Paper 484-2013

A Survey of Shared File Systems: Determining the Best Choice for Your Distributed Applications

Barbara Walters, Ken Gahagan,

Leigh Ihnen, and Vicki Jones, SAS Institute Inc.

ABSTRACT

A shared file system is an integral component of all SAS® Grid Manager deployments, Enterprise Business Intelligence deployments with load balanced servers on multiple systems, and other types of distributed SAS applications. This paper explains how SAS software interacts with the file system and how a shared file system behaves differently from a non-shared file system. It describes what factors will determine whether a particular file system is a good fit for your environment and how to choose the file system that best meets your needs.

INTRODUCTION

A shared file system is a required and integral component of all SAS® Grid Manager deployments, Enterprise Business Intelligence deployments with load balanced servers on multiple systems, and other types of distributed SAS applications. In order to determine which shared file system is the best choice for a given deployment, it is important to understand how SAS software interacts with the file system and how a shared file system might behave differently compared to a non-shared file system.

Here are the important characteristics of shared file systems with respect to SAS performance:

- whether the file system data is retained in memory in a local file cache
- handling of file system metadata
- implications for the physical resources

This paper briefly describes these aspects and examines the behavior of several shared file systems in the context of performance of a representative deployment.

RETAINING FILE SYSTEM DATA IN MEMORY

Data read from storage such as a solid state device or disk is referred to as a physical read. When the data is retained in and read from the local file cache, the reads are called logical reads. Programs that perform logical reads from data that has already been read physically into memory perform substantially better than when the program performs physical reads.

SAS programs often read the same data multiple times. If the data are retained in a local file cache, the program can perform logical reads from the local file cache much faster and more efficiently than re-reading it from the physical device.

Operations that access small files tend to get a substantial performance benefit from the local file cache. Files that are larger than the size of the cache memory cannot be cached in their entirety and might not be cached at all.

Most non-shared file systems will retain data in the local file cache until the memory is needed for some other purpose. In comparison, shared file systems implement a wide variety of rules that govern the retention of data in the local file cache. Some file systems allow tuning of the local file cache, providing options for limiting the amount of memory that can be used for file cache or the number of files kept in the local file cache, for example.

MAINTENANCE OF FILE SYSTEM METADATA

File system metadata includes such information as lists of files in a directory, file attributes such as file permissions or file creation date, and other information about the physical data.

Various shared file systems differ in the maintenance of the file system metadata. Shared file systems are required to make information about file system metadata and file locking available to all systems participating in the shared file system. File system metadata is updated whenever a file is created, modified, deleted or extended, when a lock is obtained or dropped, and on some file systems, when a file is accessed.

The sizes of files created by SAS programs are not known when files are first created. As the file is written, additional file system blocks, or extents, are requested. Each extent request results in an update to file system metadata. In a shared file system, file system metadata changes are coordinated across systems, so there is more overhead associated with file system metadata with a shared file system compared to a non-shared file system.

Due to the differences in local file cache behavior and the additional overhead required to maintain file system metadata, programs that performed well with a non-shared file system might behave quite differently with a shared file system. To minimize overhead from the shared file system, a general recommendation is to maintain SAS WORK directories in a non-shared file system and place only permanent data in a shared file system. A shared file system might be necessary for GRIDWORK.

SAS WORK files are not usually shared and are considered to be private to the process that created the directory, so there is no benefit for SAS WORK files to reside in a shared file system. SAS WORK files also tend to generate a higher number of file system metadata updates because files are continually created, extended, and deleted, which can greatly increase the overhead of coordinating file system metadata in the shared file system. In addition, a non-shared file system is more likely to retain SAS WORK files in the local file cache.

IMPLICATIONS FOR PHYSICAL RESOURCES

Shared file systems require coordination between multiple host systems and place more demand on the network and storage devices. This means that the network and storage devices must be provisioned appropriately. In addition, less efficient file cache management might result in more data transfer and more metadata requests, both of which might result in more physical I/O. The storage subsystem might need to support both a higher number of I/O operations as well as a higher throughput rate compared to the needs of a non-shared file system.

WORKLOADS

In order to characterize different shared file systems and storage devices, we developed a small set of internal benchmarks to exercise typical SAS software scenarios. The following is a description of two of the workloads that are being used to study a variety of shared file systems. The outcomes for each of those shared file systems are described later in the paper.

LARGE SORTS

Large sorting workloads measure throughput of the file/storage subsystem. This workload is intended to push the I/O subsystem bandwidth to its limit. The workload launches multiple simultaneous sorts, typically 1 to 2 per CPU core. The files used by this workload are typically larger than system memory and will not be retained in the local file cache. File system metadata is exercised lightly.

