Paper 285-28

SAS® System on Network Appliance

Darrell Suggs, Network Appliance Inc.
Margaret Crevar, SAS Institute
Leigh Ihnen, SAS Institute

ABSTRACT

The goal of this paper is to help customers maximize SAS application performance while reaping the benefits of network attached storage. Specifically, this is a guide for performance tuning in an enterprise class environment containing the SAS application and using a Network Appliance NAS (network attached storage) subsystem (aka Filer). The recommendations include configuration and parameter changes to the SAS environment, the platform specific Unix operating system, NFS (Network File System) client, and the NetApp Filer subsystem. This version of the paper covers only Solaris and HP-UXchanges. A complete set of tuning recommendations for other operating systems is contained in the full version of this paper located on both SAS's and Network Appliance's company web pages.

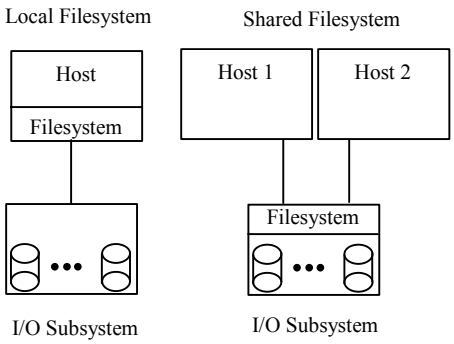
INTRODUCTION

This document is divided into two major sections. The first section discusses concepts and issues important to performance in a SAS./NetApp environment. The second section contains specific recommendations involving SAS deployment issues as well as operating system specific configuration recommendations.

LOCAL VERSUS SHARED FILESYSTEMS

There are two filesystem paradigms of interest for SAS deployment.

Modern Unix platforms offer two paradigms for filesystem deployment: local filesystems and shared filesystems.



These options are also referred to as "direct-attach or SAN" (local filesystem) or "network attached storage – NAS" (shared filesystem). Commercially these paradigms often lead to different storage subsystem products with different costs, capabilities, and scalability. Conceptually, they have only a couple of differences. For example, shared filesystems allow more efficient use of storage resources by enabling multiple hosts to access data while keeping the data management responsibilities (and costs) off the host system.

The shared filesystem solution (NAS) scales to multiple hosts in a simple fashion that introduces very little overhead.

Shared filesystems (NAS) provide flexibility and increase storage efficiency for SAS environments.

SAS PERFORMANCE

SAS application performance is a complex topic.

SAS is a powerful and diverse application. Specifying a "canonical" workload generated by SAS is impossible. However, at the core, SAS's main function is to load, manipulate, and store data. In any given SAS deployment, the underlying compute platform, operating system and I/O infrastructure work together to service these data manipulation operations.

SAS performance is typically measured in terms of run time or "wall-clock time". Stated simply, the goal is to complete each SAS job in as little time as possible. There are two main components to performance in a SAS deployment: computational capability and I/O subsystem performance. An optimally tuned deployment requires a balanced consumption of these two resources.

SAS Workloads Differ from Database Workloads

SAS performs most operations by doing large block (mostly) sequential I/O. This workload has very different characteristics than a typical Database On-Line Transaction Processing (OLTP) workload. As a result, typical performance tuning techniques that improve performance for OLTP (and other database) workloads are not necessarily applicable (or even positive) in SAS environments.

Exploring computational capabilities is beyond the scope of this document.

The first component in SAS performance is computational speed. Each platform has a CPU and memory subsystem with some level of capability. These capabilities vary among platforms and range from slow, inexpensive, single CPU platforms (e.g. PCs) to large, expensive, multiple CPU systems (e.g 32 CPU Solaris platforms). Exploring the performance comparison of various computational platforms is beyond the scope of this document.

SAS application performance is very dependent upon I/O subsystem performance.

The second component in SAS performance is I/O subsystem performance. SAS workloads that manipulate large amounts of data are typically constrained more by I/O performance than by computational capabilities. Inspecting SAS I/O performance closely reveals several important performance characteristics.

