Best Practices for Configuring your IO Subsystem for SAS®9 Applications

Margaret A. Crevar, SAS Institute Inc.
Tony Brown, SAS Institute Inc.
Leigh A. Ihnen, SAS Institute Inc.

ABSTRACT

The increased power of SAS®9 applications allows information and knowledge creation from very large amounts of data. Analysis that used to consist of 10s-100s of gigabytes (GBs) of supporting data has rapidly grown into the 10s of terabytes (TBs). This data expansion has resulted in more and larger SAS data stores. Setting up file systems to support these large volumes of data, as well as ensuring adequate storage space for the SAS® temporary files can be very challenging. This paper will present some best practices for configuring the IO subsystem for your SAS9 applications, ensuring adequate capacity, bandwidth, and performance to keep your SAS9 users moving.

INTRODUCTION

Before we get into the best practices and guidelines, we wanted to state some basics up-front.

Starting with, this paper was written to help Windows, Linux and UNIX customers determine how to best set-up IO for their SAS applications. This paper is not applicable to z/OS users.

Key to the success of configuring the storage for your SAS application is to fully understand the workload and characteristics of your SAS application. The understanding can assist in the configuration of the storage. By understanding the SAS workload, you will be able to initially configure storage appropriately, improve current SAS application's performance, determine where potential performance issues may occur, plan upgrades to storage before performance issues occur, and insure a healthy IO throughput rate for your SAS applications.

Once you understand what the IO throughput rate for your SAS application needs to be, you need to work with your storage administrator to make sure that the storage behind the file systems you will be using can sustain the required IO throughput rate. We have found that lots of smaller disks work much better with SAS than a few large disks. In other words, the more IO spindles you have access to, the better.

Throughout the paper, if you come across a term in bold font, this indicates it is in the glossary (Appendix A) at the end of the paper. You can check there for a definition of the term. There is also a basic storage performance terminology description in Appendix B with general IO and storage descriptions that could be used by any application.

SAS IO CHARACTERISTICS

Let's start with describing how SAS does IO. SAS differs from a traditional **relational database** (while it is common to have only a single instance of the RDBMS running at any given time) in SAS, it is common to have many SAS instances running at any given time. Generally, each SAS user starts their own SAS session for each SAS job/application they are running. With the new SAS®9 BI Architecture, there are also several SAS servers that are started to support the JAVA applications, but the tendency is for each active SAS user to have their own back-end SAS server/process running.

Here are some of the characteristics of SAS software:

- SAS sessions started by SAS users tend to perform large sequential reads and writes. Some of the new SAS BI applications do some **random access** of data, but for the most part the SAS workload can be characterized as predominately **large sequential** IO requests with high volumes of data. One thing to note is if there are multiple concurrent SAS sessions accessing the same data file, (each SAS session is accessing the file in a sequential fashion), the access pattern for all the SAS sessions will appear random to the operating system and the IO subsystem needs to be tuned for random access rather than sequential access.
- SAS does not pre-allocate storage when SAS initializes or when performing writes to a file. When SAS creates a file it allocates a small amount of storage, but as the file grows during a SAS task, SAS extends the amount of storage needed.
  NOTE: file extension is limited to the amount of available space within the file system currently being used. SAS datasets and individual partitions within an SPDS table do not span file systems.
- Reading and Writing of data is done via the OS' file cache. SAS does not use direct IO by default. NOTE: Since SAS uses the OS's file cache to read and write data, the maximum IO throughput rate is restricted by how fast the OS's file cache can process the data. The maximum we have seen has been around 1GB/sec, but it generally is around 600MB/second per operating system instance.
- A large number of temporary files can be created during long running SAS jobs. These files are created, may be renamed towards the end of the task, deleted, and/or potentially manipulated many times within a long running SAS job. The size of the files may range from very small (under 100MB) to very large (in the 10s of GBs) on average.
- SAS creates standard OS files, for its data store (SAS datasets, indexes, etc.).
- When executing writes, there is a single writer thread per SAS session. Some SAS tasks that support threading do start multiple reader threads though.

