IBM Systems MEDIA

Basic Tuning Concepts for a Spectrum Scale Cluster

This article examines AIX tunables that help with Scale performance.

By Jaqui Lynch

06/03/2019

Spectrum Scale (formerly known as GPFS) is a non-blocking filesystem that is used in high performance computing clusters, Oracle environments and many other environments where you want a high performance, shared filesystem that goes beyond what NFS or Samba can provide. In this article we will discuss some of the performance settings that impact Scale performance. We will first look at some basic AIX settings and will then look at Scale Specific settings. Scale settings apply to Scale 4.2.3 or higher unless otherwise stated.

Basic AIX Setup

With AIX, there are a few AIX tunables that help with Scale performance. In particular, you should ensure that your fiber adapters and disks are tuned for queue_depth, You should also check network tunables are correctly set as well.

Fiber adapter settings and hdisk queue_depth settings are specific to the disk vendor. In the case below I was using a v7000 with 16GB HBAs and I set the adapters as follows:

```
chdev -l fcs0 -a max_xfer_size=0x200000 -a num_cmd_elems=2048 -P
```

Each fiber adapter on each node needs to be set and then the node will need to be rebooted for the change to be enabled. If you are using VIO servers, they must be at least as big as the client LPARs and should be changed and rebooted first.

hdisks should have their queue_depth and reserve_policy set and they need to have a PVID on them, otherwise you need to set the values using chdev. For direct attach or NPIV the hdisks are updated on the VIO client. For vSCSI they are updated on the VIO servers. This change also requires a reboot of the node.

```
chdev -l hdisk4 -a pv=yes
chdev -l hdisk4 -a queue_depth=64 -a reserve_policy=no_reserve -P
```

Perform this on every hdisk that will be in the cluster. The changes to settings on the fibre adapters and the hdisks are dependent on what your storage supplier can support so you may need to check with them.

In a multiple node system, occasionally the hdisks show up in a different order so it may be helpful to rename them on the nodes so that they all match. You can do this using rendev. Don’t use rendev to rename the hdisks to anything other than hdisk; Scale doesn’t recognize the disk type when you run mmcrnsd if it isn’t called hdisk.

You should also check the following settings
/etc/security/limits
Look at fsize and nofiles – I usually set them to fsize=-1 and nofiles=20000 or nofiles=-1
This allows for large file sizes and lots of files
/etc/environment
Add /usr/lpp/mmfs/bin to the end of the PATH statement
Add WCOLL=/usr/local/etc/gpfs-nodes.txt

Spectrum Scale Specific Tunables

There are a number of things that can impact Scale performance ranging from poor design to default tunable settings.

RAID Level

One of the first choices you will have to make is the RAID level for the underlying disks. Typically, you will be choosing between raid-10, raid-5 or raid-6. Random write performance and metadata benefit from raid-10, whereas sequential write performance and space efficiency benefit from raid-5. It is possible to separate metadata and data into separate LUNs which allows you to have the metadata set to a smaller blocksize and on raid-10 disks while the actual data is on raid-5 disks that have a different blocksize (if so desired). The only restriction is the two have to live in different Scale pools with the metadata in the system pool. For performance reasons, different RAID types shouldn't be mixed in a filesystem—all the data luns for a filesystem should be raid-5 or raid-6, not a mixture. This is because Scale reads and writes all disks at the same time when performing sequential operations. If the luns are different RAID levels performance will be at the speed of the slowest lun setup. Since Scale uses striping to obtain performance it is important to not just use RAID, but also to provide multiple LUNs so Scale can take advantage of striping across LUNs which also allows you to avoid queue_depth issues. For SAS it is recommended that no fewer than 8 data LUNs be provided, preferably more.

Filesystem Blocksize

The blocksize is the largest I/O that Scale can issue to the device. The block is made up of 32 subblocks, each of equal size. The default blocksize is 256KB, which means 32 x 8KB subblocks. This means that each file will use at least 8KB. Typical blocksize recommendations include:

```
 1-4MB             Large sequential I/O
512KB             Databases such as Oracle or DB2
256KB             Small sequential I/O – file services, etc
```

By default this blocksize is used for both metadata and data.

What Is Metadata?

Metadata is “data about data” and consists of three primary classes: descriptors, system metadata and user metadata.

Descriptors

Descriptors are used to describe an object within Scale. As an example, an NSD (network shared disk) descriptor is a small (4KB) structure that identifies a specific LUN as a Scale NSD and that contains the data necessary to identify that LUN using its NSD ID. NSD descriptors are used by Scale to discover Scale devices. An NSD is Scale’s version of the disk, whether it is shared over the network or directly attached.

Disk descriptors (or labels) are created when an NSD is added to a filesystem and they identify the disk as belonging to a specific filesystem. File system descriptors are similar to a UNIX superblock and are variable length structures that describe the filesystem. They range from a few KB to several MB in size and contain information for the filesystem about the disks, log files, snapshots, etc. Unlike other metadata, descriptors can reside outside the system pool.

