale and ESS
- **Performance & Integration (outlook)**





```
[root@rb3i0001 INFINIBAND]# mmdsh -N rb3h0200g,rb3h0201g,rb3h0202g,rb3h0203g,rb3h0205g,rb3h0206g,rb3h0207g,rb3h0204g,rb3h0208g "gpfsperf
create seq /gpfs/test/data/$(hostname)/100Gfile -n 100g -r 16m -th 12 -fsync" | grep "Data rate"
rb3h0206g: Data rate was 2925860.09 Kbytes/sec, thread utilization 0.771, bytesTransferred 107374182400
rb3h0201g: Data rate was 2889809.46 Kbytes/sec, thread utilization 0.749, bytesTransferred 107374182400
rb3h0202g: Data rate was 2888886.65 Kbytes/sec, thread utilization 0.770, bytesTransferred 107374182400
rb3h0203g: Data rate was 2863675.27 Kbytes/sec, thread utilization 0.766, bytesTransferred 107374182400
rb3h0205g: Data rate was 2859437.49 Kbytes/sec, thread utilization 0.771, bytesTransferred 107374182400
rb3h0200g: Data rate was 2767664.24 Kbytes/sec, thread utilization 0.835, bytesTransferred 107374182400
rb3h0207g: Data rate was 2738951.66 Kbytes/sec, thread utilization 0.867, bytesTransferred 107374182400
rb3h0204g: Data rate was 2340173.58 Kbytes/sec, thread utilization 0.917, bytesTransferred 107374182400
rb3h0208g: Data rate was 1150506.74 Kbytes/sec, thread utilization 0.749, bytesTransferred 107374182400
```

23 GB/s

```
[root@rb3i0001 hwcct]# mmdsh -N rb3h0200g,rb3h0201g,rb3h0202g,rb3h0203g,rb3h0205g,rb3h0206g,rb3h0207g "gpfsperf read seq /gpfs/test/data/$(
hostname)/100Gfile -n 100g -r 16m -th 12 -fsync" | grep "Data rate"
rb3h0200g: Data rate was 4779483.20 Kbytes/sec, thread utilization 0.968, bytesTransferred 107374182400
rb3h0203g: Data rate was 4428156.11 Kbytes/sec, thread utilization 0.973, bytesTransferred 107374182400
rb3h0206g: Data rate was 4419566.91 Kbytes/sec, thread utilization 0.980, bytesTransferred 107374182400
rb3h0205g: Data rate was 4413607.93 Kbytes/sec, thread utilization 0.972, bytesTransferred 107374182400
rb3h0202g: Data rate was 4409906.75 Kbytes/sec, thread utilization 0.985, bytesTransferred 107374182400
rb3h0201g: Data rate was 4408141.93 Kbytes/sec, thread utilization 0.982, bytesTransferred 107374182400
rb3h0207g: Data rate was 4408088.04 Kbytes/sec, thread utilization 0.984, bytesTransferred 107374182400
```

31,3 GB/s



ESS / GL6
NL - SAS

```
rb3h0201 [data] #
gpfsperf create seq /gpfs/test/data/tmp1/file100g -n 100g -r 16m -th 8 -fsync

gpfsperf create seq /gpfs/test/data/tmp1/file100g
  recSize 16M nBytes 100G fileSize 100G
  nProcesses 1 nThreadsPerProcess 8
  file cache flushed before test
  not using direct I/O
  offsets accessed will cycle through the same file segment
  not using shared memory buffer
  not releasing byte-range token after open
  fsync at end of test
  Data rate was 10439747.70 Kbytes/sec, thread utilization 0.784,
bytesTransferred 107374182400
rb3h0201 [data] #
```

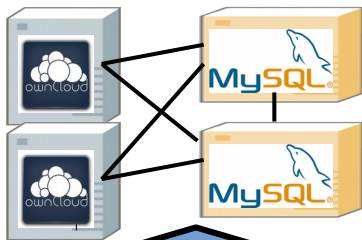
10 GB/s

HPC data



access network

streaming data



40 / 100 Gbit RDMA

/global/big/namespace

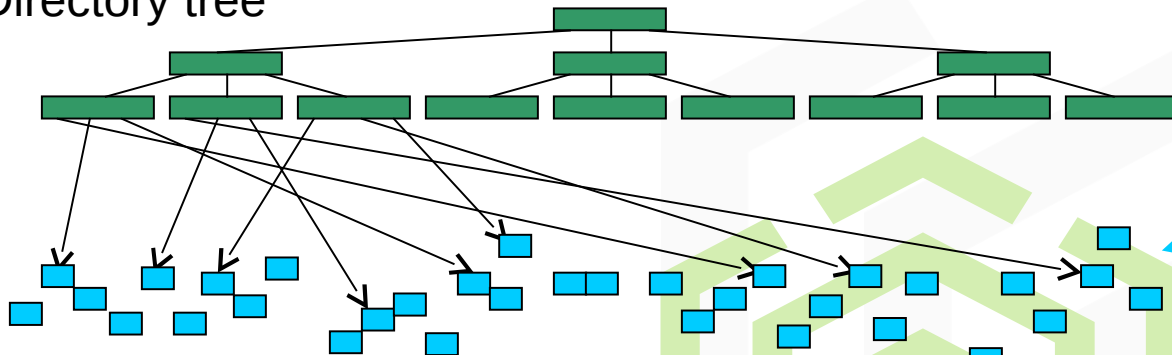
For a better integration:

- Spectrum Scale supports fast EA
- SpectrumScale has a default inode size of 4k
- massive MetaData performance by spectrum scale's internal metadata placement

poor meta data performance on file access

Directory
= Small random
reads

Directory tree



Inode's distributed...

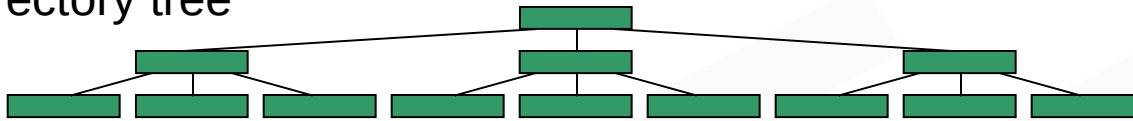
- › Random reads in Inode file to "stat" files.
- › One small block read per file.
- › Hard to parallelize.

Have you every tried a *find* / in a file system with millions of dirs/files ?



Consider Spectrum Scale is a parallel file system

Directory tree



ILM
= Large sequential
reads

Inode file

Provide direct / sequential access to file metadata!



- One large block read for n files.
- Read in parallel from all nodes.

- all meta data is organized in a hidden file
- MD can be stored separately
- can be accessed sequentially and in parallel
- mmfsd *underneath* distributes workload among all (“specified”) nodes
- high efficient policy engine as part of mmfsd

```
[root@lsdma-lab02 ~]# mmapplypolicy cache1 -P find_all-policy
```

```
[I] GPFS Current Data
```

| Pool_Name | KB_Occupied | KB_Total | Percent_Occupied |
|-----------|-------------|--------------|-----------------------------|
| beamline | 378880 | 3362684928 | 0.011267187% |
| data | 11885339648 | 103217584128 | 11.514839984% |
| system | 0 | 0 | 0.000000000% (no user data) |

```
[I] 5720949 of 69023488 inodes used: 8.288409%.
```

```
[I] Loaded policy rules from
```

```
Evaluating policy rules with
```

```
CURRENT_TIMESTAMP = 2017-01-24@15:55:50 UTC
```

```
Parsed 0 policy rules.
```

```
[W] Attention: It seems there are no effective nor useful rules. You may want to terminate this command!
```

```
/* null policy */
```

```
[I] 2017-01-24@15:56:32.787 Directory entries scanned: 2723526.
```

```
[I] Directories scan: 2519540 files, 202288 directories, 1698 d
```

```
2519540 files, 202288 directories
```

```
[I] 2017-01-24@15:56:37.618 Sorting 2723526 file list records.
```

```
[I] Inodes scan: 2519539 files, 202288 directories, 1699 other
```

```
[I] 2017-01-24@15:57:03.617 Policy evaluation. 2723526 files scanned.
```

```
[I] 2017-01-24@15:57:03.624 Sorting 0 candidate file list records.
```

```
[I] 2017-01-24@15:57:04.164 Choosing candidate files. 0 records scanned.
```

```
[I] Summary of Rule Applicability and File Choices:
```

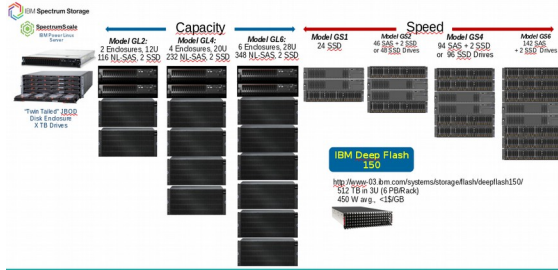
| Rule# | Hit_Cnt | KB_Hit | Chosen | Time | Rule |
|-------|---------|--------|--------|-----------|------|
| | | | | 01:14 min | |

```
[I] Filesystem objects with no applicable rules: 2723525.
```

```
Predicted Data Pool Utilization in KB and %:
```

| Pool_Name | KB_Occupied | KB_Total | Percent_Occupied |
|-----------|-------------|----------|------------------|
|-----------|-------------|----------|------------------|

Latest Tests in 2015 show 1 Billion File Scan on a single NL-SAS drive based ESS GL6 model takes under 2 minutes with just 8 nodes !



IBMs Elastic Storage Server in Synch and Share environments

ESS & Spectrum Scale:

- a real parallel file system,
- supports TCP, Infiniband and RDMA
- designed for low latency and high bandwidth
- comes with its own RAID as a real software defined storage solution
- ton's of additional functions