SAS I/O MODEL

Modern compute platforms provide two broad classes of I/O interfaces: filesystems and raw devices. The SAS I/O model depends upon the filesystem interface technology. This means that in any deployment, in addition to the raw device or underlying disk platform, SAS requires a filesystem interface and SAS I/O performance is dependent upon the performance of that interface.

Megabytes per Second

The most obvious characteristic of I/O performance is bandwidth. Bandwidth is typically stated in terms of megabytes per second (MB/s) the I/O subsystem is capable of delivering to the SAS application.

CPU Cost of I/O

The next characteristic of I/O performance is the CPU cost of doing I/O. The goal is to achieve high bandwidth levels at very low CPU utilizations. Although the absolute cost of I/O varies among host platforms, there are some general characteristics. Basically, the lowest cost typically comes from native local filesystem implementations, followed by native NFS implementations, and finally third party filesystem implementations.

I/O Caching

The final and most subtle characteristic of I/O performance involves I/O buffer caching on the host platform. The highest performing I/O solution is one that minimizes the actual amount of physical I/O traffic. For instance, a SAS job with a dataset that is smaller than the amount of host memory needs to only fetch the data from the storage subsystem once. All subsequent accesses to that data can be serviced from the host buffer cache, therefore minimizing I/O traffic and maximizing performance.

The concepts and techniques associated with I/O buffer caching are well understood. Unfortunately, the specific implementation and algorithm choices made from platform to platform vary widely. Additionally, a single platform may perform different caching techniques depending on which filesystem implementation is being used (e.g. UFS vs NFS). This variance in I/O caching can provide interesting opportunities and challenges to maximizing I/O performance in a SAS environment. Beware of performance issues that are a result of caching techniques.

SAS DEPLOYMENTS

SAS NFS performance tuning techniques depend upon the specific deployment.

SAS NFS deployments have several important characteristics: SAS version, single host versus multiple hosts configuration, number of SAS sessions per host and host memory size versus SAS data set size. This section outlines each of these characteristics

SAS Versions

SAS currently has three major releases of interest: 6.12, 8.2, and 9.0. Additionally, for some Operating System platforms there are multiple SAS versions based on the underlying compute architecture. From a performance standpoint there are several factors to consider in selecting (or changing) SAS versions.

SAS Work

Most SAS programs depend heavily on the ability to create and access temporary data files quickly and effectively. The location of this temporary data (referred to typically by the directory name "SAS Work") is configurable via the SAS configuration file. The data files created and accessed in SAS Work are not shared among multiple SAS sessions or multiple hosts. Specifically, the data files in SAS Work are specific to a given SAS user session. Given this property of SAS Work, the specifics of where SAS Work is located and which locking and caching options are set strongly impacts overall SAS performance.

For SAS NFS deployments SAS Work can be located on the NFS server (as opposed to placing SAS Work on local storage). In general, SAS Work should be accessible via an NFS mount point that is separate from other SAS data. This allows flexibility in specifying the locking and caching behavior of the SAS Work files.

Single versus Multiple SAS Sessions

On a given host running the SAS application, there can be either a single SAS session (or user) or multiple SAS sessions. The performance and scalability of multiple sessions is dependent upon the underlying platform's capabilities. A full exploration of how many SAS sessions a given platform can support, while important, is beyond the scope of this discussion.

There are however several aspects of multiple SAS sessions that are important for NFS deployments. Specifically, whether or not the multiple sessions share the same or different SAS Work space is an important issue. Additionally, the amount of memory each SAS session uses and as a result, the amount of memory left for operating system caching is an important issue. The optimal resolution of these issues is specific to the overall deployment architecture and is discussed in a later section.

