TechU

IBM

Spectrum Scale Care and Performance

A109092

Jaqui Lynch
Flagship Solutions Group
jlynch@flagshipsg.net
http://www.circle4.com/ptechu/spectrumscale-oct0219.pdf

2019 IBM Systems Technical University
October 7-11, 2019 | Las Vegas, NV

1

Agenda

- Spectrum Scale Architecture
- Tuning and Monitoring
- Disaster Recovery

FLAGSHIP SOLUTIONS GROUP

2

Spectrum Scale Architecture

3

Terminology

- Manager Nodes
- NSDs and LUNs and Storage Pools
- Metadata and Data
- Blocksizes

4

Scale Manager Nodes

- Cluster Manager
  - One per cluster
  - Chosen via election among Quorum nodes
  - Monitors disk leases
  - Detects failures and manages recovery from node failure within the cluster.
  - Determines whether or not a quorum of nodes exists to allow the Scale daemon to start and for file system usage to continue.
  - Distributes certain configuration changes that must be known to nodes in remote clusters.
  - Selects the file system manager node.
  - Prevents multiple nodes from assuming the role of file system manager, thereby avoiding data corruption
  - Handles UID mapping requests from remote cluster nodes.
- To identify the cluster manager, issue the mmlsmgr -c command.
- To change the cluster manager, issue the mmchmgr -c command.

Scale Manager Nodes

- Filesystem Manager (FSM)
  - One per filesystem
  - Handles all of the nodes using the file system
  - Takes care of:
    - File system configuration
    - Adding disks
    - Changing disk availability
    - Repairing the file system
    - Management of disk space allocation
  - If quotas are enabled the FSM also manages quotas
  - Check where it is using mmlsmgr filesystemname
    - i.e. mmlsmgr gpfs1
  - Change using mmchmgr
  - Token management
    - The file system manager node may also perform the duties of the token manager server.

lsmgr output

file system manager node
---------- ----------
gpfs1 192.168.2.14 (jlaix14)
gpfs2 192.168.2.14 (jlaix14)
Cluster manager node: 192.168.2.14 (jlaix14)

Scale Manager Nodes

- Token Manager
  - Coordinates access to files on shared disks by granting tokens that convey the right to read or write the data or metadata of a file.
  - Ensures the consistency of the file system data and metadata when different nodes access the same file.
- Quorum Manager
  - Node quorum is the default quorum algorithm for Scale
  - With node quorum: Quorum is defined as one plus half of the *explicitly defined* quorum nodes in the Scale cluster
  - There are no default quorum nodes; you must specify which nodes have this role
  - There is also the option of Node Quorum with tiebreaker disks
  - Try to avoid selecting nodes that are likely to be rebooted
  - IBM suggests a maximum of 8 quorum nodes
  - Quorum nodes are defined in the node file that is used when the cluster is created
    - node23:quorum-manager
    - mmchnode can be used to change a node to nonquorum or quorum
- Protocol Manager

7

NSDs and LUNs and Storage Pools

- LUNs are assigned to the LPAR when disk is zoned and mapped and cfgmgr is run
- LUNs show up as hdisks
- Scale requires the LUNs to be formatted as NSDs before it can use them
- NSD Stanza is the list of relationships you want built between hdisks and NSDs
- The NSD stanza is input into mmcrnsd to build the relationship
- Then the NSD stanza is input into mmcrfs to create the filesystem
- NSDs can be placed into storage pools to separate them for different usage
- The default storage pool is System and all metadata must be in System

8

NSD Stanza and create NSDs

vi gpfs1-nsdstanza.txt
%nsd: nsd=nsdhdisk4 device=/dev/hdisk4 usage=dataOnly pool=dataPool
%nsd: nsd=nsdhdisk5 device=/dev/hdisk5 usage=dataOnly pool=dataPool
%nsd: nsd=nsdhdisk6 device=/dev/hdisk6 usage=dataOnly pool=dataPool
%nsd: nsd=nsdhdisk7 device=/dev/hdisk7 usage=dataOnly pool=dataPool
%nsd: nsd=nsdhdisk8 device=/dev/hdisk8 usage=dataOnly pool=dataPool
%nsd: nsd=nsdhdisk9 device=/dev/hdisk9 usage=dataOnly pool=dataPool
%nsd: nsd=nsdhdisk10 device=/dev/hdisk10 usage=dataOnly pool=dataPool
%nsd: nsd=nsdhdisk11 device=/dev/hdisk11 usage=dataOnly pool=dataPool
%nsd: nsd=nsdhdisk12 device=/dev/hdisk12 usage=metadataOnly pool=system
%nsd: nsd=nsdhdisk13 device=/dev/hdisk13 usage=metadataOnly pool=system
%nsd: nsd=nsdhdisk14 device=/dev/hdisk14 usage=metadataOnly pool=system
%nsd: nsd=nsdhdisk15 device=/dev/hdisk15 usage=metadataOnly pool=system

Now create NSD relationships – you can check using lspv and mmlsnsd
mmcrnsd -F /usr/local/etc/gpfs1-nsdstanza.txt

9

Create Filesystem

Create the filesystem
mmcrfs gpfs1 -F /usr/local/etc/gpfs1-nsdstanza.txt --metadata-block-size=256K -B 2M -m1 -M2 -r 1 -R 2 -T /fsgpfs1

-B is data blocksize

Table 1. Supported block sizes with subblock size

Supported block sizes with subblock size
64 KiB block with a 2 KiB subblock
128 KiB block with a 4 KiB subblock
256 KiB, 512 KiB, 1 MiB, 2 MiB, or 4 MiB block with 8 KiB subblock
8 MiB or 16 MiB block with a 16 KiB subblock

-m 1 –M2 default and maximum metadata replicas
-r 1 –R 2 default and maximum data replicas

-n ??? Default is 32
Number of nodes that will mount the file system in the local cluster and all remote clusters. Used to create maximum parallelism.

