# Tuning Guide for SAS®9 on AIX 5L

**Hsian-Fen Tsao**
**Brian Porter**
**Harry Seifert**
**Edward Hayes-Hall**

## Table of contents

# Abstract

*SAS® software is a complex Business Intelligence and data analytical application that allows clients to view their data in ways beyond what a relational database can offer. Exploitation of its capabilities is possible when the system is capacity is sufficient and is properly tuned for performance.*

*In this paper, we first provide a brief overview of SAS® 9 and typical performance scenarios. Then, we focus on general best practice suggestions and performance settings for tuning the AIX 5L™ Version5.3 operating system for enhanced SAS® 9 performance on POWER5™ processor-based servers. The tuning of the disk IO subsystem and SAS application will be outlined briefly. This paper will also discuss general performance monitoring methodology and performance tools. The tuning performance scenarios, from the monitoring, analysis and tuning perspectives, will be demonstrated by using real-world data collected within a simultaneous multithreading-enabled environment on an IBM POWER5™ 1.65 GHz p5-590 server running SAS® 9 and AIX 5L V5.3 operating system.*

# Introduction

The objective of this paper is to provide general suggestions for performance optimization from a system-wide perspective in order to create an enhanced environment for SAS® 9 on IBM System p™ POWER5 processor-based servers.

The target audience for this paper can be all levels of IT personnel (e.g. system administrator, performance analyst) who are interested in or required to do SAS setup on an IBM POWER5 processor-based server.

## Objective

## Overview of SAS® 9

SAS® 9, the current production version of the SAS System, has many new features – threaded procedures, new data store that enables parallel reads, the ability to split SAS WORK and utility files into multiple locations, a new multi-tiered structure to support the new SAS Enterprise Intelligence Platform SAS BI architecture – that changes the workload characteristics when compared to SAS 8.2. SAS 9.x is a 64-bit only implementation of the SAS system whereas 8.2 could be either 64-bit or 32-bit
However, much of the base SAS® 9 system functions the same as it did with SAS 8.2.

## Typical performance scenarios

Existing SAS 8.2 applications/programs will execute, without modification, with SAS® 9 if the SAS data sets have been converted to 9.x format. In the scenarios where the applications/programs contain any of the SAS® 9 threaded procedures (SORT, SUMMARY, GLM, DMREG, etc.), these applications/programs will take advantage of the new threaded SAS® 9 features without any modifications. Please note that with more threads, a process may be able to utilize more I/O bandwidth, so additional I/O bandwidth may be required to achieve increased application performance.

In general, SAS customers perform two types of activities with SAS. The first is creating a structured BI warehouse environment (via an ETL process, generally in a batch mode) and then end-user exploitation of the structured BI warehouse. The

second is ad-hoc BI environment where the user(s) create their own separate data marts from source as necessary, perform the analysis/reporting, and persist minimal data (i.e. there is no sharing of these data marts with other user(s)).

Let's discuss the typical steps one goes through is setting up the structured BI warehouse scenario:

1. Acquisition of source data for the ODS is typically sequential and for very granular data such as customer transactions, store inventory or store sales can be quit large, in 10's of Gigabyte range, exceeding the size of the file cache. These are now often stretching into the 100's GB and TB range as well.
2. Data profiling and hygiene operations on source data are typically implemented using sequential I/O access to the source files. For performance, the data in the source files may be sorted to facilitate cross source validation. For example, a record for an item sold at a store must have a matching store inventory record.
3. When loading data from the ODS into a dimensional model, the uses of small look-up tables and SAS Format catalogs is common insuring that the tables are in the file cache which can increase the warehouse load performance.
4. Query and Reporting against a dimensional model typically involves random access of both fact and dimensional tables. For the Enterprise scale solution, the likelihood of having significant cache fit rates on accessing fact tables randomly is very low. If the typical query patterns involve dimension or look-up tables which are small, and if the tables are cached, query extraction performance will be improved.
5. Extracting data from the warehouse for an analytic exploitation data mart typically involves the same random access patterns as query and reporting applications.
6. Building the exploitation analytic data mart typically involves sorting and sequentially accessing the extracted data from the warehouse and sequentially writing to the analytic data mart. The volumes of data may be small enough that file cache provides significant performance improvements depending upon how the analytic data mart is refreshed and the physical data model.
7. Analytic modeling of the data in the exploitation data mart and scoring of models is typically a sequential process.