Single versus Multiple Host Deployments

SAS deployments are sometimes associated with very large data sets. Accessing and processing this data through a single host can be cost prohibitive and sometimes impossible with today's Unix compute platform technology. As a result, when shared filesystems are used, these deployments often involve multiple host computers processing the data simultaneously. There are several techniques for deploying and accessing/processing the data simultaneously. A full discussion of these techniques is beyond the scope of this discussion.

There are however several aspects of multiple SAS hosts deployments that are important for NFS deployments. The main issue is the applicability of locking and caching techniques that enhance single host performance. A secondary issue is the placement and usage of SAS Work directories for each individual host. The optimal resolution is again specific to the overall deployment.

SAS performance can be strongly affected by host memory size, SAS memory configuration settings, and data set size.

Each SAS session specifies (or takes a default) configuration file that specifies several options for the session. Of particular interest for I/O performance are:

- memsize: specifies how much host memory a given SAS session is allowed to use
- sortsize: specifies how much data to sort at one time (each sort is broken into multiple load, sort, store phases)
- maxmemquery: specifies how much memory a query is allowed to use.

In addition to the SAS settings, performance also depends on the amount of host memory, the amount of memory used by the application, and the amount of memory left for the operating system to allocate to the buffer cache. Note also that multiple SAS sessions can be running on a single host, increasing the total memory used for the SAS application and datasets.

Selecting the optimal setting for memory settings depends on the relationship of the SAS dataset size to the host memory size. Consider a simple, single SAS session host configuration. For purposes of this discussion, host memory is divided into three categories: SAS application memory, OS buffer cache, and "other". The SAS application memory is limited by the configuration variable memsize. In general, the operating system requires some memory for normal OS function, and allocates the rest as buffer cache. The OS buffer cache is used to hold recently read and pre-fetched data as well as recently written data.

This simple memory usage description results in two classes of SAS data sets: data sets that are smaller than host memory and data sets that are larger than host memory. For datasets that are smaller then host memory, increasing the SAS memory variables to values larger than the data set size will result in minimal I/O and maximum performance. For data sets larger than host memory, reducing SAS memory consumption and thereby allowing more memory for the OS buffer cache increases prefetch effectiveness and write caching, maximizing I/O performance and effectiveness.

The same set of variables and considerations apply for host environments with multiple SAS sessions. However, the SAS application and dataset size is computed as the sum of all active sessions.

GENERAL NFS PERFORMANCE TUNING

NETWORK FILE SYSTEM - OVERVIEW

NFS can be a "high performance I/O infrastructure".
NFS was originally created as a method for sharing data on a local area network. As network technologies advance NFS becomes more capable. Specifically, many organizations now use Network Attached Storage (NAS) as the primary technology for connecting host computers to storage subsystems. Making the transition from a simple shared environment to a high performance I/O infrastructure requires NFS configuration modifications and tuning beyond default public network settings.

NFS Clients are not all created equal, nor are they configured the same way.
Each Unix operating system (e.g. Sun's Solaris, HP's HP-UX, IBM's AIX) has an NFS client. The purpose of this client is to translate file operations, such as read and write, into NFS requests over a network. These clients share one very important characteristic: adherence to a standard protocol definition. This means that each OS client will function correctly in conjunction with a standard NFS server platform. For example, NetApp filers provide a standard NFS server that can connect with all flavors of Unix NFS clients.

Network Appliance Filers are a key component of a high speed NFS I/O infrastructure.
In an NFS deployment there are two components: NFS Clients and NFS Servers. For the duration of this paper the only NFS server discussed is the Network Appliance Filer family of NFS Servers.

NETWORK TOPOLOGIES AND CONFIGURATION

Deploying NFS as a high speed storage infrastructure requires either a private network, a dedicated VLAN, or a point-to-point configuration.
There are multiple methods for deploying NFS attached storage: public networks, private networks, Virtual LAN (VLAN), and point-to-point.

A public network is not well suited for high-speed storage infrastructure demands. Other, non-performance critical, data traffic consumes bandwidth. Many infrastructures contain older networking components that are not high performance. A high-speed NFS deployment requires either a private network, a dedicated VLAN, or a point-to-point configuration.