10

Types of MetaData

- Metadata is data about data
  - https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/General+Parallel+File+System+(GPFS)/page/Data+and+Metadata
  - http://files.gpfsug.org/presentations/2016/south-bank/D2_P2_A_spectrum_scale_metadata_dark_V2a.pdf
- Article from IBM Development on metadata
  - https://tinyurl.com/gpfsmetadata
- 3 Types
  - Descriptors
    - NSD, Files and Filesystem descriptors
  - System Metadata
    - Inodes
    - Inode allocation map
    - Block allocation map
    - Log files
    - Active control list files (ACLs)
    - Extended Attribute files
    - Quota files
    - Fileset metadata files
    - Policy files
    - Allocation summary files
  - User Metadata
    - files and directories.
    - Directories
    - Indirect blocks
    - Extended attribute overlow blocks

11

More on MetaData

- Metadata IO patterns are distinctly different from those of data
- Metadata performance is often the bottleneck
- Metadata IO Patterns
  - Directory traversal
  - Small file creation
  - Inode scan
  - Deleting large trees
- Typically 1-2% of filesystem space is needed for metadata. Can overprovision if not careful
- Random small read performance is very important, and so is random small write performance.
- Consider splitting metadata and data
  - Allows different blocksizes to address different I/O patterns

12

MetaData and Data Blocksizes

- Blocksize is the largest I/O that Scale can issue to a device
- Each data block is broken into 32 subblocks (or whatever was on –n at mmcrfs)
- Means 256KB block is 32 x 8KB blocks – smallest file will use 8KB
- Blocksize recommended depends on Raid setup (8+P is different to 4+P)

Data Blocksizes	
1-16MB	Large sequential I/O
512KB	Relational Database
256KB	Small Sequential I/O - default
256KB means 32 x 8KB subblocks	
Small data can occupy 8KB	
Large I/O operations can be 256KB	
Metadata blocksize defaults to blocksize above	
Metadata writes to large RAID stripes cause low performance	
Use 128KB or 256KB blocksize for metadata	
Use HAWC on metadata - helps write intensive workloads	
Metadata - raid 1 or 10 to avoid read-modify write penalty	

13

Blocksizes

- Using metadata blocks sizes larger than 1 MB and smaller than 256 KB is not recommended.
- To use a separate metadata blocksize to data, the metadata must be in the system pool and data must be in its own storage pool
- If both have the same blocksize they can coexist in the same pool
- Storage Pool Blocksizes
  - Data only Try 2MB
  - Metadata only Try 256KB
  - Combined Try 1MB
- Other Data options
  - 1-16MB Large sequential I/O
  - 512KB Databases such as Oracle or DB2
  - 256KB Small sequential I/O – file services, etc

Blocksize changes require recreation of filesystem so choose carefully

14

Tuning and Monitoring

15

Tunables

- AIX General
- Scale Specific
  - workerThreads
  - Pagepool
  - maxFilesToCache
  - maxMBPs
  - maxStatCache
  - Inodesize
  - Manager placement
  - Splitting data and metadata

16

AIX General

- CPU and Memory
- Sufficient disk LUNs
- Queue Depth on disks
- Tuning fibre adapters
- Network performance
- RAID settings on storage system

Disk LUNs

- Have more, smaller disks
  - Reduces queueing at the LUN level
  - Each LUN (NSD) has an OS queue for IO, more queues typically improves overall throughput and the workload can better utilize multiple IO paths
  - Lets Scale take advantage of its striping mechanisms
  - Ensure queue depth is set on hdisks
  - Check num_cmd_elems on fibre adapters
  - RAID levels

RAID Levels

Table 3-3 RAID levels comparison

RAID	Description	Application	Advantage	Disadvantage
0	Stripes data across multiple drives.	IOPS Mbps	Performance, due to parallel operation of the access.	No redundancy. If one drive fails, the data is lost.
1/10	The drive data is mirrored to another drive.	IOPS	Performance, as multiple requests can be fulfilled simultaneously.	Storage costs are doubled.
3	Drives operate independently with data blocks distributed among all drives. Parity is written to a dedicated drive.	Mbps	High performance for large, sequentially accessed files (image, video, and graphics).	Degraded performance with 8-9 I/O threads, random IOPS, and smaller, more numerous IOPS.
5	Drives operate independently with data and parity blocks distributed across all drives in the group.	IOPS Mbps	Good for reads, small IOPS, many concurrent IOPS, and random I/Os.	Writes are particularly demanding.

19

Scale Specific

- Memory
- Tunables
- Cluster Manager (pri and sec)
  - Have this on non busy LPAR
- Filesystem and Token Manager
  - Do not put on busy server or NSD server
- Quorum Managers
  - Recommended to include cluster managers as quorum nodes if non-CCR
- High write nodes should not be doing anything except data due to write workload
- Split metadata and data to separate NSDs
- Use Scatter for Block allocation
  - Recommended if more than 8 disks or more than 8 nodes

20

gpfsperf and mmpmon

- gpfsperf
  - Scale tool that allows you to test various I/O scenarios.
  - You can select random or sequential I/O along with filesizes and blocksizes to be tested.
  - Run multiple copies on multiple nodes at the same time
    - This tests concurrency within a node as well as concurrency between nodes.
- https://www-01.ibm.com/support/docview.wss?uid=isg15readmebbb63bf9mples_perf

- mmpmon
  - Used to collect I/O statistics for each mounted filesystem or for the whole node.
  - Run from one node using "nlist add" to add all the nodes you want to monitor at the same time.
  - The files produced are not intuitive, but there is information available in the references on how to run it and how to interpret the data as well as an awk script.
- https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.2/com.ibm.spectrum.scale.v5r02.doc/bl1adv_mpmover.htm
- https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.2/com.ibm.spectrum.scale.v5r02.doc/bl1adv_aganfior.htm

mmdiag

- "mmdiag –config"
  - Shows you the current configuration settings for the node.
  - Anything with an ! in front of it has been changed from the default.
  - mmlsconfig shows you the changed parameters but mmdiag shows all the parameters.
- "mmdiag –waiters"
  - Shows you any outstanding waits which is useful when trying to review buffer setups.
- "mmdiag –all"
  - Gets you everything.
- There are a number of other options to get an IO history or to review memory etc

