

Spectrum Scale Expert Talks

Episode 5:

Update on functional Enhancements in Spectrum Scale



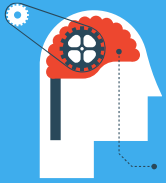
IBM
**Spectrum
Scale**

Show notes:

www.spectrumscaleug.org/experttalks

Join our conversation:

www.spectrumscaleug.org/join



SSUG::Digital

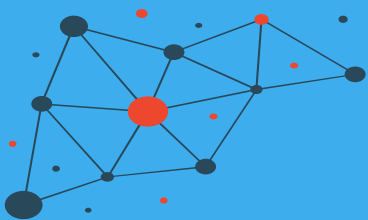
Welcome to digital events!



IBM
**Spectrum
Scale**

Show notes:
www.spectrumscaleug.org/experttalks

Join our conversation:
www.spectrumscaleug.org/join

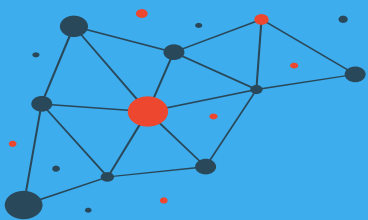


About the user group

- Independent, work with IBM to develop events
- Not a replacement for PMR!
- Email and Slack community
- <https://www.spectrumscaleug.org/join>

#SSUG





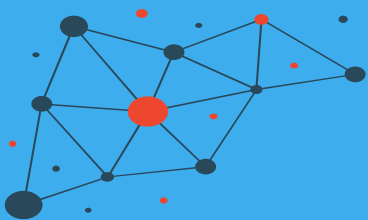
We are ...

- Simon Thompson (UK)
- Kristy Kallback-Rose (USA)
- Bob Oesterlin (USA)
- Bill Anderson (USA)
- Chris Schipalius (Australia)



#SSUG

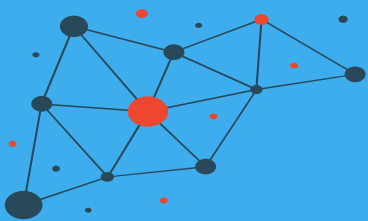




Check <https://www.spectrumscaleug.org/experttalks>
for charts, show notes and upcoming talks

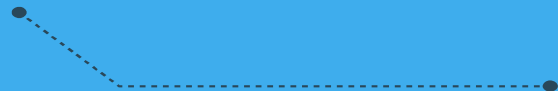
- Past talks:
 - 001: What is new in Spectrum Scale 5.0.5
 - 002: Best practices for building a stretched cluster
 - 003: Strategy update
 - 004: Update on performance enhancements in Spectrum Scale (file create, MMAP, direct IO, ESS 5000)
- Today:
 - 005: Update on functional enhancements in Spectrum Scale (inode management, vCPU scaling, NUMA considerations)
- Next:
 - Oct 6: Update on Spectrum Scale Container Storage Interface (CSI)
 - Oct 21: Best practices for information lifecycle management (ILM)
 - Nov: SSUG @ SC20





Speakers

- Karthik Iyer (IBM)
 - Inode management
- Mike Harris (IBM)
 - vCPU scaling
 - NUMA considerations



Disclaimer



IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

IBM reserves the right to change product specifications and offerings at any time without notice. This publication could include technical inaccuracies or typographical errors. References herein to IBM products and services do not imply that IBM intends to make them available in all countries.