NETWORK SPEEDS

Deploying NFS as a high speed storage infrastructure requires Gigabit Ethernet.
Current IP network technology has several speed alternatives. Common choices are 10 Mbit (mega-bit), 100 Mbit, 1 Gb (or 1000 Mbit). Many company's public networks (aka intranets) are currently deployed on 100 Mbit (or even 10 Mbit) technology. Deploying Gigabit networks requires upgrading to high speed NICs (Network Interface Cards) and Gigabit capable switching infrastructures. Gigabit deployment continues to become cheaper and easier as the required components become commodities.

NETWORK PROTOCOLS

UDP delivers more performance than TCP in high speed storage infrastructures.
Most Unix environments provide two protocol options for an NFS to IP transport: TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). In clean networks (e.g. point-to-point) UDP is a higher performance protocol than TCP. In general network infrastructures however TCP provides more predictable (less variable) performance due to the flow control mechanisms employed.. In high speed storage infrastructures the benefits of the consistent predictability of TCP must be weighed against the potential performance gains of UDP.

In addition to the TCP/UDP options, network deployments must also select which version of the NFS protocol to use, Version 2 or Version 3. NFS version 3 should be used when available.

MAXIMUM TRANSFER UNIT (MTU) – JUMBO FRAMES

NFS storage infrastructures gain increased bandwidth and increased efficiency with Jumbo frames.
IP networks can configure the Maximum Transfer Unit (MTU). This value specifies the maximum packet size for each transfer on the IP wire. The default value for this is 1500 bytes. Some network components (switches and NICs) support larger MTU size, specifically, 9000 bytes. An MTU size of 9000 is typically referred to as having "Jumbo frames".

Jumbo frames provide two advantages: more efficient use of the IP wire and fewer host interrupts per data unit transferred.

NFS CONFIGURATION

NFS clients require parameterization changes to achieve optimal performance in high-speed deployments.
Typical NFS deployments for applications other than high-speed I/O infrastructures require no changes to the default configuration for basic. However, the default settings are insufficient for high-speed deployments. Specifically, each NFS client has a default set of variable settings to control important concurrency and throughput settings. The most common variables control:

- Maximum threads: concurrent threads performing NFS operations on behalf of the user application
- Number of read aheads: concurrent, asynchronous read aheads performed on behalf of the user applications.
- Hiwater transmit and receive values for the transport protocol
- Maximum read and write transfer size (rsize, wsize)

The technique for setting these variables and the specific names vary among platforms.

OPERATING SYSTEM PARAMETERS

Autonegotiation - Ensure transfer speeds and flow control

settings are enabled and configured correctly.
Between every NFS Client and NFS Server there are three important network components: Host NIC, Switch, Server NIC. These three components (or in a point to point configuration, the two NICs) must agree on and equally support two important parameters: transfer speed and flowcontrol settings.

Kernel Patches - High performance NFS clients are still evolving. Aggressively follow OS vendor provided NFS client patches.
Each platform vendor periodically delivers patches for various OS related issues (bugs). A high-performance I/O infrastructure often requires the very latest patch levels.

CACHES AND LOCKS

A high performance SAS NFS deployment must carefully scope sharing, locking, and caching to maximize performance.
A primary feature of NFS is the ability to share data coherently across multiple host platforms. The NFS file server provides a centralized location for managing data sharing, locking, and coherency. Each individual host can have high-speed, cacheable, coherent access to the data. This key feature of NFS also provides a challenging environment for implementing a high-performance infrastructure.