STORAGE CONSIDERATIONS FOR SAS
Here are some general tips for setting up the file systems required by many SAS applications. Please note that these tips are very general in nature. *A specific SAS application or SAS Solution may require more file systems than are listed below.* Also, the exact configuration of the various file systems depends on the SAS usage and the underlying data model. But for the purpose of this paper, here are some general guidelines for setting up the file systems required by basic SAS applications.

It is generally recommended that a minimum of three file systems be setup to support SAS. Ideally, the SAS file systems would be their own independent set of physical disks. Use the below SAS file system characteristics and locations as a reference especially if the disk must be shared between SAS applications and/or users. The system administrator or installer should avoid sharing these heavily-used IO file systems with other applications (whether these applications are performing heavy IO or doing random access to the data) to avoid IO conflicts between SAS and these other applications.

SAS files and file systems:

- Root Operating System - location for the operating system and swap files
- SAS Executables - these could be placed on the operating system file systems
- SAS Data - location for the permanent SAS data files and raw input data, which can be SAS data marts. There will also be writes when the data mart is refreshed, and some occasional writes at the end of some SAS jobs.
- SAS WORK - temporary space for SAS sessions. There will be lots of sequential reads and writes of data in this file system. Depending on the SAS tasks, these

reads and writes could be lots of small files or several large files. The files written here are only available during the duration of a SAS session and are erased when the SAS session terminates normally. This file system can get the majority of the IO activity since this is where the temporary files are created during a SAS job. <u>NOTE:</u> some of the IO pressure associated with the creation of the utility files created by the threaded procedures can be alleviated by pointing the UTILLOC parameter (the file system where these files will be placed – the default file system is SAS WORK) to a different file system. More information on this subject can be found in the *"ETL Performance Tuning Tips"* paper starting on page 27. This paper is on the SAS Technical Support web site at:
support.sas.com/documentation/whitepaper/technical/ETLperformance07.pdf
<u>NOTE:</u> to avoid IO bottlenecks by using a single file system for SAS WORK, you can look into setting up multiple file systems and rotating SAS WORK between the different file systems for different SAS processes. This will balance the workload of the users across separate file systems (and if set up correctly, across separate physical disks). More information on this subject can be found in the above paper *"ETL Performance Tuning Tips"* starting on page 27 as well.

Recommended **RAID** configurations for each file system:

- <u>Operating System</u> – Mirror (RAID1) this ensures high availability of this critical file system
- <u>SAS Executables</u> – RAID1 these can be placed with the operating system file systems
- <u>SAS Data</u> - most SAS users want this to be a redundant file system to ensure the availability of the SAS data. RAID10 in general will give the best redundancy and performance (full mirror plus parity), but the mirror requires you to have twice as many disks. *Please make sure that you mirror the disks before you stripe them for reliability.* Most storage administrators employ RAID5 to avoid the cost of extra disks, but keeping the redundancy that parity offers.
- <u>SAS WORK</u> - In the past, SAS recommended the striping of the SAS WORK file system (RAID0) without redundancy/high availability for the best performance. This was rationalized because the files created in this file system are temporary in nature and can not be re-accessed if the file system or SAS session crashes. However, many customers now have the requirement of a highly available SAS WORK file system. Once again RAID10 generally will give the best redundancy and performance for a highly available storage solution. In addition, RAID5 (especially on the storage arrays that have large amounts of cache) is also very popular and can perform as well as RAID0 configurations.

Choosing the number of disks in a RAID5 array is a trade off between performance and the amount of time it takes to reconstruct a disk after a disk failure. SAS applications can be both bandwidth intensive for sequential activities (e.g. long reads and writes, sorts, etc.) and require a high number of **IOs per second (IOPs)** for random activities, (e.g. HOLAP operations, indexed reads, etc.). Typically a SAS LIBNAME maps to a directory on a single file system. If the IO throughput rates or IOPs requirement would necessitate using more disks than is prudent given recovery concerns, use striping to create a file system that spans multiple RAID5 volumes.