The ad-hoc BI environment typically only does steps 1, 2, 6 and 7 from the above list against the data mart they create on the fly, not a 'formalized, shared, data warehouse' one.

# Performance Tuning Methodology

Performance tuning is a dynamic and time-consuming. Since there are potentially many parameters that can be changed, picking an optimal set needs experimentation. Improper system tuning can sometimes result in unexpected system behavior and performance degradation. Therefore changes should be applied only when a bottleneck has been identified.

## General best practices

It is always recommended to follow the best practices for environmental setup. The following is a list of general best practice recommendations for installing and running SAS® 9 on POWER5 processor-based servers with AIX 5L V5.3 operating system. The details of further tuning suggestions from a system-wide perspective can be found in subsequent sections of this paper.

- Use the latest version of AIX 5L Version5.3 with the latest maintenance level and e-fix.
  - As of this writing, ML3 with e-fix should be considered the latest maintenance level for SAS.

- Keep the Hardware Management Console (HMC) and microcode up to date.
  - Use the "Microcode Survey and Update Tools"[1] to keep the server and microcode up to date: http://techsupport.services.ibm.com/server/mdownload/mcodetools.html
  - To be notified on the daily basis of updates for AIX 5L, HMC and microcode: https://techsupport.services.ibm.com/server/pseries.subscriptionSvcs
  - Consult your IBM Customer Engineers regarding latest microcode availability.

- Use JFS2 on 64-bit AIX 5L kernel.
  - With the introduction of AIX 5L, IBM introduced a new file system referred to as Enhanced JFS (JFS2) that provides greater scalability than JFS. JFS2 is designed and optimized for a 64-bit kernel environment taking full advantage of 64-bit functionality. JFS2 is the default file system for a 64-bit kernel.
  - In general, SAS requires large rates of sequential disk I/O. The AIX 5L file system named JFS or JFS2 can detect and exploit the read-ahead and write-behind characteristics of the application under normal file caching policy.
  - You may choose "3. Enable 64-bit kernel" and "4. Create JFS2 File Systems" on "Install Option" screen during AIX 5L install, e.g., from CD.

- Enable the POWER5 simultaneous multithreading hardware.
  - Simultaneous multithreading designed into the IBM POWER5 processor is capable of exploiting instruction-level and thread-level parallelism. This hardware capability can be leveraged by the existing base of multi-threaded applications. Inherent in the IBM POWER5 processor's simultaneous multithreading implementation is the capability to dynamically adjust usage of hardware resources. These features help to achieve a higher system utilization and throughput.
  - Simultaneous multithreading is enabled on AIX 5L V5.3 as a default. Do not disable simultaneous multithreading.

- Use Gigabit Ethernet adapters instead of 10/100Mbps.
  - Boost network performance by using Gigabit Ethernet adapter(s). In addition to pure bandwidth, they support acceleration features such as hardware offload of TCP re-segmentation for transmitted TCP segments. Gigabit Ethernet also allows TCP to send large TCP segments to the adapter. These features reduce the software overhead, which lowers the CPU utilization, and releases CPU cycles to be used for other work by the server.

[1] For systems not connected to the Internet, a CD-ROM called **Microcode Update Files and Discovery Tool** is available. The levels of microcode that are on the CD-ROM are the latest levels available on the date that CD is created.

- It is not recommended that SASWORK or UTILLOC point to /var.
  - File system /var is used for various purposes, such as storing temporary files, mail spool files, and all security logging information. Run out of /var space may cause SAS process from being abnormally terminated.

- Increase the value of *maxuproc* to prevent SAS processes from being abnormally terminated or delayed.
  - *maxuproc* is a kernel configuration option that limits the number of processes that nonroot users are allowed to have simultaneously active. The process table is sized based on this. Increase the setting of *maxuproc* via smit or the command: *chdev –l sys0 –a maxuproc=<new value>.* It is recommended to start with the value 2,000.