- mmlsconfig
  - Displays the current configuration for the cluster

mmlsconfig

Configuration data for cluster CLGPFSDR.abc.local:
---
clusterId 17519807854055208838
autoload yes
dmapiFileHandleSize 32
ccrEnabled yes
cipherList AUTHONLY
prefetchPct 40
maxFilesToCache 20000
pagepool 64G
seqDiscardThreshhold 4G
maxMBpS 12800
workerThreads 128
maxblocksize 2048K
minReleaseLevel 5.0.3.0
adminMode central

File systems in cluster CLGPFSDR.abc.local:

/dev/GPFS1

mmdiag --config

```
===== dump config =====
Current time 2019-09-11_09:35:24-0500
  aioWorkerThreads 256
  aioWriteBufferCksum 0
! ccrEnabled 1
! cipherList AUTHONLY
! clusterId 17519807854055208838
  clusterManagerSelection PreferManager
! clusterName CLGPFS3.na.fti.local
. commandEmergencyThreads 4
. commandThreads 32
! dmapiFileHandleSize 32
. flushedDataTarget 32
. flushedInodeTarget 32
. logBufferCount 3
  logBufferSize 262144
  logWrapAmountPct 3
  logWrapBuffers -1
. logWrapThreads 16
  logWrapThreadsPerInvocation -1
  logWrapThresholdPct 17
. maxAllocRegionsPerNode 4
. maxBackgroundDeletionThreads 4

When I set workerThreads to 128
It changes the values in bold
```

```
! maxblocksize 2097152
. maxBufferCleaners 32
  maxBufferDescs -1
. maxFileCleaners 32
! maxFilesToCache 20000
. maxGeneralThreads 512
. maxInodeDeallocPrefetch 8
! maxMBpS 12800
# minReleaseLevel 2100
! myNodeConfigNumber 1
  nsdBufSpace (% of PagePool)  30
  nsdMaxWorkerThreads 512
  nsdMinWorkerThreads 16
! pagepool 68719476736
  pagepoolMaxPhysMemPct 75
  pagepoolPageSize 65536
. parallelWorkerThreads 21
! prefetchPct 40
. prefetchThreads 72
! seqDiscardThreshhold 4294967296
. sync1WorkerThreads 32
. sync2WorkerThreads 32
. syncBackgroundThreads 32
  worker1Threads 128
. worker3Threads 8
! workerThreads 128
```


Other Commands

- mmlsconfig
  - Displays the current configuration for the cluster
- mmlscluster
  - Shows the cluster architecture including nodes and their roles
- mmlsmgr
  - Identifies the filesystem manager for each filesystem
- mmlsfs all
  - Shows the settings for each filesystem
- mmlsnsd
  - Shows the NSDs and how they are allocated. You can run it with no flags or use -aL or -aM to get additional information on the NSDs.
- mmlsdisk
  - "mmlsdisk filesystemname" shows the information for the filesystem including sector size, whether the NSD is holding data or metadata and so on. You can add the -L or -M flags at the end to get additional data.
- mmcachectl
  - New command in Scale 5.0.2
  - Allows you to look at how the filesystems and pagepool are being used.
  - "mmcachectl show"
  - Can show overall use or by device, fileset, inode or filename.
  - https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.1/com.ibm.spectrum.scale.v5r01.doc/bl1adm_mmcachectl.htm

Scale and Memory

- Scale uses three areas of memory:
  - Memory allocated from the kernel heap
    - Control structures such as vnodes and related structures that establish the necessary relationship with the operating system.
  - Memory allocated within the daemon segment
    - file system manager functions
    - Will be larger on FSM since token states for the entire file system are initially stored there
    - File system manager functions requiring daemon memory include:
      - Structures that persist for the execution of a command
      - Structures that persist for I/O operations
      - States related to other nodes
  - Shared segments accessed from both the daemon and the kernel
    - Shared segments consist of both pinned and unpinned memory that is allocated at daemon startup

Scale and Memory

- Pinned Memory
  - This is called the Pagepool
  - Configured using the Pagepool parameter
  - Stored file data
  - Optimizes performance for various data access patterns
- Non-Pinned Memory
  - Information about open and recently opened files
    - Full inode cache
    - Stat cache

mmdiag - -memory

- Displays information about mmfsd memory usage
- There are several distinct memory regions that mmfsd allocates and uses

- Heap memory allocated by mmfsd
  - This area is managed by the OS and does not have a preset limit enforced by GPFS.

- Memory pools 1 and 2
  - Both of these refer to a single memory area, also known as the shared segment.
  - It is used to cache various kinds of internal GPFS metadata, as well as for many other internal uses.
  - This memory area is allocated using a special, platform-specific mechanism and is shared between user space and kernel code.
  - Shows preset limit on the maximum shared segment size, current usage, and some prior usage information

- Memory pool 3
  - Token manager pool.
  - Used to store the token state on token manager servers.
  - Shows preset limit on the maximum memory pool size, current usage, and some prior-usage information

mmdiag - -memory

mmfsd heap size: 131083488 bytes

Statistics for MemoryPool id 1 ("Shared Segment (EPHEMERAL)")
128 bytes in use
261133015385 hard limit on memory usage
262144 bytes committed to regions
1 number of regions
4 allocations
4 frees
0 allocation failures

Statistics for MemoryPool id 2 ("Shared Segment")
392491544 bytes in use
261133015385 hard limit on memory usage
589168640 bytes committed to regions
2192 number of regions
47254987 allocations
47126223 frees
0 allocation failures