Spectrum Scale Inode Management & Enhancements

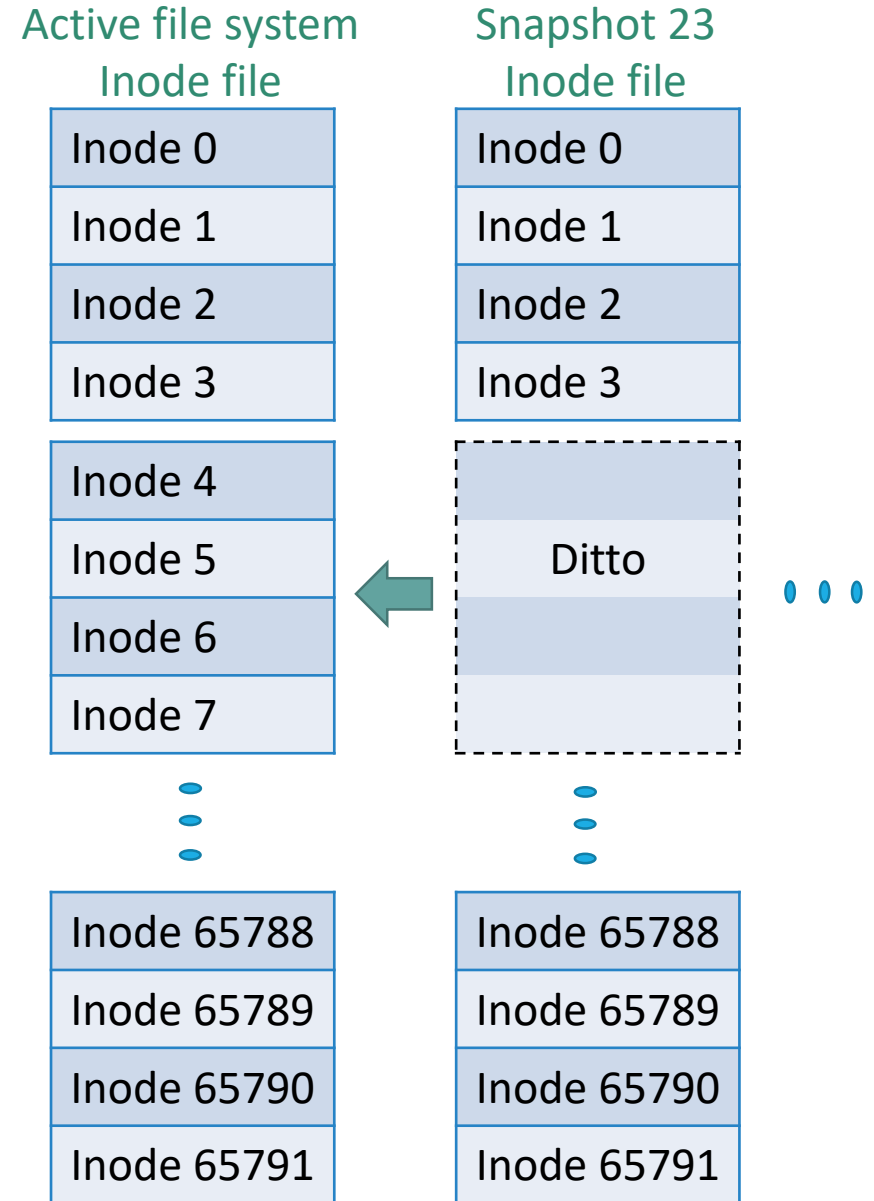
Karthik Iyer

Spectrum Scale Development



Inode File

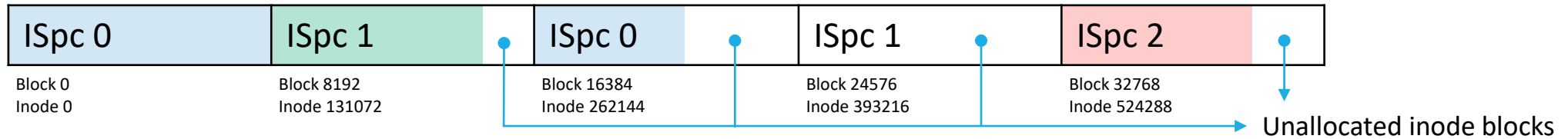
- System file with inode number 0
 - One inode file per active FS/Snapshot
- An **array** of inode records (default 4K size)
 - Indexed by inode number (Int64)
 - Mgmt. functions like restripe walk this array
 - Inode scan interface
 - `tsfindinode` maps inode numbers to file paths
- Inodes added dynamically up to max limit
- Max number of inodes limited by -
 - Number of small files that can possibly fit in the FS
 - User specified a lower max inodes limit
- Block level snapshot copy on write





Inode Spaces

- Supports **independent filesets**
- Dependent filesets partition file system namespace, but
 - Each inode block contains inodes from different filesets, so
 - Per-fileset inode scan is expensive
 - Fileset snapshot is expensive
- Partition inode range into set of disjoint inode ranges