- Increase user process resource limits for SAS users and database instance - 'unlimited' for all resources.
  - In /etc/security/limits file, '-1' is set for all resources.

- Configuring paging space at least with the following suggestions:
  - Place Paging spaces on dedicated disk(s) to eliminate I/O contention.
  - Use multiple paging spaces spread over multiple disks.
  - Make the primary paging space hd6 a little bigger than the secondary paging spaces.
  - Insure that the paging space is sufficient to support the number of concurrent SAS process as the number of SAS process can be dynamic depending upon the application workload.
  - Set nokilluid=1 with vmo

- Determining the application's I/O access patterns is important for I/O layout and tuning.
  - To achieve the best I/O performance, the access patterns and storage configuration should be compatible. If the application's I/O patterns are not known, then additional data may be gathered to determine dominant patterns. For example, in our experiments, AIX 5L trace indicates that the SAS Revenue Optimization application drives traditional large sequential I/O characteristics but it also contains a fair amount of random I/O. Thus optimization for different I/O access patterns (dominant and non-dominant) is recommended.

- Evaluate the use of Direct I/O (DIO) and release-behind method for "large" sequential I/O under CPU constraints due to high I/O wait.
  - File caching consumes more CPU and a significant portion of system memory. Data caching for large sequential I/Os might result in heavy page replacement activity. If cache hit rates are low, most read requests end up going to disk. The system may get saturated with many processes waiting on I/O. The caching of file data can be disabled using DIO or the file page can be discarded from the memory using the Release-Behind mount options. They both don't cache the data. The benefit DIO has over release-behind is that it eliminates the VMM layer so there is some CPU savings. But there is no read-ahead with DIO. Applications can compensate for the loss of this read-ahead by using methods such as issuing large enough reads.
  - Please see the details of "release-behind" and "DIO" in the next section "system-wide performance tuning".

- Ensure tuning from a system-wide perspective (e.g VMM, LVM, FS, disk storage) for SAS workload.

- Use the appropriate number of host bus adapters from the storage to the host server to provide the required front end application bandwidth.
    - Many SAS I/O workload patterns can be throughput intensive. However this is not always the case for all SAS applications or necessarily true during the entire SAS application's execution.
    - High performance storage channels should be considered such as Fibre Channel technology over slower mediums.
    - Use dynamic multipathing if possible to spread the I/O load over multiple adapters. Otherwise care needs to be exercise when locating SAS data libraries on mount points.

- Spread the I/O workload across many physical disk spindles rather than fewer larger capacity disks.
    - Provide better I/O performance by sizing for quantity of disks instead of capacity of disks.
    - Implement storage system RAID striping across multiple physical disks. Note: In general testing it has been observed that there is a slight performance advantage to using RAID10 over RAID5 for SAS temp space file systems. This is not necessarily the case for other SAS file systems. Use RAID10 or RAID5 depending on the level of redundancy and total capacity versus usable capacity that is required for each type of file system.
    - Use LVM stripping instead of concatenation.

- Minimize disk contention between SAS temporary space and data spaces.
    - Avoid disk contention by placing SAS temp space file systems and SAS data file systems on physically separate disks.
    - Use multiple storage server controllers to further separate and isolate the I/O traffic between SAS temp and data spaces. This also provides a more robust disk back end to handle I/O's.
    - Use multiple mount points for SAS file systems. Place system O/S, SAS, user, SAS temp, and SAS data file systems on separate physical disk.
    - If multiple users will share the SAS temp space (SASWORK) and increase disk or file system contention consider separating each user into separate SAS temp space file systems with physically separate disk.
    - Create separate JFS2 log files on separate physical disks for each SAS file system.

- Isolate SAS I/O from non-SAS workloads.
    - In general, SAS applications can be highly sequential large I/O workloads. Disk contention between SAS applications and other non-SAS small I/O random IOPS applications will increase service times of all applications and decrease I/O performance.