DEMAND MODEL FORECASTING AND CALIBRATION

This workload runs a configurable number of concurrent SAS processes that execute a demand model calibration process. The workload emulates a SAS solution deployed in a distributed environment using shared input data and a shared output analytic data mart. The SAS programs are written using SAS macro language and create many small to moderate data sets.

There are 750 models in the workload and each model is calibrated by an individual SAS process. Each of these processes creates about 100 permanent output data sets and 7,000 temporary files. Many solutions, including SAS® Drug Development, SAS® Warranty Analysis, SAS® Enterprise Miner™, and others behave in a similar manner. This workload will exercise the management of file system metadata and some shared file systems have failed with what we consider a representative load. This workload tends to benefit a great deal from the file cache, because there are many small files that are read multiple times.

CONFIGURATION

It is important to configure your system appropriately before running SAS applications. Below we discuss file system and SAS options used for our shared file system studies.

FILE SYSTEM CONFIGURATION

Some shared file systems are available as NAS (network attached storage) rather than SAN (storage area network). With 10 GbE and other high bandwidth network connectivity, NAS devices have become much more attractive because much higher throughput rates can be achieved compared to lower bandwidth connectivity such as 1 GbE. However, even with the higher bandwidth networks, throughput might not be sufficient for systems with a high core count. Systems with 12 or more cores might not be fully used due to constraints and latency of the network.

Some file system options exist that enable or disable recording when a file is accessed. These are available on most systems that are based on UNIX or Windows. Unless the workload specifically requires information about when a file was accessed or modified, disabling the metadata update with shared file systems reduces some of the load on the file system. On systems that are based on UNIX, mounting file systems with the option NOATIME= disables the recording of file access times. The mount option NOATIME= improved the performance of all benchmarks and is a recommended setting if accurate file access time is not required for correct program execution.

On Microsoft® Windows operating systems, setting the registry value

```
HKLM\System\CurrentControlSet\Control\Filesystem\NtfsDisableLastAccessUpdate
```

to 1 also disables this type of file system metadata update.

SAS SYSTEM OPTIONS

Where possible, the SAS options are set to coordinate well with the file system page size and device stripe size. The option ALIGNSASIOFILES aligns all data sets, including SAS WORK data sets on the specified boundary. It performs the same alignment as the LIBNAME option OFFSET specified for non- WORK libraries.

Other options align SAS I/O requests to the boundary specified by the buffer size. I/O alignment may reduce the number of physical I/O operations. For shared file systems, these options are particularly useful because many shared file systems will not coalesce unaligned I/O requests. Misalignment might cause a much higher number of requests, either over a network or to the I/O device, compared to aligned requests.

The following is a description of the SAS® Foundation 9.3 options used to tune performance of the benchmarks.

```
CPUCOUNT=2
```

These benchmarks all ran multiple SAS processes concurrently. The option CPUCOUNT was set to 2 in order to minimize contention for CPU resources.

```
MEMSIZE=1G
```

This option MEMSIZE was set to 1G minimize contention for memory resources.

```
SORTSIZE=768M
```

The option SORTSIZE was set to 75% of memsize.

```
ALIGNSASIOFILES
```

The option ALIGNSASIOFILES aligns access to SAS data sets on a specific boundary. The boundary is defined by BUFSIZE. This allows a file system to align read and write requests and avoid split I/O operations.

```
BUFSIZE=128k
```

The option BUFSIZE sets the read/write size of each request made to SAS data sets.

```
UBUFSIZE=128k
```

The option UBUFSIZE sets the read/write size of SAS utility file requests.

```
IBUFSIZE=32767
```

The option IBUFSIZE sets the read/write size of SAS index file requests.

```
IBUFNO=10
```

Some of the workloads use indexes. Setting IBUFNO to 10 reduced the number of file system requests for indexed traversal.

```
FILELOCKS=none
```

By default, on UNIX systems the SAS option FILELOCKS=FAIL is set. This setting obtains read and write file locks on non- WORK files to protect programs from making concurrent writes to a file or from reading data that might be in an inconsistent state. File systems make metadata requests in order to manage locks. The benchmarks were run with both FILELOCKS=FAIL and FILELOCKS=NONE in order to determine the overhead associated with file locks.

In order to reduce metadata requests, there are SAS options that eliminate file lock requests. For data that is not shared, is only read, or if the application is designed not to need file locks, setting the option FILELOCKS=NONE will reduce file system metadata requests. Also, some file systems will flush files from memory when any file attribute changes. This means files accessed with a lock might be forced out of the local file cache and re-read from storage.