- All inodes in an inode block belong to the same inode space
- Simple translation between inode number and inode space ID using inode masks

				Inode Space ID = 1				Internal Inode Number = 161034			
Inode Number	423178	0 0 0 0	0 1 1 0	0 1 1 1	0 1 0 1	0 0 0 0	1 0 1 0				
Inode Block Mask	0x5FFFF	0 0 0 0	0 1 0 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1				
Inode Space Mask	0xA0000	0 0 0 0	1 0 1 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0				



Inode Allocation Map

- System file with inode number 2 (only one per file system)
- Logically a large **bit vector** that tracks free/in-use status of all inodes
 - 2 bits per inode (free, in-use, being-created, to-be-deleted)
 - Efficient lookup of free inodes for file/directory/symlink creates
 - Tracks unlinked inodes for deferred deletion in case of node failure
- Physically divided into fixed sized lockable units called **segments**
 - Segments can be smaller than the inode allocation map file block size
 - The number of segments determine the node parallelism
 - Function of number of nodes (mmcrfs -n NumNodes) and metadata disks
 - A node can lock up to **maxActiveIallocSegs** segments per inode space
- Supports multiple inode spaces using **ialloc masks** (similar to inode masks)
 - Each segment belongs to a single inode space
 - Inode bits from a single inode space are not necessarily contiguous
 - Can have holes (sparse file)



Inode Allocation Map - Gotchas

- ‘mmcrfs –inode-limit’ only applies to the **root fileset**
 - Use mmcrfileset/mmchfileset to set inode limits for independent filesets
- Do not **pre-allocate** a large number of inodes during file system create
 - There is no option to de-allocate inodes to reclaim space
 - Can only de-allocate entire independent fileset (mmdelfileset)
 - Cannot deallocate root fileset
 - Allow inodes to expand on demand
- Do not use a large value of ‘mmcrfs –n **NumNodes**’
 - This causes too many ialloc segments to be created and not enough inode bits per segment
 - Small ialloc segments get used up fast and nodes have to search for free segments often
 - Nodes lock random segments to look for free inode and increase chances of lock collision
 - Increases size of inode allocation map
 - Caveat: Too small NumNodes can also lead to lock contention on the inode allocation map
- Possible to ‘re-layout’ inode allocation map after file system creation (5.0.4.0)
 - ‘tsdbfs <fs> patch imap test/format’ - use under IBM support guidance
 - Can test with different nNodes, nMetadataDisks and nInodes to find the optimal layout



Inode Expansion

- Adds blocks to inode file of active file system and inode allocation map file
- Expansion limited by max inodes or file system capacity
- Automatic expansion
 - Triggered when the number of free inodes in an inode space falls **below 25%**
 - Expands by $\min(25\%, 4M)$
 - **Now checked during free inode stats update instead of background sync (5.0.4.0)**
 - Even if 75% of segments are full, each node has at least one segment with free inodes
- Manual expansion
 - Forced by 'mmchfileset -inode-limit MaxNumInodes:**NumInodesToPreallocate**'
- Map format V1 -> V2
 - V1
 - Surplus bits in ialloc segment are marked used so inode allocators cannot use them
 - After expansion, some of the surplus bits are marked free
 - Requires locking all affected segments and invalidating segment caches
 - **V2 (4.1.1.0)**
 - **Surplus bits in ialloc segment are marked free**
 - **Inode allocators compute the allocated range every time and stay within the range**



Inode Manager

- Tracks the **number of free inodes** in each ialloc segment
 - In-memory accounting that is initialized during stripe group manager init/takeover
 - Reads all ialloc segments from disk
- Segment owner nodes periodically update the inode manager on free inodes count
 - Allows automatic inode expansion to determine if threshold has been reached
 - Shows free inode accounting in **mmdf**
- Problem: Inode expansion forces re-init of inode manager accounting
- Solution (5.0.3.0):
 - Avoided reading the ialloc segments from disk for re-init
 - Changed inode manager update RPC to reconcile stale and new free inode count
- Problem: Inode manager initialization slow for large inode allocation map
- Solution (5.0.4.0)
 - Lockless read of ialloc segments in units of full disk blocks
 - Stale free inodes accounting auto-corrected as part of async recovery
 - Operation is user interruptible