Statistics for MemoryPool id 3 ("Token Manager")
12724104 bytes in use
261133015385 hard limit on memory usage
16778240 bytes committed to regions
1 number of regions
661 allocations
512 frees
0 allocation failures

Pagepool

- Pinned memory
- Pagepool defines the amount of physical memory that should be pinned by Scale at startup.
- Caches data and metadata objects (indirect blocks, directory blocks).
- Allows Scale to implement read as well as write requests asynchronously.
- Increasing the size of the pagepool attribute increases the amount of data or metadata that Scale can cache without requiring synchronous I/O.
- Pagepool also supplies memory for various types of buffers like prefetch and write-behind
- Can't exceed total memory * pagepoolMaxPhysMemPct which is 75%
  - So if 128GB memory then Pagepool cannot be greater than 96GB

maxFilesToCache

- Default 4000
- Limits the number of files that can be cached in the pagepool
- When you hit this number the read I/O rate remains at the same level even if you enlarge the pagepool
- Set fairly large to assist with local workload
- It can be set very large in small client clusters, but should remain small on clients in large clusters to avoid excessive memory use on the token servers.
- Valid values of maxFilesToCache range from 1 to 100,000,000
- Should be large enough to handle the number of concurrently open files plus allow caching of recently used files
- Memory required is about maxFilesToCache x 3KB bytes

maxMBPs

- maxMBpS affects the depth of prefetching for sequential file access.
- Should be at least as large as the maximum expected hardware bandwidth.
- If too low then it limits I/O bandwidth.
- Indicator of the maximum throughput in megabytes that can be submitted by Scale per second into or out of a single node.
- It is not a hard limit
- Scale uses it to calculate how many prefetch/writebehind threads should be scheduled (up to the prefetchThreads setting) for sequential file access.
- Good starting point is 2X the I/O throughput the node can support

WorkerThreads

- Set WorkerThreads
- Run mmdiag - -config
  - The above shows you what else WorkerThreads changed
- For IBM Spectrum Scale 4.2.0.3 or 4.2.1 or later, it is recommended that the following configuration parameters not be changed (setting workerThreads to 512, or (8*cores per node), will auto-tune these values):
  - parallelWorkerThreads
  - logWrapThreads, logBufferCount
  - maxBackgroundDeletionThreads
  - maxBufferCleaners, maxFileCleaners
  - syncBackgroundThreads
  - syncWorkerThreads, sync1WorkerThreads, sync2WorkerThreads
  - maxInodeDeallocPrefetch
  - flushedDataTarget, flushedInodeTarget
  - maxAllocRegionsPerNode
  - maxGeneralThreads
  - worker3Threads
  - prefetchThreads.

maxStatCache

- Sets aside pageable memory to cache additional file attributes
- Default is either 1000 or 4 x maxFilesToCache
- Memory used is maxStatCache x 400 bytes

Memory Used

- Total memory used for caching
  - The combined memory to hold inodes, control data structures and the stat cache is limited to 50%of real memory. Current required total can be calculated by adding:
    maxStatCache * 400 bytes
    maxFilesToCache x 3172 bytes
    Pagepool size
      Pagepool can be changed dynamically
      maxStatCache & maxFilesToCache require a shut down and restart of the Scale daemons.

logBufferCount

- Number of Scale log buffers.
- Default is 3
- **Auto tuned if WorkerThreads is set**
- Having lots of these allows the log to absorb bursts of log appends. For systems with large page pools (1 G or more),
- Log buffers are the size of the metadata block size
- There is a separate set of such buffers for each file system
- Increasing LogBufferCount cen help performance when you have a few file systems (1-4 for example).
- Log buffers are allocated per file system, so when you have a large number of file systems increasing this value typically does not improve performance.
- When you have a few file systems you can increase LogBufferSize
- mmchconfig logBufferCount=20
- Insufficient will show as IO waits - mmdiag - - waiters
- Other logging parms:
  - Scale log flush controls. When the log becomes logWrapThresholdPct, the log flush code is activated to flush dirty objects so the log records that describe their updates can be discarded. This percentage defaults to 50%, and although there is some code to allow changing it, modifying this value is not supported by mmchconfig. Log wrap will start logWrapThreads flush threads (default 8), which will flush enough dirty objects so the recovery start position can be moved forward by logWrapAmountPct percent (default 10%).

inode-limit

- -inode-limit
  - Should be increased from the default if you plan to support a large number of files in the filesystem.
  - Estimate a value for this using the following calculation:
    (metadata_disk_size * #metadatadisks) / (inodesize * defaultMetadataReplicas)
  - "df -i" or "mmdf filesystemname -F" shows how many inodes are free
    - Inode Information----------
    - Number of used inodes: 4038
    - Number of free inodes: 496186
    - Number of allocated inodes: 500224
    - Maximum number of inodes: 1228864
  - "mslsfs filesystemname –inode-limit" shows the current limit.
    - --inode-limit 1228864 Maximum number of inodes
- This value can be increased dynamically using mmchfs.