In most SAS configurations a single SAS WORK directory is shared by all users on a host system. This can create a need for a file system which has high IO bandwidth. Some storage arrays have a maximum number of disks that can be configured in a single RAID5 volume. Different striping strategies can be used to maximize IO bandwidth (either MBs/sec or IOPS) for a file system.

Details on how to setup the disks in your storage arrays is beyond the scope of this paper. We will give you some guidelines on the IO throughput rate recommendations and you can work with your storage administrator and/or storage vendor to determine how to accomplish this with the hardware and operating system you will be using with SAS. However, here are a couple of recommendations:

- When creating a standard operating system file system, you need to choose the best file system to support how your SAS application is going to run. If you are going to be doing lots of writes to the file system, then we do not recommend that you create an NFS file system to a file system on another computer. If you are going to be doing lots of sequential reads and writes, then we recommends the following operating system file systems – on Solaris 8 or 9 – VxFS, on Solaris 10 – ZFS; on AIX JFS2; on HP-UX JFS; on Windows NTFS.
- When setting up the file systems, please make sure that Read-Ahead and Write-Behinds/Write-Through (this term differs on various hardware platforms, but what we want is for the SAS application to be given a signal that the write has been committed to cache as opposed to disk) is enabled. <u>NOTE:</u> A word of warning with Microsoft Windows 2003 (32-bit) and processing large volumes of data. On some occasions, we are seeing issues with the Windows file cache when processing large volumes of data if the IO throughput rate is very high coming from the storage system. The Windows standby list appears to be getting over-run, stressing the ability of the OS to manage it and effectively keep up with refreshing pages. Windows sometimes responds by issuing an RC 1450 error message that is reflected in the SAS log as an "IO Error" message. If you are seeing this error message, please contact SAS Tech Support for guidance.
- When attaching a storage device via multiple fibre channels, you need to enable dynamic multi-pathing in your storage management software to spread the IO across the multiple connections to the storage device. (name and functionality differs per operating system)
- If you will be running SAS on a cluster of computers in a grid-like fashion, then you need to determine if the various nodes within the cluster/grid need to share file systems in a read/write manner. If so, you will need to look into a Clustered File System so that you have the maximum IO bandwidth between the shared file systems and each computer in the cluster. Again, when looking at these file systems you need to make sure the one you select can handle the IO bandwidth required of your SAS application.
- Hardware Tuning Guides can be found on the external SAS web site at www.sas.com/partners/directory/index.html. You will see a link to a web site for each of our Technology (hardware) Partners. From this web site there will be a link to various White papers and many of these contain hardware tuning recommendations.

Recommended IO throughput rates for each file system:

- <u>SAS Data</u> – for ETL processes that create SAS permanent data files, your application throughput requirements may dictate that you have a minimum IO throughput rate of 50-75MBs/sec for each SAS session that you are running. Reads and writes to the file system will occur during the ETL process.

  For end-user exploitation, SAS data files are typically only read from, and very little data is written back to the file. For most SAS users, a 15-25MBs/sec IO throughput rate per user may be sufficient. However, for some heavy SAS users you may need to plan on 50-75MBs/sec IO throughput rate per user.

- <u>SAS WORK</u> - There will be reads and writes occurring in this file system at all times. For most SAS processes, you will only need 15-25MBs/sec IO throughput. However, for some heavy SAS processes, like ETL, you may need to plan on 50-75MBs/sec IO throughput rate per user to achieve desired run-times.

The above are general guidelines regarding the IO needed for SAS. More specific guidelines regarding how to setup the file systems would require a deeper understanding of the specific SAS application and the data model that will be used. <u>The main concern when setting up a file system is to ensure that SAS gets the sustained IO bandwidth needed to complete the SAS jobs in the timeframe required by the SAS users.</u> It does not matter if you use internal drives, locally attached storage arrays, SAN arrays or NAS arrays as long as they can give you the sustained IO bandwidth needed by your SAS application/jobs.