Inode Allocation

- To allocate a free inode
 - Picks a segment (among maxActiveIallocSegs) based on **parent directory inode number**
 - To ensure new inodes in same directory are allocated from the same segment & new inodes in different directories are allocated from different segments
 - Looks in recently deleted inode list (dealloc history) of this segment
 - Else looks for a free inode in this segment
 - Else searches for a free inode in other segments -
 - Uses ialloc segment hints (**ProbablyHasInodes**, HasInodes, ProbablyNoInodes, NoInodes)
 - Tries a **random segment** in order of (HasInodes, ProbablyHasInodes, ProbablyNoInodes)
 - **Co-ordinates thread searches to avoid redundant searches (5.0.1.0)**
 - **Ignores empty ialloc segments (5.0.4.0)**
 - Else tries to expand number of inodes and retries inode allocation
 - Else returns E_NOSPC error
 - Marks inode as **being-created** in the segment
 - Prefetches inode to replenish dealloc history on this segment if needed
- Initializes the inode on disk and adds directory entry for inode
- Marks inode bit as **in-use** in the inode allocation map



Inode updates and the MetaNode

- **Multi-node** writes to a single file can block on inode metadata update
- MetaNode **co-ordinates** inode metadata updates
 - Only metaNode reads and writes inode and indirect blocks
 - Collectes file size, mtime, and indirect block updates from other nodes
 - Merges inode updates by keeping largest file size and latest mtime
 - Other nodes only cache inode header and leaf indirect blocks
- Is a **dynamically elected** node role per file
 - Node that does first file access becomes metaNode
 - Acquires ww metaNode token; Other nodes acquire ro metaNode token
 - MetaNode token helps with inode destroyOnLastClose semantics
 - Can move due to operations like trunc, chmod, chown or frequency of metaNode requests
 - Can migrate to a remote node
- Also enables **fine grained directory locking**
 - Only metaNode reads and writes directory blocks
 - Other nodes obtain directory entries from the metaNode



Inode Deletion

- Unlink reduces inode link count; If link count becomes zero -
 - Marks inode as **to-be-deleted** in inode allocation map
 - If node fails after this point, async recovery will do deferred deletion of the inode
 - Deferred deletion now an interruptible multi-threaded lockless block scan operation (5.0.4.0)
 - 'mmfsadm dump deferreddeletions'
 - Now sends update request to token owner instead of revoking ialloc segment token (5.0.2.0)
 - Inode cannot be deleted as long as there are open instances or VFS references to the inode
 - Sets **destroyOnLastClose** flag on the open file object
- The node that closes the last open file instance to the inode -
 - Sends revoke request to all nodes that have a metaNode token
 - Nodes that don't have open instances relinquish their metaNode token
 - Others set destroyOnLastClose and take ownership of deleting the inode
 - If all nodes relinquished metaNode token, queues the inode for background deletion
 - Background file deletion is multi-threaded
 - Processed via dealloc function shipping
 - Once inode is deleted, marks inode as **free** in inode allocation map

IBM Spectrum Scale: vCPU Scaling and NUMA Considerations

Michael Harris



vCPU Scaling and NUMA Considerations

Spectrum Scale V5

The Spectrum Scale version 5 introduced pivotal enhancements to Spectrum Scale NUMA awareness and vCPU scaling.

- The FAQ stated 64 CPUs was the maximum single board SMP hardware supported.
- Spectrum Scale was not NUMA nor Cgroup / cpuset aware

In 5.0.x vCPU scaling work began and the NUMA subsystem was introduced within Spectrum Scale.

vCPU Scaling

Limits are based on virtual CPUs "vCPUs". Not on sockets nor physical cores.

A vCPU is

- A hardware slice or subdivision of a physical CPU; comprised of all or part of a physical CPU
- Based on configured hardware core threading levels – “HT” for x86_64 and “SMT” for Power
- Is the scheduled processor entity by LINUX / AIX

Tested and established Warning and Hard vCPU Limits

- Based on classic SMP and NUMA hardware configurations
 - Low NUMA Complexity; Low internode latency; High internode bandwidth
- Classic SMP machines generally qualify
- Large LPAR machines may (E980, MH9119), or may not (UV 2000 types)

vCPU Scaling

FAQ: What is the current maximum tested limit for SMP Scaling?