Busy System inode-limit

Inode Information

Number of used inodes: 33908050
Number of free inodes: 1776302
Number of allocated inodes: 35684352
Maximum number of inodes: 45745152

Others

- seqDiscardThreshold
  - Helpful to increase if multiple threads need to sequentially read the same file on a node
  - Beneficial with SAS workloads
    - Increase to value larger than largest file you want Scale to cache
- prefetchPCT
  - Default is 20%
  - Tells pagepool how much prefetching to do
  - For SAS I increase this to 40%
- Block Allocation
  - Two options – cluster and scatter
  - Cluster is default if fewer than 8 nodes or 8 NSDs in the cluster
  - Set at filesystem creation
  - Scatter is recommended
    - Especially for large configurations or random allocation patterns

Scale I/O Waits

- Primary I/O Waiters (log entries) when busy
  - Can get information using:
    - mmfsadm dump waiters
    - mmdiag --waiters
- Review tunables around Logs and Buffer cleaning
  - logBufferCount, logWrapAmountPCT and logwrapThreads
  - Possible Changes if workerThreads does not increase these:
    - Increase logBufferCount from 3 to 20
    - Increase log buffer size
- Run queue increases when load increases, blocked processes cause swapping and memory shortages
- Make sure there is plenty of physical memory
- Monitor for CPU/Memory spikes once changes have been made

mmdiag –waiters - Examples

=== mmdiag: waiters ===
0x7FF4C4010470 (81851) waiting 0.121286000 seconds, **CleanBufferThread**: for I/O completion
0x7FF4AC006FB0 (31467) waiting 0.050805000 seconds, **SyncHandlerThread**: for I/O completion
0x7FF4DC02E140 (183683) waiting 0.001407000 seconds, **AllocRecoveryWorkerThread**: for I/O completion
0x7F9350006DA0 (113166) waiting 0.126921262 seconds, **FileBlockRandomWriteFetchHandlerThread**: on ThCond 0x7F949C01A138 (0x7F949C01A138) (LogFileBufferDescriptorCondvar), reason '**force wait for buffer write to complete**'
0x7F92C80032D0 (128387) waiting 0.025805000 seconds, **SGExceptionLogAlmostFullThread**: for I/O completion
0x7F95401A31F0 (15034) waiting 0.022351000 seconds, **LogWrapHelperThread**: for I/O completion
0x7F92FC0008F0 (111289) waiting 0.020273000 seconds, **SharedHashTabFetchHandlerThread**: for I/O completion
0x7F953400CE20 (113196) waiting 0.015772000 seconds, **ExpandLastBlockHandlerThread**: for I/O completion
0x7F94A400F7B0 (113172) waiting 0.013171000 seconds, **FileBlockRandomWriteFetchHandlerThread**: for I/O completion

Above needed more log buffers, more workerThreads, etc

Linux example where disk is slow:
0x7FF074003530 waiting 25.103752000 seconds, WritebehindWorkerThread: for I/O completion on disk dm-14

https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/General+Parallel+File+System+(GPFS)/page/Interpreting+GPFS+Waiter+Information
https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/General%20Parallel%20File%20System%20(GPFS)/page/mmfsadm

Splitting Data and Metadata

- Split metadata and data to separate NSDs
  - Allows us to customize blocksizes
  - IO pattern for metadata is usually very different from data
  - Metadata is typically lots of small random reads and writes and it is much smaller
    - Great candidate for SSDs or flash
  - Need separate pools if different blocksizes
    - Reduces wasted space and MetaData capacity usage due to indirect blocksize
  - Allows use of Raid-10 for metadata (avoid write penalty), Raid-5 or 6 for data
  - Metadata must go in system pool

RAID 10 for MetaData

- Allows us to customize blocksizes
  - See next slide
- Reduces wasted space and MD capacity=usage due to indirect blocksize
- Use battery backed write cache for raid-10 if available
- Raid-10 for metadata, raid-5 or 6 for data
  - RAID 10 uses two write IOs to write a single block of data (one write IO to each drive in the mirrored pair). RAID 5 requires two read IOs (read original data and parity) and then two write IOs to write the same block of data. For this reason, random writes are significantly faster on RAID 10 compared to RAID 5.
  - RAID 10 rebuilds take less time than RAID 5 rebuilds. If one drive fails, RAID 10 rebuilds it by copying all the data on the mirrored drive to a replacement/hotspare drive. RAID 5 rebuilds a failed disk by merging the contents of the surviving disks in an array and writing the result to a spare.

Setting up disks for split data/metadata (AIX)

Note hdisk4-11 are 250GB luns for data
Assign disks to the LPAR and run cfgmgr to see them
hdisk 12-15 are 100GB luns for metadata

```
chdev -l hdisk4 -a pv=yes
chdev -l hdisk5 -a pv=yes
chdev -l hdisk6 -a pv=yes
chdev -l hdisk7 -a pv=yes
chdev -l hdisk8 -a pv=yes
chdev -l hdisk9 -a pv=yes
chdev -l hdisk10 -a pv=yes
chdev -l hdisk11 -a pv=yes
chdev -l hdisk12 -a pv=yes
chdev -l hdisk13 -a pv=yes
chdev -l hdisk14 -a pv=yes
chdev -l hdisk15 -a pv=yes
```

```
chdev -l hdisk4 -a queue_depth=64 -a reserve_policy=no_reserve
chdev -l hdisk5 -a queue_depth=64 -a reserve_policy=no_reserve
chdev -l hdisk6 -a queue_depth=64 -a reserve_policy=no_reserve
chdev -l hdisk7 -a queue_depth=64 -a reserve_policy=no_reserve
chdev -l hdisk8 -a queue_depth=64 -a reserve_policy=no_reserve
chdev -l hdisk9 -a queue_depth=64 -a reserve_policy=no_reserve
chdev -l hdisk10 -a queue_depth=64 -a reserve_policy=no_reserve
chdev -l hdisk11 -a queue_depth=64 -a reserve_policy=no_reserve
chdev -l hdisk12 -a queue_depth=64 -a reserve_policy=no_reserve
chdev -l hdisk13 -a queue_depth=64 -a reserve_policy=no_reserve
chdev -l hdisk14 -a queue_depth=64 -a reserve_policy=no_reserve
chdev -l hdisk15-a queue_depth=64 -a reserve_policy=no_reserve
```