System Metadata

System metadata consists of a number of small files. One of these file types is the inode which is a basic structure that describes a single file or directory. By default, the inode size is 4KB and it consists of a fixed 128 byte header, plus data such as disk addresses pointing to data, or indirect blocks, or extended attributes. Every filesystem contains at least 1 inode. The limit on the maximum number of inodes is set at filesystem creation time (see –inode-limit). There is an associated inode map that uses 2 bits per inode to represent the allocation state of each inode in the filesystem. There is also a block allocation map that uses 1 bit per subblock to represent the allocation state of each subblock in the storage pool. There is one inode allocation map per filesystem and one block allocation map per storage pool.

Scale also performs logging or journaling to assist with committing transactions. By default, there are 3 logfiles that are 4MB in size (or the metadata blocksize if it is larger). Blocks in each log are striped across the disks. Insufficient or undersized logs will cause buffer waits which affect performance—the number of logfiles is auto-tuned when setting the workerThreads parameter.

Additional metadata may be present and consists or ACLs (access control lists), extended attributes, quota files, fileset metadata, and policy files (for placement policies). There is also an allocation summary file that contains block and inode usage for each storage pool. It’s updated periodically.

User Metadata

User metadata describes files and directories when the filesystem is in use.

Metadata and Data Setup

For performance reasons. it is often recommended that data and metadata be separated –this allows you to have a

larger blocksize for the data (1MB or 2MB) and a smaller blocksize for the metadata (256KB). The IO pattern for metadata is usually very different from data and metadata is often the bottleneck. Metadata is typically lots of small random reads and writes. It's also much smaller than the actual data so can be put onto much faster expensive media such as flash or SSDs. The metadata LUNs can be set to RAID-10, which avoids the RAID-5 write penalty. Using blocksizes larger than 1MB or smaller than 256KB for metadata is not recommended.

When the filesystem is created a disk can be designated as dataOnly, metadataOnly or dataAndMetadata. This can be changed later using mmchdisk and mmrestripefs, but you can only change the metadata to a different blocksize if metadata is the only thing in the system pool. If you are not sure you want different blocksizes, then make sure to setup a separate pool for data at creation time. The articles in reference 3 on metadata provide more information on metadata and sizing. Typically, metadata is between 1 and 5% of the filesystem space, but this can vary.

Node Types

There are a number of node types possible within a Scale cluster. For the purposes of this article we will look at the four main node types in a general Scale cluster.

The first is the cluster manager. There is one per cluster, although you can define a secondary as backup. The cluster manager monitors disk leases, detects failures and manages recovery, distributes configuration changes, determines quorum, and many other management functions.

There is also a file system manager which acts as an initial token manager. There is one per filesystem. This node requires more daemon memory since token states are initially stored there. That memory is stored in the Scale daemon (mmfsd). The filesystem manager can become very busy if there is a lot of activity, so it is recommended that this be a separate node to the actual data nodes (NSD servers). The special functions of the file system manager consume extra processing time and memory so benefit from being on their own node. Additionally, the NSD servers consume both memory and processor cycles that could impact the operation of the file system manager, so it makes sense to separate them whenever possible.

Quorum nodes are nodes that maintain quorum within the cluster. It's recommended that the cluster managers and filesystem manager act as quorum nodes. There is a maximum of 8 quorum nodes.

Finally, there are the actual data nodes. These can reside on any or all of the above nodes, however, for performance reasons it's best to design the cluster so that the nodes handling all the I/O are doing nothing except handling data.

Important Scale Tunables

Pagepool

The Scale pagepool is used to cache user data and file metadata. This allows Scale to service read and write requests asynchronously. Increasing this increases the amount of data and metadata that Scale can cache without requiring synchronous I/O. The pagepool uses pinned memory so the mmfsd (Scale daemon) will always require at least that much memory and will not release it. The pagepool is set by node—all nodes can be set to the same pagepool size, or they can differ depending on their needs.

Increasing the pagepool helps significantly for large file access, where data is reused, or any time the workload can benefit from prefetching data. It also benefits random I/O workloads. Tests I have run show that a pagepool that is too small significantly impacts performance.

maxFilesToCache

This is the total number of different files that Scale can cache at one time. This reserves memory for the content of the file's inode plus some control data structures. This is separate to the actual data that is also cached. It's particularly beneficial when there are large numbers of files being accessed. The default is 4000 which is far too small for most Scale filesystems. The total needs to be large enough to handle the currently open files plus some caching of recently used files. Memory required for this is maxFilesToCache x 3KB bytes

maxStatCache

This is used to set aside pageable memory to cache additional file attributes. The default is either 1000 or 4 x maxFilesToCache. The memory used by this can be calculated using: maxStatCache x 400 bytes

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, but the other two parameters require shutting down and restarting the Scale daemons.

maxMBpS

This indicates the maximum throughput in MB per second that Scale can submit into or out of a single node. This variable is also used to calculate how many prefetch and writebehind threads should be scheduled for sequential access.