The overhead from managing file locks and the change in file cache retention can be substantial. Therefore, limiting the use of file locks to just those files and libraries that require concurrent read/write access is recommended for best performance. File locks are necessary to avoid corrupting data or accessing data in an inconsistent state. Here are examples of situations that require the use of file locks:

- concurrent processes attempting to update the same file
- a single process updating a file and other processes attempting to read it concurrently.

We are in the process of running these workloads on a variety of shared file systems. The following sections describe the file systems benchmarked as of April 2012.

FILE SYSTEM PERFORMANCE STUDIES

In this section we discuss the shared file systems studied to date. We plan to study additional file systems and report those results also.

IBM GENERAL PARALLEL FILE SYSTEM™ (GPFS™)

The IBM General Parallel File System™ (GPFS™) performed well on both Red Hat® Enterprise Linux® (RHEL) and Microsoft® Windows operating systems. Both permanent data and SAS WORK files were managed by GPFS with excellent throughput and low overhead for file system metadata management.

GPFS requires dedicated system memory and the amount of memory is defined by the file system option PAGEPOOL=. The client systems used in the benchmarks have 96 GB of memory, and 32 GB of it was dedicated to the page pool. It is likely that client computer systems that use both a non-shared file system and GPFS will require additional memory to allow both file systems to perform adequately.

The workloads were run with several GPFS configurations on RHEL 5.7 and 6.1 and Windows Server 2008 R2. Also, GPFS is deployed by many customers with SAS software on AIX. For RHEL, two configurations were run: a NAS configuration and a SAN configuration. The NAS configuration used a DataDirect™ Networks SFA10K-E™ GridScaler™ Appliance connected via 10 GbE with multiple GPFS clients running on RHEL 6.2 and Windows Server 2008 R2. The SAN configuration had SAS software running on the same systems as the GPFS NSD (network shared disk) servers using fibre channel connections to a storage device. Both configurations performed well. There was no measurable overhead using the default setting FILELOCKS=FAIL.

For both the SAN and NAS configurations, GPFS was able to transmit data at a rate limited only by the throughput rate of the physical device. This rate was sustained during the large sort workload and achieved periodically for the calibration workload. The calibration workload ran 100 to 250 concurrent SAS processes spread over six 12-core systems. Maximum throughput was achieved and sustained and file system metadata overhead stayed at a manageable and responsive level.

SAS data access tends to be sequential. The GPFS file system was created with the "cluster" allocation method so that data blocks would be allocated close to one another. The documentation for GPFS specifies that cluster allocation should be used for moderate configurations, but it behaved much better for both workloads compared to the "scatter" allocation method.

GPFS allows separation of file system metadata from file system data. For SAS software, the access pattern to file system metadata is dramatically different from the access pattern for data. File system metadata tends to be random access of small blocks, and file system data access tends to be sequential access of larger blocks. We found that creating separate volumes for file system metadata and using solid state devices (SSDs) provided a significant improvement in the responsiveness of the file system.

The file system was created with a block size of 256 KB. Because the SAS system options BUFSIZE=128k and ALIGNSASIOFILES were specified, each data set runs a minimum of 256 KB, with the first block containing SAS header information and the next block containing data.

The following GPFS options were changed from the default values.

```
maxMBpS=
```

The GPFS option MAXMBPS= was set to approximately twice the bandwidth available to any single system. For example, for the systems connected via 10 GbE, the value was set to 2000 (2 GB/sec).

```
pagepool=
```

The SAS software and all data resided in GPFS, so system memory was primarily used for either SAS processes or GPFS file cache. The GPFS option PAGEPOOL= determines the size of the file cache and was set to 32 GB.

```
seqDiscardThreshold=
```

The GPFS option SEQDISCARDTHRESHOLD= limits the size of a file that is read sequentially and will be retained in the file cache. This option was set to 4000000000 (approximately 4 GB). This value was chosen because the input data for the calibration workload is shared among many processes. If one process has already read the file, other processes can consume the same data without requiring physical I/O. However, if processes are not sharing data, a smaller value is recommended so that a few large files will not dominate the file cache.

```
prefetchPct=
```

Because SAS software usually reads and writes files sequentially, we increased the GPFS option PREFETCHPCT= from the default 20% to 40%. This option controls the amount of the page pool that is used for read-ahead and write-behind operations.

```
maxFilesToCache=
```

The calibration workload creates a high number of relatively small SAS WORK files. The GPFS option MAXFILESTOCACHE= was increased from the default 1000 to 60000. The system should be monitored using a program like MMPMON to determine the appropriate value for this option.