Splitting Data and Metadata (Scale)

vi /usr/local/etc/gpfs1-nsdstanza.txt

```
%nsd:  nsd=nsdhdisk4 device=/dev/hdisk4 usage=dataOnly pool=dataPool
%nsd:  nsd=nsdhdisk5 device=/dev/hdisk5 usage=dataOnly pool=dataPool
%nsd:  nsd=nsdhdisk6 device=/dev/hdisk6 usage=dataOnly pool=dataPool
%nsd:  nsd=nsdhdisk7 device=/dev/hdisk7 usage=dataOnly pool=dataPool
%nsd:  nsd=nsdhdisk8 device=/dev/hdisk8 usage=dataOnly pool=dataPool
%nsd:  nsd=nsdhdisk9 device=/dev/hdisk9 usage=dataOnly pool=dataPool
%nsd:  nsd=nsdhdisk10 device=/dev/hdisk10 usage=dataOnly pool=dataPool
%nsd:  nsd=nsdhdisk11 device=/dev/hdisk11 usage=dataOnly pool=dataPool
%nsd:  nsd=nsdhdisk12 device=/dev/hdisk12 usage=metadataOnly pool=system
%nsd:  nsd=nsdhdisk13 device=/dev/hdisk13 usage=metadataOnly pool=system
%nsd:  nsd=nsdhdisk14 device=/dev/hdisk14 usage=metadataOnly pool=system
%nsd:  nsd=nsdhdisk15 device=/dev/hdisk15 usage=metadataOnly pool=system
```

mmcrnsd -F /usr/local/etc/gpfs1-nsdstanza.txt
To use a 2M blocksize for data have to shutdown GPFS as largest blocksize allowed is 1M then:
mmchconfig maxblocksize=2048K
mmstartup
mmcrfs gpfs1 -F /usr/local/etc/gpfs1-nsdstanza.txt --metadata-block-size=256K -B 2M -m1 -M2 -r 1 -R 2 –T /fsgpfs1
The above uses a 256K blocksize for metadata and 2M for data
Then use chmod and chown to set up permissions, mmmount gpfs1 and redo chmod and chown

45

mmcrfs

mmcrfs gpfs1 -F /usr/local/etc/gpfs1-nsdstanza.txt --metadata-block-size=256K -B 2M -m1 -M2 -r 1 -R 2 –T /fsgpfs1

https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.3/com.ibm.spectrum.scale.v5r03.doc/bl1adm_mmcrfs.htm

-F	Points to stanza file for NSDs for the filesystem
--metadata-block-size	Blocksize for metadata
-A	Default is yes, filesystem will automount when Scale starts
-B	Blocksize for data
-i	inodesize – can be 512, 1024 or 4096. 4096 is default
-j	Block allocation map – cluster or scatter
-L	Internal log file size . Specified in K or M. Usually allow to default
-m	Default metadata replicas (copies of inodes, directories and indirect blocks)
-M	Max metadata replicas
-n	Numnodes. Estimate of maximum nodes that will mount the filesystem. Used to determine initial size of some structures. Default is 32
-r	Default data replicas (copies of each data block)
-R	Max data replicas
-T	Filesystem mount point

46

Check status of new filesystem

mmdf gpfs1
mmlsdisk gpfs1
mmlsnsd
mmlscluster
mmlsconfig
mmgetstate -aLs
df -g /fsgpfs1
mmlsfs gpfs1

mmdf gpfs1

```
disk                disk size  failure holds    holds           free in KB          free in KB
name                    in KB    group metadata data        in full blocks        in fragments
--------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: system (Maximum disk size allowed is 1.02 TB)
nsdhdisk12          104857600       -1 yes      no        104018688 ( 99%)           472 ( 0%)
nsdhdisk13          104857600       -1 yes      no        104018688 ( 99%)           600 ( 0%)
nsdhdisk`4          104857600       -1 yes      no        104018432 ( 99%)           376 ( 0%)
nsdhdisk`5          104857600       -1 yes      no        104018176 ( 99%)           376 ( 0%)
             -------------                         -------------------- -------------------
(pool total)        419430400                           416073984 ( 99%)          1824 ( 0%)

Disks in storage pool: dataPool (Maximum disk size allowed is 1.16 TB)
nsdhdisk4           104857600       -1 no       yes       104787968 (100%)          3904 ( 0%)
nsdhdisk5           104857600       -1 no       yes       104787968 (100%)          3904 ( 0%)
nsdhdisk6           104857600       -1 no       yes       104787968 (100%)          3904 ( 0%)
nsdhdisk7           104857600       -1 no       yes       104787968 (100%)          3904 ( 0%)
nsdhdisk8           104857600       -1 no       yes       104787968 (100%)          3904 ( 0%)
nsdhdisk9           104857600       -1 no       yes       104787968 (100%)          3904 ( 0%)
nsdhdisk10          104857600       -1 no       yes       104787968 (100%)          3904 ( 0%)
nsdhdisk11          104857600       -1 no       yes       104787968 (100%)          3904 ( 0%)
             -------------                         -------------------- -------------------
(pool total)        838860800                           838303744 (100%)         31232 ( 0%)

             =============                         ==================== ===================
(data)              838860800                           838303744 (100%)         31232 ( 0%)
(metadata)          419430400                           416073984 ( 99%)          1824 ( 0%)
             =============                         ==================== ===================
(total)            1258291200                          1254377728 ( 99%)         33056 ( 0%)

Inode Information
-----------------
Number of used inodes:            4038
Number of free inodes:          496186
Number of allocated inodes:     500224
Maximum number of inodes:      1228864
```


mmlsdisk gpfs1

```
disk         driver   sector     failure holds    holds                            storage
name         type       size       group metadata data  status        availability pool
------------ -------- ------ ----------- -------- ----- ------------- ------------ ------------
nsdhdisk4    nsd         512          -1 no       yes   ready         up           dataPool
nsdhdisk5    nsd         512          -1 no       yes   ready         up           dataPool
nsdhdisk6    nsd         512          -1 no       yes   ready         up           dataPool
nsdhdisk7    nsd         512          -1 no       yes   ready         up           dataPool
nsdhdisk8    nsd         512          -1 no       yes   ready         up           dataPool
nsdhdisk9    nsd         512          -1 no       yes   ready         up           dataPool
nsdhdisk10    nsd         512          -1 no       yes   ready         up           dataPool
nsdhdisk11    nsd         512          -1 no       yes   ready         up           dataPool
nsdhdisk12    nsd         512          -1 yes      no    ready         up           system
nsdhdisk13    nsd         512          -1 yes      no    ready         up           system
nsdhdisk14    nsd         512          -1 yes      no    ready         up           system
nsdhdisk15    nsd         512          -1 yes      no    ready         up           system
```