KEY POINTS ABOUT SAS IO
SAS tasks involving large amounts of data can present different processing and access patterns, creating multiple challenges for system designers. However, some generalizations can be made about SAS data management:

- SAS usually does large, blocked IO, as opposed to a classical On-Line Transaction Processing (OLTP) environment. In general, SAS IO measurements typically focus more on MBs/sec rather than IOPS, however you must be mindful of IOPs if you are performing heavily indexed reads, returning numerous, small record requests, using HOLAP, etc.

  When feasible, it is a good idea to select a SAS dataset BUFSIZE to match the underlying IO architecture (e.g. stripe width). The degree to which sequential access is the predominant IO pattern determines the benefit of using a large BUFSIZE. If indexes are used to facilitate random access, unless the index has good **locality of reference**, then using a large BUFSIZE may decrease performance as larger amounts of data must be read per request. Typically, choosing BUFSIZE involves selecting a value which accesses a complete stripe width for the IO device. In particular, when RAID5 is the data availability strategy, matching the BUFSIZE with the **stripe size** minimizes the cost of calculating parity. Accessing data in stripe width amounts allows all devices in the stripe to be concurrently used, which typically generates the highest IO bandwidth from the device.

- <u>NOTE:</u> If your primary access pattern to the data will be random access which will return a small percentage of rows, then setting a BUFSIZE too large will cause you to do unnecessary IO since you will be only pulling a small percentage of information from the large buffer of data defined by the BUFSIZE parameter.

- SAS IO buffer sizes are fixed upon creation of the file. Therefore, optimizations to data movement must be external to the SAS process (e.g., OS kernel and file system parameters).

  If the SAS data set has been created, then changing the settings that control the preferred IO sizes for the file system may provide performance improvement. Typically this involves setting the degree of aggressiveness for the OS when doing sequential read-ahead/write-behind and caching of data for reuse on the application's behalf. However, total system throughput should be considered as well as individual job throughput. Aggressive settings that maximize individual performance may cause system-wide degradation when the system is under full load.

- SAS data sets and associated files are built within the confines of the underlying Operating System (OS), and can be managed by file management utilities which are a part of the OS (or may be a part of optional products). This also means that file placement can be determined by the definition of directory structures within the OS.

  For example, in the UNIX environment, if there is contention on a file system for multiple SAS data sets, because SAS data sets are OS objects, OS commands can be used to move the data set to a different device and create a symbolic link to a new location in the current directory. This is an example of after-the-fact performance tuning that is available. NOTE: On UNIX, after index creation, this method might work to separate index and data, which is a typical RDBMS tuning tip. SAS indexes are opened using the physical path where the SAS data set was found. As a result using concatenated libraries to separate indexes and data does not work.

- As previously stated, different types of SAS tasks have different IO patterns. Classical Decision Support Systems (DSS) and Data Warehousing (DW) may be highly sequential, while On-Line Analytical Processing (OLAP) environments may utilize multiple readers and writers (e.g. when accessing a sub-cube that does not exist) in a more random access pattern across multiple file systems. It is very important that you determine the SAS usage since this can determine the best way to setup the underlying IO storage system.

- SAS, as a rich discovery environment for BI and analytical solutions, creates many temporary files (primarily in the SAS WORK area) of unknown size. Both the creation of many small temporary files and very large sequentially accessed temporary files need to be supported by the storage subsystem and OS. In both cases, the size of the temporary file is not known at creation time.

- Because SAS IO does not use pre-allocated storage but is block-based in user specifiable sizes, the SAS application must create its own extents in a 'pay-as-you-go' design. As a result, contention for file system metadata (file system block to physical device volume mapping) can become a limiting factor for achievable IO bandwidth. A possible solution is to use a file system that allows for the configuration of large fixed extents. While this may be wasteful in terms of unused space, when dealing with large objects the percentage of waste should be small.