- Use the AIX 5L Scalable volume group or Big volume group with mklv –T 0 option to avoid the logical volume control block reserve of the first 4K of space.
    - With the LVCB present the first data block will start with a 4K offset.
    - When LVCB's exist on an lv, they can cause I/O's to span multiple physical volumes due to this offset.

- Be mindful that AIX 5L file systems are aligned on a 16K boundary when choosing the disk stripe or segment size or array stripe size.
    - A strip is the size of data to be written to each physical disk in the array. A stripe is the size of the full write across all the physical disks in the array. Example: strip size x number of disks = stripe size.
    - Note that the AIX 5L LVM stripe size that can be selected from the smit lv create panel is actually the single strip size (not stripe) or size of data to be written to each of the array disks and not the full stripe size across all the physical disks.

- Sync up SAS BUFSIZE with the storage system stripe size and the AIX 5L LVM stripe size (if using LVM striping), and VMM read-ahead increments.
    - Syncing of I/O sizes results in more efficient I/O's while reducing the total number of I/O requests to the storage subsystem.
    - Note: LVM striping may or may not provide better performance depending on the SAS application or the storage subsystem configuration. Testing your specific application is recommended.

## System-wide performance tuning

A SAS system does not operate in isolation. It will interact with the applications and the operating system. The performance will be impacted by each component of the entire system. e.g. the I/O layout, middleware configuration, etc. Therefore, performance monitoring and tuning should be considered from a system-wide perspective.

### AIX 5L VMM tuning

The AIX 5L virtual memory manager (VMM) is used to control memory resources. Virtual memory provides two major functions. First, it provides a security infrastructure for memory. Second, it allows the active amount of memory to be greater than the real memory in the system. This is accomplished by keeping the active portions of storage in real memory and spilling the less active portions to disk.

Virtual memory is managed on the basis of pages and segments. Pages are the granules of storage allocation. Groups of pages are managed within segments. Segments are divided into three types, based on the location of the storage that is used to back the page: persistent (JFS-backed), working (temporary or non-persistent storage), and client (JFS2, NFS, and others). Virtual-memory segments are classified as containing either computational (non-persistent) or file (persistent). The delineation of storage types between computational and file is used to balance the types of pages stolen by page-replacement algorithm.

The AIX 5L operating system allocates real memory to satisfy initial page faults for computational and file pages. Under default circumstances, file pages can be cached in real memory for file systems. The AIX 5L operating system uses memory mapping of files as the default method of file access. The contents of the files are cached in RAM through the VMM's use of real memory as a file buffer cache. This method is based on the fact that the access to memory is much faster than to the disk. However, under periods of heavy disk I/O, file caching consumes more CPU and a significant portion of system memory. The caching of file data can be disabled using Direct I/O or Concurrent I/O **mount** options; also, the Release-Behind mount options can be used to quickly discard file pages from memory after they have been copied

to the application's I/O buffers if the read-ahead and write-behind benefits of cached file systems are needed. The details of these tunings will be discussed in “AIX FS tuning”.

Due to SAS general workload characteristics, it is suggested to start with an AIX 5L file system like JFS2 to take advantage of its advanced file management techniques on AIX 5L (e.g. prefetching, caching, only logging metadata, etc) and the ease of the maintenance. Since AIX 5L file systems do almost all I/O through VMM, tuning VMM to enable file system to perform efficient caching is critical for SAS optimal performance. The parameters listed in table 1. can be tuned via the **vmo** command.