mmlsnsd

```
mmlsnsd -al

 File system   Disk name    NSD volume ID      NSD servers
------------------------------------------------------------------------
 gpfs1         nsdhdisk4    0AC0820E5CEEBDEF   jlaixn14.abc.local
 gpfs1         nsdhdisk5    0AC0820E5CEEBDF0   jlaixn14.abc.local
 gpfs1         nsdhdisk6    0AC0820E5CEEBDF2   jlaixn14.abc.local
 gpfs1         nsdhdisk7    0AC0820E5CEEBDF3   jlaixn14.abc.local
 gpfs1         nsdhdisk8    0AC0820E5CEEBDF4   jlaixn14.abc.local
 gpfs1         nsdhdisk9    0AC0820E5CEEBDF5   jlaixn14.abc.local
 gpfs1         nsdhdisk10   0AC0820E5CEEBDF6   jlaixn14.abc.local
 gpfs1         nsdhdisk11   0AC0820E5CEEBDF8   jlaixn14.abc.local
 gpfs1         nsdhdisk12   0AC0820E5CEEBDF9   jlaixn14.abc.local
 gpfs1         nsdhdisk13   0AC0820E5CEEBDFA   jlaixn14.abc.local
 gpfs1         nsdhdisk14   0AC0820E5CEEBDFB   jlaixn14.abc.local
 gpfs1         nsdhdisk15   0AC0820E5CEEBDFC   jlaixn14.abc.local
```


mmlsfs gpfs1

```
flag                value                    description
------------------- ------------------------ -----------------------------------
 -f                 8192                     Minimum fragment (subblock) size in bytes (system pool)
                    65536                    Minimum fragment (subblock) size in bytes (other pools)
 -i                 4096                     Inode size in bytes
 -I                 32768                    Indirect block size in bytes
 -m                 1                        Default number of metadata replicas
 -M                 2                        Maximum number of metadata replicas
 -r                 1                        Default number of data replicas
 -R                 2                        Maximum number of data replicas
 -j                 scatter                  Block allocation type
 -D                 nfs4                     File locking semantics in effect
 -k                 all                      ACL semantics in effect
 -n                 32                       Estimated number of nodes that will mount file system
 -B                 262144                   Block size (system pool)
                    2097152                  Block size (other pools)
 -Q                 none                     Quotas accounting enabled
                    none                     Quotas enforced
                    none                     Default quotas enabled
 --perfileset-quota no                       Per-fileset quota enforcement
 --filesetdf        no                       Fileset df enabled?
 -V                 20.01 (5.0.2.0)          File system version
 --create-time      Wed May 29 12:38:15 2019 File system creation time
 -z                 no                       Is DMAPI enabled?
 -L                 33554432                 Logfile size
 -E                 yes                      Exact mtime mount option
 -S                 relatime                 Suppress atime mount option
 -K                 whenpossible             Strict replica allocation option
 --fastea           yes                      Fast external attributes enabled?
 --encryption       no                       Encryption enabled?
 --inode-limit      1228864                  Maximum number of inodes
 --log-replicas     0                        Number of log replicas
 --is4KAligned      yes                      is4KAligned?
 --rapid-repair     yes                      rapidRepair enabled?
 --write-cache-threshold 0                   HAWC Threshold (max 65536)
 --subblocks-per-full-block 32               Number of subblocks per full block
 -P                 system;dataPool          Disk storage pools in file system
 --file-audit-log   no                       File Audit Logging enabled?
 --maintenance-mode no                       Maintenance Mode enabled?
 -d                 nsdhdisk4;nsdhdisk5;nsdhdisk6;nsdhdisk7;nsdhdisk8;nsdhdisk9;nsdhdisk10;
nsdhdisk11;nsdhdisk12;nsdhdisk13;nsdhdisk14;nsdhdisk15  Disks in file system
 -A                 yes                      Automatic mount option
 -o                 none                     Additional mount options
 -T                 /fsgpfs1                 Default mount point
 --mount-priority   0                        Mount priority
```


51

Monitoring Script

- Download script from:
  - http://www.circle4.com/ptechu/gpfsgrabptechu.txt
  - Ensure you test it first on a test system
  - Expects to write to /usr/local/perf but you can change this
- Gathers cluster configuration information
- Runs mmdiag to look at performance items such as waiters, memory, iohist, etc
- Gathers filesystem specific information (tailor to add your filesystems)
- Runs mmpmon to get ios and fsios information

52

Summary

AIX Tunables
- CPU
- Memory
- Number of disk LUNs
- Queue depth on disks
- Fibre adapter num_cmd_elems
- Network tunables
- RAID settings on storage
- Watch the runqueue

Scale Tunables
- Placement of Manager nodes
- Metadata and Data Placement
- Blocksizes
- Scatter block allocation
- workerThreads
- Pagepool
- maxFilesToCache
- maxMBPs
- maxStatCache