Overall, this file system behaved very well, with both the file cache benefiting performance and little overhead for maintaining file system metadata. The measured workloads had both permanent and SAS WORK files managed by GPFS.

There was one problem found with GPFS 3.4.0.10 on Windows and is documented in PMR 85347,756,000. IBM provided a temporary fix, which enabled all workloads to complete successfully. The permanent fix is now available.

NFS

NFS client and server implementations show a wide variety of behavior that affect performance. For that reason, the specific client and server should be measured to ensure performance goals can be met.

The NFS client maintains a cache of file and directory attributes. The default settings will not ensure that files created or modified on one system will be visible on another system within a minute of file creation or modification. The default settings might cause software to malfunction if multiple computer systems are accessing data that is created or modified on other computer systems. For example, if a workspace server on system A creates a new SAS data set, that file might not be visible on System B within one minute of its creation. In order to ensure a consistent view of the file system, the mount option ACTIMEO= (attribute cache timeout) should be set to 0. This setting will increase the number of requests to the NFS server for directory and file attribute information, but will ensure that the NFS client systems have a consistent view of the file system.

File data modifications might not be visible on any NFS client system other than the one where the modifications are being made until an NFS commit is executed. Most NFS clients will issue a commit as part of closing a file. If multiple systems are reading files that might be modified, file system locking should be used. This is controlled by the SAS system option FILELOCKS=, the SAS library option FILELOCKS=, or the SAS LOCK statement.

In order to ensure file data consistency, when an NFS client detects a change in a file system attribute, any data in the local file cache is invalidated. The next time the file is accessed, its data will be retrieved from the NFS server. This means that retention in the file cache might have much different behavior with an NFS file system compared to other file systems and the file system storage devices and network must be provisioned to handle a larger demand compared to either a local file system or a shared file system that uses a different strategy for cache coherence.

Some NFS clients treat locks as file attributes. Therefore, obtaining any lock including a read/share lock might invalidate any data that is in the local file cache and force the NFS client to request file data from the server. On UNIX systems, files in the SAS WORK directory are accessed without file locks, because those files are assumed to be private to a process. By default, the SAS system requests read and write file locks for all other files. For best performance, it is important to use locks only when necessary in order to avoid invalidating the file cache and making additional file data requests of the NFS server. File locks are needed when data can be updated by multiple concurrent processes or updated by one process and read by other concurrent processes. File locks prevent data

corruption or processes from reading data in an inconsistent state. The usage of file locks with NFS might have a much greater effect on overall system performance compared to other file systems. This behavior should be considered when sizing the I/O subsystem.

NFS performed reasonably well on RHEL 5.7 and 6.1 in a variety of configurations and storage devices. The clients and servers were connected with 10 GbE, which provided sufficient bandwidth and throughput for reasonably high CPU utilization.

The large sort workload gets very little benefit from the local file cache, so its performance did not change with either the SAS FILELOCK default setting or with FILELOCKS=NONE. The calibration workload gets a large benefit from the local file cache and the overall elapsed time for the job increased by 25% or more with the SAS default setting for file locks compared to running with FILELOCKS=NONE. Please note that file locks were requested only for permanent data and that SAS WORK files were accessed with no file locks. For SAS programs that do not rely on file system locks to prevent multiple concurrent read/write access to a file, we recommend FILELOCKS=NONE to improve the NFS's local file cache behavior. With FILELOCKS=NONE, file cache retention is improved.

The benchmarks were run on a variety of devices including EMC® Isilon® and Oracle® Exalogic. In general, there are few tuning options for NFS, but NFS benefits from isolating file system metadata from file system data. Some storage devices like Isilon allow this isolation. Excellent results were obtained from devices that both isolate file system metadata and use SSDs for file system metadata.

Some NFS devices generated errors under the load generated by the calibration workload. SAS Research and Development is working with several vendors to resolve these issues. SAS Technical Support is aware of these issues and can be contacted to determine whether there are any known issues with any particular device.

GLOBAL FILE SYSTEM2

Global File System 2 (GFS2) is a cluster file system supported by Red Hat. GFS2 is provided in the RHEL Resilient Storage Add-On. Note that Red Hat strongly recommends an architecture review be conducted by Red Hat prior to the implementation of a RHEL Cluster. A previous release of this paper indicated performance concerns with GFS2 however improvements delivered with RHEL 6.4 + errata patches through mid-March 2013 have provided marked enhancements to the performance of GFS2. If SAS WORK directories need to be shared then the recommended configuration is to place on a different GFS2 file system than the GFS2 file system to store shared permanent files.

GFS2 is limited to 16 systems in a cluster.