- The largest SMP scale tested to date is 192 cores
- The largest vCPU count tested to date is 1536 vCPUs
 - 192 Cores at SMT 8
- The largest NUMA Complexity metric tested to date is 3

Code: Startup time vCPU Limits are now checked

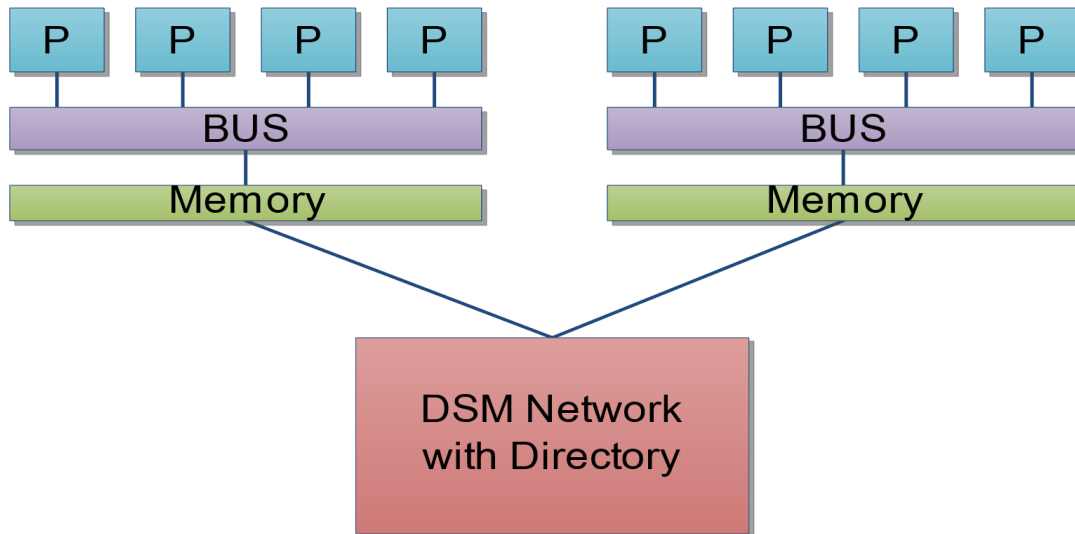
- Supported ≤ 256 vCPUs
- Warning Limit 256 - 1536 vCPUs; Please engage with Engineering for approval
- Hard Limit > 1536 vCPUs; Spectrum Scale will not complete initialization.
- vCPUs are as reported by the OS and Cgroups cannot be used to bypass limits.

NUMA Considerations

What is NUMA? NUMA refers to “Non Uniform Memory Access” between memory and processors

In the following simple illustration all processors can talk to all memory – but not uniformly

- The inter-processor and memory access is faster within their own group (eg Node) than to others
- This inter-node latency impacts multi threaded programs expecting symmetric multi-processing uniformity.



NUMA Considerations

How is NUMA characterized?

- The number of NUMA nodes (sometimes “domains”)
 - Spectrum Scale coined the term *NUMA Complexity* as a metric to characterize this value
- The latency between nodes.
 - *NUMA node distance* as reported by Linux *numactl --hardware*
- Values are as reported by, and dependent thereof on, firmware.
 - Essentially equivalent hardware may be reported differently by different vendors

NUMA Considerations

NUMA Complexity

- A Spectrum Scale coined metric to characterize (generally) the complexity of the NUMA environment
- Defined as the number of distinctly different NUMA node distances as reported by `numactl --hardware`
- The warning limit is 3 which corresponds to machines Spectrum Scale is known to work well with
- This is a characterization metric

The following system has a NUMA Complexity Metric of 3

- Can be misleading as nodes 252-255 are identifiable as graphics adapters not affecting Spectrum Scale

```
$ numactl --hardware
```

```
node distances:
```

```
node 0 8 252 253 254 255
```

```
0: 10 40 80 80 80 80
```

```
8: 40 10 80 80 80 80
```

```
252: 80 80 10 80 80 80
```