Disaster Recovery

Set up remote DR

- This assumes you are using SAN replication to the remote site
- Define the remote cluster with no NSDs
- Configure tunables, etc
- Assuming cluster name is: CLGPFSDR.abc.local
- On primary cluster add a file called gpfs-dr.txt in /usr/local/etc
- It should contain just the name of the remote DR cluster
- Setup reciprocal SSH between the two systems
- Now sync the cluster definitions across for each filesystem (gpfs1 in this case)
  - mmfsctl gpfs1 syncFSconfig -n /usr/local/etc/gpfs-dr.txt
- NSDs will now show on DR system but will not be usable as the disks are not there yet

mmfsctl Example

```
cat /usr/local/etc/gpfs-dr.txt
CLGPFSDR.abc.local
On primary:
mmfsctl gpfs1 syncFSconfig -n /usr/local/etc/gpfs-dr.txt
mmfsctl:  Exporting file system information from the source cluster . . .
mmexportfs: Processing file system gpfs1 ...
mmfsctl:  Importing file system information into the target cluster on node CLGPFSDR.abc.local . . .
mmimportfs: Processing file system gpfs1 ...
mmimportfs: Processing disk nsdhdisk4
mmimportfs: Processing disk nsdhdisk5
……..
mmimportfs: Committing the changes ...
mmimportfs: The following file systems were successfully imported:
mmimportfs: 6027-1155 The NSD servers for the following disks from file system gpfs1 were reset or not defined:
    nsdhdisk4
    nsdhdisk5
mmimportfs: 6027-1157 Use the mmchnsd command to assign NSD servers as needed.
    gpfs1
mmlsnsd now shows the NSDs
```


Failing over

- Make sure DR cluster is not up
- On production cluster need to flush the last storage to storage replication
  - mmfsctl gpfs1 suspend-write
    - Writing dirty data to disk.
    - Quiescing all file system operations.
    - Writing dirty data to disk again.
  - mmfsctl gpfs1 resume
    - Resuming operations.
- Stop replication
- On both systems check status (both should now be down)
  - mmgetstate –a
- On DR System
  - Enable access to replicated disks in DR (break mirror first) and then run cfgmgr followed by lspv
    - lspv
    - hdisk4 none nsdhdisk4
    - hdisk5 none nsdhdisk5
  - Note down disk names
  - Run mmlsnsd and mmlsnsd –m – make sure disk names match what is expected by the NSDs
  - If they don't then create /usr/local/etc/gpfs-dr-nsds.txt with correct definitions
    - mmchnsd -F /usr/local/etc/gpfs-dr-nsds.txt
  - mmstartup
  - Testing

Extend a filesystem

To add disks to a running filesystem

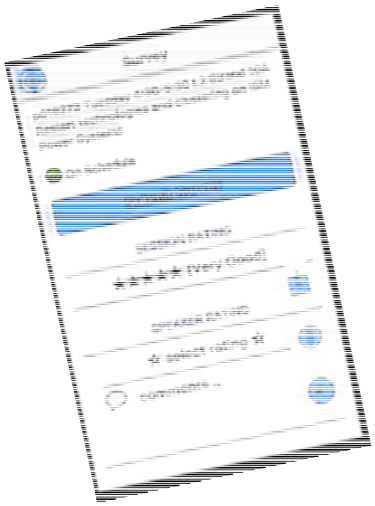
- Add disks to system and run cfgmgr and check you can see them
- Put a PVID on the disks and set queue_depth and reserve_policy
- Create the NSD stanza for the new disks
- Create the NSDs using mmcrnsd
- mmlsnsd | grep free | wc –l
  - The above will show you the number of NSDs that are available to use and it should match the number of disks you are trying to add
- Use mmadddisk to add disks to the filesystem (the -r rebalances the filesystem):
  - mmadddisk gpfs1 -F /usr/local/etc/gpfs1-nsdstanza-new.txt –r
  - It took about 2.5 hours to restripe 3TB of data
  - You may also want to look at mmrestripefs - https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.3/com.ibm.spectrum.scale.v5r03.doc/bl1adm_mmrestripefs.htm
- Run checks
  - mmlsnsd
  - mmlspv
  - mmdf gpfs1
  - df –g | grep gpfs1
- Resync to DR
  - mmfsctl gpfs1 syncFSconfig -n /usr/local/etc/gpfs-dr.txt
  - gpfs-dr.txt contains the name of the DR cluster to be sync'd to
- Now go and add the new NSD definitions to the overall NSD stanza n case you ever have to recreate the filesystem

Thank you!

Jaqui Lynch

jlynch@flagshipsg.com

Please complete the Session Evaluation!



Thank you for your time

If you have questions please email me at:
jaqui@circle4.com or jlynch@flagshipsg.net
http://www.circle4.com/ptechu/spectrumscale-oct0219.pdf

Also check out:
http://www.circle4.com/movies/

Spectrum Scale Specific Useful Links

- Article on Spectrum Scale Performance
  - http://ibmsystemsmag.com/Power-Systems/06/2019/basic-tuning-concepts
- Implementing a single node Spectrum Scale Cluster
  - http://archive.ibmsystemsmag.com/aix/administrator/systemsmanagement/implementing-single-node-cluster/
- Implementing a 3 node Spectrum Scale Cluster
  - http://archive.ibmsystemsmag.com/aix/administrator/lpar/three_node_gpfs/
- Spectrum Scale 5.0.3 Documentation
  - https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.3/ibmspectrumscale503_welcome.html
- Spectrum Scale 5.0.3 release notes
  - https://www-01.ibm.com/support/docview.wss?uid=isg400004570
- Spectrum Scale 5.0.3 Upgrades Presentation at Spectrum Scale User Group
  - https://www.spectrumscaleug.org/wp-content/uploads/2019/05/SSUG19US-Day-1-02-What-is-new-in-Spectrum-Scale-5.0.3.pdf

Useful Links

- Jaqui Lynch Articles
  - http://www.circle4.com/jaqui/eserver.html
  - http://ibmsystemsmag.com/Authors/jaqui-lynch
  - http://www.ibmsystemsmag.com/authors/Jaqui-Lynch/
- IBM US Virtual User Group
  - http://www.tinyurl.com/ibmaixvug
- Power Systems UK User Group
  - http://tinyurl.com/PowerSystemsTechnicalWebinars
- Spectrum Scale User Group
  - https://www.spectrumscaleug.org