NFS CLIENT CACHING

Caching with respect to the NFS client has several aspects:

- **Data read once can be cached in the host buffer cache.** Subsequent accesses to the same data can be satisfied from the host cache without fetching the data from the NFS server on each read. This property also enables prefetching: the host senses a sequential access pattern and asynchronously prefetches data on behalf of the application. When the application actually requests the data, the data is found in the host buffer cache – a performance benefit
- **Data written to the host buffer cache is first written to the NFS Server (cleaned).** Subsequent reads to that data can be satisfied from the host OS buffer cache. So data that is written and then read sees a performance benefit from the buffer cache.
- **Data set size plays a role in host buffer caching.** Data set sizes that are smaller than the available OS buffer cache (which is a subset of total host memory) can benefit from caching. Data sets that are larger than the OS buffer cache and have a non-predictable I/O pattern are difficult to cache. Although the host attempts to cache the data, the probability of accessing data in the cache decreases as a function of the ratio of data set size to buffer cache size. The end result is that most data is fetched from the NFS server on most accesses.

Locks
Applications accessing data via NFS can maintain data coherency with file and region locks. The application can choose to lock a file, guaranteeing that all accesses to the data from multiple hosts find the correct data. This technique is used by SAS to ensure multiple SAS sessions can access data files coherently.

Unfortunately, some NFS clients take a brute-force approach to maintaining coherency of locked data. Specifically, on some platforms, locking a file or data region results in all data associated with the file being invalidated from cache when the file is closed. This creates a performance degradation, especially when comparing performance with a native filesystem that maintains the data in cache for subsequent opens and reads.

Local Locking
As an antidote to the "locks == invalidate cache on close" issue, some platforms provide the concept of "local locking". NFS filesystems mounted with a "local locking" option enabled have two properties:

- **Locks are scoped only to the local host.** Applications sharing data on that host and using locks to provide coherency are safe. However, any other host accessing that data DOES NOT obey the locking/coherency semantics. This is acceptable for some deployments, and not others.
- **Host buffer caching IS enabled.** Since all locking and data activity (of interest) happens locally on the host, caching and coherency work the same as with a native filesystem. Specifically, data is maintained in cache across close/open.

Each SAS NFS deployment can maximize performance by understanding and applying the most appropriate locking and caching options.

Weak Cache Consistency

Beware poor implementations of NFS "Weak Cache Consistency" algorithms.
NFS Version 3 provides for implementation of a "Weak Cache Consistency" algorithm. This basically allows two or more clients to write-share a file while maintaining some level of consistency. Unfortunately, some implementations cause "false invalidations". Basically, even in a single host environment, a recently written set of data may be invalidated, even though no other hosts are accessing the data.

A scenario where this behavior decreases performance is quite common with SAS. Consider for example a common "data sort". The data sort has two phases:

- read the original data, perform a partial sort and write the partially sorted data to a "temporary file"
- read the temporary file, perform a merge, and write the result to the final data destination

An important factor in this operation is the relationship of temporary data size to OS buffer cache size. If the temporary data set is too large to fit in the OS buffer cache, then the false invalidates do not decrease performance. However, if the temporary data set fits in the OS buffer cache, then the invalidates are precisely the wrong behavior.

PERFORMANCE TUNING A SAS CONFIGURATION

This section discusses specific steps necessary to tune different SAS configurations for optimal I/O performance in an NFS environment.

Infrastructure configuration

- Deploy a gigabit Ethernet infrastructure.
- Enable Jumbo frames at the Ethernet level.
- Evaluate transport protocol (UDP or TCP).

Operating System Tuning

- Install latest kernel patches
- Disable auto-negotiation for Ethernet connections
- Increase maximum NFS threads, hi and low water marks, and streams settings

Network Appliance Filer Configuration

- For maximum performance in a single Filer environment:
    - create one (1) volume using all disks (except for

spares). This configuration assures maximum performance from the disks.
  - Store SAS Work in same volume as data files, but mount SAS Work through a separate mount point

Single Host Environments
- If multiple SAS sessions, all sessions share same SAS Work mount point
- SAS Work mount point is mounted with "local locking" option if available. In the absence of "local locking" option, enable SAS with the "filelocks=none" option.
- Mount all other data mount points with "local locking" if available. However, do NOT disable filelocks as a substitute for "local locking".