LOCAL vs CLUSTERED/SHARED FILE SYSTEMS
A local file system (including SAN storage) is the preferred practice from a performance perspective, but in today's ever growing SAS applications, there are times where you may consider splitting your SAS application across multiple smaller computers instead of purchasing a single larger computer (grid computing is an example of doing this) or you may want to separate different SAS applications or user communities onto their own computer (or operating system instance). In both cases, the SAS users or applications need to share a common SAS data mart. To do this, we recommend that you investigate using a clustered file system (sometimes called a shared file system) so that all the computers or instances of the operating system have the same read/write access to the SAS data mart as they would with a local file system. Some examples of clustered files systems are IBM's GPFS, Sun's QFS, and Veritas' SFCFS. It is also very important that the clustered file system you setup must give you the IO throughput required by your SAS application.

Please note that if you are looking to write back to a centralized file systems, we do not recommend that you use NFS for performance reasons since you tend to access NFS arrays via your LAN/WAN and there is an extra overhead for writes using NFS. NFS may be fine if you want to setup a centralized location for READ ONLY data. However, using NFS to a dedicated storage array that is directly attached to the computer should be fine.

PERFORMANCE CORRECTION EXAMPLES
Example 1, a SAS customer reported concerns to SAS Tech Support questioning why a SAS job was taking over 4 hours of Real Time compared to 30 minutes of CPU time per the SAS log. In looking into how the computer was configured, it was determined that the SAS WORK file system was pointing to a single disk drive in the storage array and since this file system was being used by all the SAS users on the computer, the SAS job accrued extra elapsed time as it waited for the single disk head on the single disk drive to become available. By moving the SAS WORK file system to a volume (of multiple disks drives) within an external storage array, the customer was able to drop the real time down to 30 minutes, which was the same as the CPU time for the job. The volume in the SAN had a higher IO throughout rate than the single disk drive, (the throughput of multiple disks striped together aggregated the throughput of the disks in the stripe, (for example if a single disk produced 20 megabytes per second, a stripe across 5 disks could produce nearer to 100 megabytes per second throughput) this is why the SAS job finished faster on the newer configuration.

Example 2, a SAS customer reported poor performance when accessing a very large file system on a local SAN. Most of the time the SAS users are happy with performance of their SAS jobs, but occasionally they are upset. The delta between Real Time and CPU Time increased dramatically during the periods of time when the customers were complaining. After monitoring the SAN, it was determined that the SAS WORK file system was sharing disks in SAN with other applications. The other applications were moved to a different set of disks with the SAN so that the SAN could sustain the required IO throughput all the time.

Example 3, a SAS customer was accessing a very large file system on a local SAN. Most of the time they are happy with performance, but occasionally they are upset. This is very similar to the previous example, but the computer becomes very sluggish during the times the customers complain when doing any application on the computer. In this case it was not an IO issue, but the fact that another application started running on the system (a back-up tool) and all the current applications were paused until the backup finished.

Example 4, a SAS customer determined they needed an IO throughput rate of at least 800MB/second to support 50 analytical modelers. A single RAID5 (4+1 disks) volume could not deliver the desired IO throughput. So working with their storage administrator, the customer used software to stripe across multiple RAID5 volumes and IO controllers to increase IO throughput potential for the computer.

CONCLUSION
It is strongly recommended that a detailed assessment regarding how SAS will function, the volumes of data that will be processes/analyzed/manipulated, and the concurrent number of SAS sessions running is done before you start to setup the IO subsystems. Use this assessment to determine the IO throughout rate. Remember SAS uses the operating system's file cache and this can become a limiting factor with the IO throughput rate that can be achieved (maximum of 1GB/second). You should always work very closely with your storage administrator and/or your hardware representative to ensure your IO subsystem can meet the IO throughput rates required by your detailed assessment.