**Table 1: Most frequently used VMM tuning parameters**

| Parameters | Description |
|---|---|
| maxclient% | • Percentage of memory used to cache JFS2, NFS client, VxFS pages. The default is 80%.<br>• The number of these pages can exceed this value if *strict_maxclient* is 0.<br>• If the % of memory occupied by these pages exceeds *maxclient%*, when page replacement needs to occur, only these file pages are replaced.<br>• Set to a value that is less than *maxperm%*, particularly in the case where the value of *strict_maxperm* is set to 1. |
| lru_file_repage | • If the **lru_file_repage** parameter is set to 0 (1 by default), only file pages are stolen if the number of file pages in memory is greater than the value of the **minperm** parameter. |
| maxperm% | • Percentage of memory used to cache JFS pages. The default is 80%.<br>• Number of JFS pages can exceed this value if *strict_maxperm* is 0.<br>• If the % of memory occupied by JFS pages exceeds *maxperm%*, when page replacement needs to occur, only file pages are replaced. |
| minperm% | • The ratio of page frames (20% by default) used for files versus those used for computational segments is controlled by the **minperm** and **maxperm** values.<br>• **minperm** specifies the point below which file pages are protected from the repage algorithm. |
| lru_poll_interval | • The lru_poll_interval controls whether “lrud” should stop stealing pages in order to poll and process interrupts. The default behavior is not to stop processing.<br>• The value specifies the interval in milliseconds when lrud checks to see if there are disk iodone's that need to be processed.<br>• 0 is by default in AIX 5L V5.3C and 10 since AIX 5L V5.3D. |

| Parameters | Description |
|---|---|
| strict_maxclient | • If 1 (by default), then the limit for JFS2, NFS client, and VxFS pages is a hard limit and cannot exceed the value of *maxclient%*.<br>• If 0, then the number of these pages is a soft limit. |
| minfree | • The **minfree** limit (960 by default) specifies the free-list size below which page stealing to replenish the free list is to be started. |
| maxfree | • The **maxfree** limit (1088 by default) specifies the free-list size above which page stealing to replenish the free list is to be stopped. |

## AIX 5L FS tuning

The AIX 5L file system (FS) is called Journaled File System (JFS) or enhanced Journaled File System (JFS2). FS presents a logical view of files and directories linked together to form a hierarchical tree structure.

In general, SAS applications have a great deal of large sequential read and write disk I/O. If the workload has many large I/Os to a file system (e.g. large sequential I/O to JFS2), the I/Os may be bottlenecked at the file system level while waiting for a construct called *bufstructs*. The bufstructs for JFS2 is dynamic and the number of *bufstructs* per file system can be increased. The file system must be remounted for the new value to take effect.

The I/O characteristics of SAS usually create the situation where VMM read-ahead, and write-behind algorithm can be used to improve the performance of sequential file access. The parameters listed in table 2 can be tuned via the **ioo** command.

**Table 2: Most frequently used AIX 5L FS tuning parameters**

| Parameters | Description |
|---|---|
| j2_dynamicBufferPreallocation | • This tunable (16 by default) specifies the number of 16k chunks to preallocate when the filesystem is running low of bufstructs. |
| j2_nBufferPerPagerDevice | • This tunable (512 by default) specifies the number of bufstructs that start on the paging device. JFS2 will allocate more dynamically. It may be appropriate to change this value if *j2_dynamicBufferPreallocation* tuning has already been attempted and the number of external pager filesystem I/O requests blocked due to no fsbuf increases rapidly. |
| j2_maxPageReadAhead | • This tunable (128 by default) specifies the upper limit for AIX 5L JFS2 prefetching. It affects efficiently when doing large I/O. |
| j2_nPagesPerWriteBehindCluster | • Control the gathering IOs for sequential write behind. The default is 32. |

## Release-behind mechanism for JFS and Enhanced JFS

Release-behind is another suggested tuning technique for SAS. This feature allows the file system to release the file pages from file system buffer cache as soon as an application has read or written the file pages. This feature helps the performance

when an application performs a great deal of sequential reads or writes and most often, once accessed, these file pages will not be accessed again in the near future.

If release-behind is not used, it could cause threads to wait on page replacement to supply enough free frames to handle file reads or writes; in the worst cases, the page replacement activity may cause paging; When writing a large file without using release-behind, writes will go very fast whenever there are available pages on the free list. When the number of pages drops to *minfree*, VMM uses its Least Recently Used (LRU) algorithm to find candidate pages for eviction.

A trade-off of using the release-behind mechanism is that the application can experience an increase in CPU utilization for the same read or write throughput rate (as compared to not using release-behind). This is because of the work required to free pages, which is normally handled at a later time by the LRU daemon. Also note that all file page accesses result in disk I/O since file data is not cached by VMM. However, applications (especially long-running applications) with the release-behind mechanism applied will still perform more optimally and with more stability.