NUMA Considerations

NUMA Node Distance

- A platform metric to characterize the latency between any two given nodes
- Intended to be integral multiples of 10
- Intended to be accurate conveyance of inter-node latency
- As reported by platform firmware, *if reported by firmware*
- As often as not ad-hoc

Spectrum Scale reports values and views of Numa Complexity and Numa Distance(s)

- Does not make performance decisions based on this information.
- Advisory in logs and dumps when there are performance issues to be identified.
- Helps us understand machines we may not have seen before

NUMA Considerations

NUMA Awareness

- Awareness increasingly enhanced.
- Spectrum Scale can now be configured to startup with most Linux *numactl* options
- Verbs RDMA is NUMA aware but not yet pro-active
- Infiniband RDMA adapters may benefit from thread and memory locality
- We have done experiments to improve the NUMA efficiency of page pool allocations and are evaluating. We expect the benefits of any such improvements will depend on the hardware configurations being evaluated.
- What is possible depends on where hardware and memory reside.

Cgroup Awareness

A Cgroup is a grouping of processors and memory that a process or thread may be assigned to.

- Can be created and changed dynamically or pre-configured (such as for system components)
- Generally use to limit resources – and/or affine them as appropriate

Spectrum Scale has basic Cgroup awareness for Cgroups V1

- Largely informational logging and dumping of system vs actual run environment.
- Cgroups V2 is in RHEL 8; is not the RHEL default; is not qualified yet.

Not formally supported as

- Spectrum Scale isn't notified of Cgroup cpuset changes so can't log nor react to any changes.
- Spectrum Scale utilizes system limits for limits checking
 - Cgroups/cpusets are dynamic, so Spectrum Scale must be ready to handle the entire (maximum possible) system configuration (vCPUS, memory, hardware)
 - Therefore cannot be used to bypass vCPU and memory limits checking

Summary

Spectrum Scale 5.0 brought vCPU, NUMA, and Cgroup awareness and an enhancement roadmap.

- Classic SMP multi-socket motherboards with sane NUMA characteristics remain the gold standard for Spectrum Scale.
- A sane NUMA configuration has a Numa Complexity of 3 or less, with low latency and high bandwidth between NUMA nodes.
- Intelligent use of VMs/LPARs to high counts can work on sane NUMA architectures
- The amount of CPU and memory required for the reasons you use Spectrum Scale require sane NUMA.
- vCPU scaling, NUMA and Cgroup awareness and enhancement are ongoing
- Spectrum Scale is run in many different hardware configurations and we work closely with our customers to stay the best at what we do. We're glad to work on issues that might be seen in new cutting edge configurations.

Links

Wikipedia NUMA: https://en.wikipedia.org/wiki/Non-uniform_memory_access

Wikipedia Cgroups: <https://en.wikipedia.org/wiki/Cgroups>

Wikipedia Symmetric Multiprocessing: https://en.wikipedia.org/wiki/Symmetric_multiprocessing

Spectrum Scale 5.0.5 Limits FAQ: https://www.ibm.com/support/knowledgecenter/STXKQY/Spectrum_Scaleclustersfaq.html#smp


Thank you!



Please help us to improve Spectrum Scale with your feedback

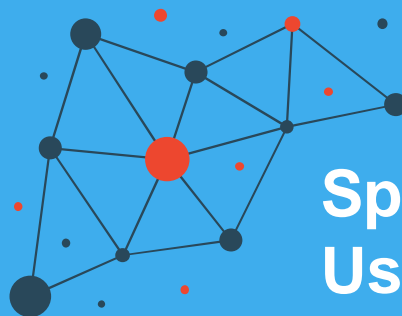
- If you get a survey in email or a popup from the GUI, please respond
- We read every single reply

Provide Feedback ×



Tell IBM What You Think

Let us know what you think about IBM Spectrum Scale. It takes only a couple of minutes for you to help us improve our service. [IBM Privacy Policy](#)



Spectrum Scale User Group

The Spectrum Scale (GPFS) User Group is free to join and open to all using, interested in using or integrating IBM Spectrum Scale.

The format of the group is as a web community with events held during the year, hosted by our members or by IBM.

See our web page for upcoming events and presentations of past events. Join our conversation via mail and Slack.

www.spectrumscaleug.org