The workloads were run using the tuned utility with the "enterprise-storage" profile. This configures the system to use the deadline I/O scheduler and sets the dirty page ratio to 40 by setting the option VM.DIRTY_RATIO= to 40. These settings greatly improved the performance of the workloads.

To improve the behavior of the Distributed Lock Manager (DLM), the following changes were made:

```
echo 16384 > /sys/kernel/config/dlm/cluster/dirtbl_size

echo 16384 > /sys/kernel/config/dlm/cluster/rsbtbl_size

echo 16384 > /sys/kernel/config/dlm/cluster/lkbtbl_size
```

These commands must be executed after the cman and clvmd services are started and before the file system is mounted.
A good place to apply this tuning is in /etc/init.d/gfs2.

Use LVCHANGE-R <the value should be appropriate for the workload> to set read-ahead for the file system.

For optimal GFS2 performance run RHEL 6.4 and ensure that all patches are applied at least through mid-March 2013. These errata provide fixes to the tuned service as well as address a concern with irqbalance. With RHEL 6.4 and the mid-March 2013 errata SAS found that the workload performed very well on GFS2. This version of GFS2 has not displayed rapid performance degradation observed with prior versions.

The other shared file systems measured on the RHEL platform were GPFS and NFS. GPFS is an excellent choice when there is a need for extreme performance or the need to have greater than 16 nodes in the cluster. GFS2 performance has improved markedly and is now a very good choice for clusters up to 16 nodes. NFS is an option when performance is not a primary concern.

COMMON INTERNET FILE SYSTEM (CIFS)

Common Internet File System (CIFS) is the native shared file system provided with Windows operating systems. With recent patches, CIFS can be used for workloads with moderate levels of concurrency and works best for workloads that get limited benefit from the local file cache. The recommended configuration is to place SAS WORK directories on a non-CIFS file system and use CIFS to manage shared permanent files.

With the release of the Windows Vista operating system in 2006, many improvements were made to address performance issues, and connectivity via 10 GbE greatly improves the throughput and responsiveness of the CIFS file system. With changes made both to SAS Foundation 9.3 software and Windows Server 2008 R2 operating system, CIFS is functionally stable. However, benchmark results showed there was relatively poor retention of data in the local file cache. Workloads that reuse data from local file cache will not perform nearly as well with CIFS compared to a local file system.

The benchmark configuration had three systems running the Windows Server 2008 R2 operating system, one acting as file server and two systems as clients all connected via 10 GbE.

In order to function properly and perform optimally, the following registry settings, Microsoft patches, and environment variable settings used by SAS 9.3 software should be used:

- `[HKLM\SYSTEM\CurrentControlSet\Control\SessionManager\Executive]`
  `AdditionalDelayedWorkerThreads = 0x28`
  `AdditionalCriticalWorkerThreads = 0x28`
- `[HKLM\SYSTEM\CurrentControlSet\Services\lanmanworkstation\parameters]`
  `DirectoryCacheLifetime=0x0`
- FileNotFoundCacheLifetime=0x0Hotfix referenced by http://support.microsoft.com/default.aspx?scid=kb;EN-US;974609
- Hotfix referenced by http://support.microsoft.com/kb/2646563
- Environment variable SAS_NO_RANDOM_ACCESS = 1

Hotfix 2646563 disabled file read-ahead under some circumstances. Microsoft released an additional patch in April 2012 to address this issue.

Comparing CIFS to other shared file systems measured on the Windows platform, GPFS had much better throughput, local file cache retention and metadata management compared to CIFS. This translated to better scalability, and GPFS can serve a larger number of client systems compared to CIFS.

QUANTUM STORNEXT®

Quantum StorNext® demonstrated relatively poor local file cache retention and performs best for workloads that get minimal benefit from the local file cache. A recommended configuration is this: SAS WORK files should be served from the local file system while SAS permanent files can be managed by StorNext.

The benchmark configuration had three 8-core client systems running Microsoft® Windows Server® 2008 R2 operating system. The sort workload behaved well because its files are too large to cache. The calibration workload ran successfully but performed poorly because this workload benefits from local file cache reuse.

In general, GPFS had much better cache retention and metadata management compared to StorNext. For workloads that benefit from local file cache, GPFS will be able to serve a larger number of client systems compared to StorNext.

CONCLUSION

This paper contains only the file systems that we have tuned and measured in our performance lab. Other shared file systems have been used successfully in distributed deployments, such as Veritas™ Cluster File System. We plan to continue running benchmarks against other configurations and will publish the results as they become available.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author:

Barbara Walters
SAS Campus Drive
SAS Institute Inc.
E-mail: Barbara.walters@sas.com