A good starting point is setting this to 2 x the I/O throughput the node can support. This is not a guarantee, but it can restrict I/O bandwidth if set too low.

workerThreads

workerThreads controls several other variables in a Scale system. It is the maximum number of simultaneous local Scale requests. The default is 48 for a base system and 512 if protocols are installed. When this parameter is adjusted it automatically tunes the following parameters:

```
logBufferCount
preFetchThreads
worker3Threads
```

When I changed workerThreads from 48 to 128 the following values were also auto tuned, and a number of additional parameters suddenly showed up in mmdiag with a "." to the left, indicating they are newly visible.

```
logBufferCount
    preFetchThreads was set to 72
    worker3Threads stayed at 8
    worker1Threads went to 128 from 48
```

seqDiscardThreshold

For certain workloads (SAS is an example) it can be helpful to increase this to a value larger than the largest file you want Scale to cache. This is particularly helpful if multiple threads need to sequentially read the same file on a node.

Block allocation

There are two options for block allocation – cluster and scatter. The default for fewer than 8 nodes or 8 NSDs is cluster, however it is not recommended for larger configurations or for random allocation patterns. The block allocation for a filesystem is set at creation time so it is recommended that you use scatter, which avoids you having to rebuild the filesystem as you add NSDs or nodes.

--inode-limit

The inode limit should be increased from the default if you plan to support a large number of files in the filesystem. You can estimate a value for this using the following calculation:

```
(metadata_disk_size * #metadatadisks) / (inodesize * defaultMetadataReplicas)
```

You can use "df -i" or "mmdf filesystemname -F" to see how many inodes are free and "mslsfs filesystemname –inode-limit" to check the current limit. This value can be increased dynamically using mmchfs.

prefetchPCT

In a SAS environment consider increasing this to 40% (from 20%) as a great deal of the SAS data access is

sequential and this will cause the pagepool to do more prefetching.

Settings I Typically Use

On a large system (256GB or more of memory and several nodes and many cores) the settings I currently start with for a SAS workload are:

```
 pagepool 64G
seqDiscardThreshhold 4G
maxMBpS 15000
prefetchPct 40
maxFilesToCache 200000
workerThreads 256
```

I set block allocation type to scatter and ensure I have more than 8 luns per filesystem for data. I also split data and metadata to separate LUNs with different blocksizes. Your settings will vary depending on workload

Commands to Help Make Performance Decisions

gpfsperf

gpfsperf is a 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. I've found the best way to get valuable data is to run multiple copies on multiple nodes at the same time—this tests concurrency within a node as well as concurrency between nodes.

mmpmon

mmpmon is used to collect I/O statistics for each mounted filesystem or for the whole node. It should be 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.

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

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

As of Scale 5.0.2 there is a new command that allows you to look at how the filesystems and pagepool are being used. The command is “mmcachectl show” and you can look at overall use or by device, fileset, inode or filename.

Summary

In this article we have touched on just a few Spectrum Scale parameters that should help with your performance. However, there are significantly more parameters that can impact performance, especially when you add additional node types (such as protocol nodes) into the mix. It is recommended that you start by looking at the output from the various commands listed above to determine where your problems may lie. In particular, the mmdiag and mmdump parameters should be very helpful.

References

For more information:

1. Scale 5.0 administration guide

https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.0/com.ibm.spectrum.scale.v5r00.doc/pdf/scale_adm.pdf

(https://www.ibm.com/support/knowledgecenter/STXKQY_5.0.0/com.ibm.spectrum.scale.v5r00.doc/pdf/scale_adm.pdf)

2. Scale 4.2 advanced administration guide

https://www.ibm.com/support/knowledgecenter/STXKQY_4.2.0/c2370326.pdf (https://www.ibm.com/support/knowledgecenter/STXKQY_4.2.0/c2370326.pdf)

3. Metadata

https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/General+Parallel+File+System+(GPFS)/page/Data+and+Metadata (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 (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 (https://tinyurl.com/gpfsmetadata)

4. GPFSPERF

https://www-01.ibm.com/support/docview.wss?uid=isg15readmebbb63bf9mples_perf (https://www-01.ibm.com/support/docview.wss?uid=isg15readmebbb63bf9mples_perf)

5. mmpmon

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_mpmover.htm)

https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.2/com.ibm.spectrum.scale.v5r02.doc/bl1adv_aganfior.htm (https://www.ibm.com/support/knowledgecenter/en/STXKQY_5.0.2/com.ibm.spectrum.scale.v5r02.doc/bl1adv_aganfior.htm)

About the author
Jaqui Lynch has over 38 years of experience working with a projects and OSes across vendor platforms, including IBM Z, UNIX systems and more.

Related Content

Application development (/application-development) Service Programs and Signatures → (https://ibmsystemsmag.com/Power-Systems/10/2003/service-programs-signatures)
Systems management (/systems-management) A Look at File Systems → (https://ibmsystemsmag.com/Power-Systems/09/2004/file-systems-commands)
Systems management (/systems-management) Accessing the Data in Core Dumps → (https://ibmsystemsmag.com/Power-Systems/01/2006/core-dumps-data-access)