Multiple Host Environments
- If multiple SAS sessions on a host, all sessions share same SAS Work mount point
- SAS Work mount point is mounted with "local locking" option if available. In the absence of "local locking" option, enable SAS with the "filelocks=none" option. Then use the LIBNAME command to selectively enable locks for all non-SAS Work directories.
- Mount all other data mounts without "local locking" option.

SOLARIS CONFIGURATIONS

Operating System Version
Optimal performance is gained by using Solaris 2.9 or Solaris 2.8.

Gigabit Ethernet Configuration
On the Solaris Platform
- Sun provides Gigabit Ethernet cards in both PCI and SBUS configurations. The PCI cards deliver higher performance than the SBUS versions.
- Syskonnect is one third party NIC vendor that provides Gigabit Ethernet cards. The PCI versions have proven to be high performance NICs.

On the NetApp filer
- NetApp Filers provide Gigabit Ethernet NIC's as an optional connection technology.

Enabling Jumbo Frames
On the Solaris Platform
- Sun Gigabit Ethernet cards do NOT support jumbo frames.
- Syskonnect (third party NIC vendor) provides SK-98xx cards which do support jumbo frames. To enable jumbo frames execute the following steps:
  - Edit /kernel/drv/skge.conf
  - uncomment the line JumboFrames_Inst0="On";
  - Edit /etc/rcS.d/S50skge
  - add line: ifconfig skge0 mtu 9000
  - Reboot

On the NetApp filer
- Change MTU to 9000
  - Change the value with the command: ifconfig <interface> mtusize 9000
  - Make this permanent by adding to /etc/rc on the filer

NFS Protocol Configuration
On the Solaris Platform
- Edit the /etc/vfstab
- For each NFS mount participating in the high speed I/O infrastructure make sure the mount options specify UDP version 3 with transfer sizes of 32K:
  - vers=3,proto=udp, rsize=32768, wsize=32768,…

On the NetApp filer
- Ensure NFS v3 is enabled by entering the command
  - options nfs.v3.enable on

Kernel Patches
On the Solaris Platform
  - Access www.sun.com for latest patches

Auto Negotiation
On the Solaris Platform
Solaris GigE cards need to have auto-negotiation forced off and transmit flow control forced on. This can be done by
- Edit /etc/system and add the following lines
  - set ge:ge_adv_1000autoneg_cap=0 # force autonegotiation off
  - set ge:ge_adv_pauseTX=1 # force transmit flow control on

With Syskonnect provides third party Gigabit Ethernet cards for Solaris systems
- Edit /kernel/drv/skge.conf and verify the following lines exist
- AutoNegotiation_A_Inst0="Off";
- DuplexCapabilities_A_Inst0="Full";

On the NetApp filer
- Set the filer flow control setting is set to "full"
  - issue the command: ifconfig <interface> flowcontrol full

NFS Tuning
In Solaris, the most common method for setting NFS configuration variables so they remain persistent across system reboots is to edit the file /etc/system to include the following entries
- set nfs:nfs3_max_threads=64
- set nfs:nfs3_nra=64

Hiwater settings are typically added to the /etc/rc/init.d script by adding the following lines:
- ndd –set /dev/udp udp_recv_hiwat 65535
- ndd –set /dev/udp udp_xmit_hiwat 65535

Note that a system reboot is required for these variable changes to take affect.

Local Locking
On the Solaris Platform
- Edit the /etc/vfstab (follow with an unmount/re-mount)
- For each NFS mount qualified for "local locking" add the llock option
  - …,vers=3,proto=udp, llock, …

HP-UX CONFIGURATIONS

Operating System Version
Optimal performance is gained by using Version 11i and later.

Gigabit Ethernet Configuration
On the HP-UX Platform
- HP-UX supports Gigabit Ethernet NIC's

On the NetApp filer
- Filers provide Gigabit Ethernet as a connection technology.