In addition to this paper which gives general information on setting up IO subsystems, we are working with various storage vendors on additional white papers that discuss how to take these best practices and apply them to how you would setup their storage arrays.

CONTACT INFORMATION

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

Margaret Crevar
SAS Institute Inc.
100 SAS Campus Drive, R-2427
Cary, NC 27513
Work Phone: 919-531-7095
E-mail: Margaret.Crevar@sas.com

Tony Brown
SAS Institute Inc.
Millennium I
15455 North Dallas Parkway
Suite 1200, Room 1350
Dallas, TX 75001
Work Phone: 214-977-3919 x1655
E-mail: Tony.Brown@sas.com

Leigh Ihnen
SAS Institute Inc.
100 SAS Campus Drive, R-5206
Cary, NC 27513
Work Phone: 919-531-7631
E-mail: Leigh.Ihnen@sas.com

APPENDIX A: GLOSSARY:

IOs per Second (IOPS) the maximum number of IO (reads or writes) that can be done within a second based on the hardware and storage configuration. You need to make this value large if your SAS applications are going to be doing mostly random access to the underlying data store. This is not critical for most SAS applications which are sequential in nature.

Large Sequential – this is done by most SAS procedures that read the entire input file in order to do the task at hand. When this occurs, indexes are not used and a complete table scan, starting with the first row, is done.

Locality of Reference relates to data that is being stored in consecutive locations in memory. When the distance between this related data becomes so great that the data is no longer in physical memory, but in virtual memory that requires swapping of the data back into memory before it can be processed.

Megabytes per Second (MBs/sec) the maximum amount of data in megabytes that can be done within a second based on the hardware and storage configuration. You need to make this value large if your SAS applications are going to be doing mostly large sequential reads and writes.

RAID (Redundant Array of Independent Disks) is used to join multiple hard drives together in a way that can ensure you will not loose any data if you have a hard drive failure. There are different levels of RAID and the levels dictate performance, cost, and protection. There are two different types of implementation:

- hardware RAID, which is at the controller level
- software RAID, which may be implemented on the host system or within an intelligent storage device.

This means that you can potentially use software RAID to protect data, without using storage devices that support hardware RAID. In some cases, there is a performance penalty for using software RAID.

Random Access - this is done by SAS procedures that query a SAS table and return a small sample of the table. When this occurs, indexes are used and only the rows that fit the query criteria are read from the underlying SAS table.

Relation Database (RDBMS) – like Oracle, DB2, Teradata, SQL Server

Stripe size is the amount of data that is written to a device. Stripe width is the number of devices involved in the stripe set times the stripe size.

Striping is the foundation of RAID. Disk drives in a RAID group are partitioned into stripes, which may be as small as one kilobyte or as large as several megabytes. The stripe size is the amount of data that is written to one disk before moving on to the next disk in the array. Data is divided up by the RAID controller, according to the stripe size, and written across the drives. To maximize performance RAID arrays should be configured with stripe sizes that correspond to the average I/O request size and the number of drives in the array (stripe width). As the stripe size of an array is decreased, files are divided up into smaller pieces and distributed across more drives. As the stripe size of an array is increased, files are divided up into larger pieces and distributed across fewer drives. In a perfect world if all of your files were the same size, you would take the size of your I/O request and divide it by the number of drives in the array (less one for parity) to get your stripe size optimized for throughput.

APPENDIX B: BASIC STORAGE PERFORMANCE TERMINOLOGY:

These are the basic components, both hardware and software, that make up a typical storage system. The combined configuration of these hierarchical components can have a significant impact on the performance of the workload as it demands IO from the storage controller.