Enabling Jumbo Frames

On the HP-UX Platform
- Use the HP-UX admin tool to enable jumbo frames

On the NetApp filer
- Change MTU to 9000
  - Issue the command: ifconfig <interface> mtusize 9000
  - Make this permanent by adding to /etc/rc on the filer

NFS Protocol Configuration

On the HP-UX Platform
- Edit the /etc/checklist (follow with an unmount/umount)
- For each NFS mount make sure the mount options specify UDP version 3 with transfer sizes of 32K:
  - …,vers=3,proto=udp, rsize=32768, wsize=32768,…

Kernel Patches

For the HP-UX 11i platform, the latest patches can be found at http://itrc.hp.com

Auto Neogiation

On the HP-UX Platform
- Verify in /etc/rc.config.d/hpgelanconf that HP_GELAN_AUTONEG is set to "1" for proper operation.

On the NetApp filer
- Issue the command: ifconfig <interface> flowcontrol full

NFS Tuning

In HP-UX, the most common NFS tuning variable is located in /etc/rc.config.d/nfsconf (reboot required after change):
- NUM_NFSIOD=32

Local Locking and Weak Cache Consistency
- HP-UX does invalidate cache when a file is closed IF the file was locked. Local locking may be supported later.

CONCLUSIONS

This paper shows how to deploy a SAS environment using NFS as a high performance I/O infrastructure with Network Appliance Filers. Included are recommendations for configuration and parameter changes to the SAS environment, the platform specific Unix operating system, NFS (Network File System) client, and the NetApp Filer subsystem. The result is a powerful, flexible, high performance SAS solution.

A full version of this paper, including additional tuning recommendations for other operating systems is located on the company web pages of both SAS and Network Appliance.

BIOGRAPHIES

Darrell Suggs is a performance engineer from Network Appliance. Margaret Crevar and Leigh Ihnen are from the SAS Corporate Technology Center (CTC).

Solaris	SAS v8.2	SAS v9.0
SAS Work Files	**If Solaris "Weak Cache Consistency" bug #(4407669) patch is available, then:** Place SAS Work on filer. Mount SAS Work with "llock" option. **If Solaris "WCC" patch is NOT available, then:** Use SAS "filelocks=none" option. Enable filelocks for all data other than SAS Work.	**If Solaris "Weak Cache Consistency" bug #(4407669) patch is available, then:** Place SAS Work on filer. Mount SAS Work with "llock" option. **If Solaris "WCC" patch is NOT available, then:** Place SAS work files on non-NFS filesystem. Watch for Solaris patch availability.

SAS Data Files	Place all Data Files on the filer an Enable Prefetch settings
	If data files are NOT write shared among multiple hosts, then
	Mount with "llock" option.
	If data files ARE write shared among multiple hosts, then
	Do NOT mount with "llock" option. Consider creating separate mount points for write shared and non-write shared data files.	

HP-UX	**SAS v8.2**	**SAS v9.0**
SAS Work Files	**If "llock" option is available, then:** Place SAS Work on filer. Mount SAS Work with "llock" option. **If "llock" option is NOT available, then:** Use SAS "filelocks=none" option. Enable filelocks for all data other than SAS Work.	**If "llock" option is available, then:** Place SAS Work on filer. Mount SAS Work with "llock" option. **If "llock" option is NOT available, then:** Place SAS work files on non-NFS filesystem. Watch for "llock" option availability on HP-UX.
SAS Data Files	Place all Data Files on the filer (according to above guidelines) **If "llock" option is available, then:** If data files are NOT write shared among multiple hosts, then mount with "llock" option. If data files ARE write shared among multiple hosts, then do NOT mount with "llock" option. Consider creating separate mount points for write shared and non-write shared data files. **If "llock" option is NOT available, then:** No action required. Watch for "llock" option availability on HP-UX.	