- **Storage performance:** The amount of work completed per unit of time, which refers to the number of IO requests (work) completed by the storage system per second. There are two measurements typically used to define the performance of a storage system:

    - **Throughput:** The volume of requests that can be processed by the storage system per unit of time. For storage performance, this is usually measured in megabytes per second or MB/sec. Throughput can be described as how effectively the storage controller is able to supply the application with data. Generally, batch jobs and scientific applications such as seismic evaluation are most affected by throughput.
    - **Response Time:** The number of IOs processed per second by the storage system, (number of IOPS) is another metric used to define the performance of a storage system. This describes how efficiently the storage controller processes transactions. Generally, applications that support on-line transaction are most concerned with IOPS.

- **Workload:** The specific IO sequence of reads and writes, cache hits and misses, and sequential or random activity that an application demands of a storage system.
- **Seek:** The initial operation a disk performs to place the read head on the right track of a disk drive.
- **Latency:** The secondary operation that occurs after the "seek", which is the time it takes for the data to reach the read/write head of a disk drive.
- **Transfer time:** The time it takes for data to be read from or written to the host after seek and latency.
- **Disk Drive Service Time:** The sum of seek + latency + transfer time.

IO Subsystem Definitions
When setting up an IO subsystem to store 100+GBs of data, a typical IO subsystem is not just a set of disk drives, but a hierarchical layering of software and hardware as depicted in Illustration 1 below.
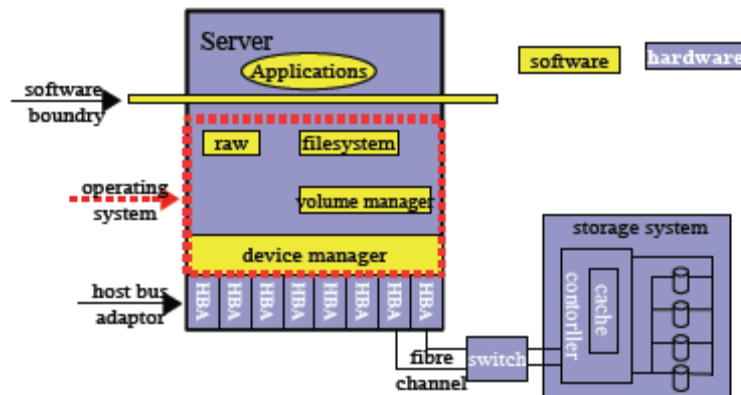

Illustration 1 Typical Storage System

The hierarchy starts at the server (hardware) level where the applications (software) reside. SAS accesses data which is stored via a file system that is created and managed by the server's Operating System (OS).

A **file system** is a physical partition on a disk drive or a group of disks, which is created by an OS command. It includes metadata which the OS uses to identify where to write files and other important file information -- specifically, inodes, blocks, and super blocks. An additional benefit of a file system is that it contains a file system cache that helps speed up access to data, as the cache is employed to access frequently used data, thus bypassing the latency associated with direct access to the physical disk drive.

Files systems can become very large, and the best way to manage them is to implement a **volume manager**. A volume manager is an application that manages disks by combining disk sectors into "pools" of storage space typically referred to as volumes. These volumes can then be subdivided and/or combined into RAID (Redundant Array of Inexpensive Disks) sets for redundancy and performance.
Additionally, the volume manager is typically utilized to help spread the data across many devices (either physical disks or storage arrays), which again, increases performance and redundancy.

The application must use an **interpreter** to access the storage system. This interpreter is called a **device driver**, which is a small program that controls a device and acts as a translator for programs that use a physical device. For a storage system, this device is called a Host Bus Adaptor (HBA). The HBA translates and transmits the IO request from the application to the storage controller via a connection, which today is most commonly a combination of fibre channel interconnect and storage switch. The fibre interconnects and storage switch are the conduit used to move the IO from the server to the storage. However, these are not discussed at length in this paper.

The **storage controller** resides inside the storage array and consists of a mixture of hardware (e.g. storage cache, cpu) and software, (RAID software, RAID algorithms). The controller is the interface into the actual physical disk drives. The controller contains a storage system cache, which is composed basically of memory chips that enable the storage system to access frequently used data directly from memory rather than from the slower mechanical disk drives.