This feature can be configured on a file system basis. When using the **mount** command, enable release-behind by specifying one of the three flags below:

- release-behind sequential read flag *(-rbr*),
- release-behind sequential write flag *(-rbw*),
- release-behind sequential read and write flag *(-rbrw*).

### Direct I/O

In 2003, the AIX 5L Version5.2 operating system introduced direct I/O (DIO) for the JFS2. DIO is just like raw I/O except DIO is supported under a file system. They both bypass file system buffer cache, which reduces CPU overhead and makes more memory available to others (that is, to the database instance). DIO has similar performance benefit as raw I/O but is easier to maintain for the purposes of system administration. For applications that need to bypass the buffering of memory within the file system cache, direct I/O is provided as an option in JFS2 and JFS. For instance, some technical workloads never reuse data because of the sequential nature of their data access. This lack of data reuse results in a poor buffer cache hit rate, which means that these workloads are good candidates for DIO. For workload with large datasets that exceeds the size of the file cache is also a good candidate for DIO. By using the **mount** command with **-o dio** specified, all files in the file system use DIO by default.

For direct I/O to work, files must be accessed with the correct offset alignment and transfer size according to the file system used. Failure to meet these requirements will cause direct I/O to be demoted, and the data will end up going through VMM, which can cause a performance penalty. Although DIO eliminates the file system buffer cache overhead, the write-exclusive inode lock can still be a performance bottleneck. This inode lock can result in threads being blocked during context switches if the application is update-intensive.

**Note:** DIO can be restricted to a subset of files in a file system by placing the files that require these I/O techniques in a separate subdirectory and using **namefs** to mount this subdirectory over the file system. For example, if a file system **somefs** contains some files that prefer to use DIO, as well as others that do not, you can create a subdirectory, **subsomefs**, in which you place all the files that require DIO.

You can mount **somefs** without specifying **–o dio**, and then mount **subsomefs** as a **namefs** file system with the **–o dio** option using the command:

**mount –v namefs –o dio /somefs/subsomefs**
**/someotherfs/subsomefs**

## AIX 5L LVM tuning

The Logical Volume Manager (LVM) provides an abstract logical view of the underlying physical disk devices. Logical volumes are employed to contain paging spaces and dump areas, but mostly often they underlie file systems. LVM uses a construct called *pbuf* to control a pending disk I/O. A single pbuf is used for each I/O request. The application generating large amount of I/Os or striping and mirroring environment usually requires more pbufs to satisfy the system needs. Running out of pbufs can degrade the performance since the I/O initiating process is suspended until pbufs are available again.

- The parameter *pv_pbuf_count,* used to control the number of pbufs available to the LVM device driver, can be set for each logical volume via **lvmo** command.

## AIX 5L thread tuning

Within the libpthreads.a framework, the tuning knob “AIXTHREAD_SCOPE=S” has shown to improve the performance of the SAS application.

- AIXTHREAD_SCOPE={P|S} is an AIX environment variable. It is set to "P" for process-wide scope by default.
- AIXTHREAD_SCOPE=S changes the scheduling policy from process-based scheduling (m:n) to system based scheduling (1:1). In 1:1 model, each user thread is mapped one-to-one to an AIX 5L kernel thread, and each user thread runs on one virtual processor. In an SMP environment, system scope generally performs better than process scope. With process scope, the pthreads library manages internal thread scheduling using virtual processors which has higher overhead.

## Disk storage tuning

From a high level the AIX 5L I/O stack contains several layers that an I/O must traverse. At each layer, AIX 5L keeps track of the I/O. Some of the layers have specific queues that are useful to consider tuning. The I/O stack layers are:

Application
File system (optional)
LVM (optional)
Subsystem Device Driver-SDD or SDDPCM (if used)
hdisk device driver
adapter device driver
interconnect to the disk
Disk subsystem
Disk

In this section, the focus is on tuning the middle layers consisting of SDD, hdisk and adapter device drivers. The goal is to improve simultaneous I/O capability and realize efficient queue handling. See table 3 for some of the parameters that can affect disk, and adapter performance. In general, SAS applications will benefit from careful consideration and tuning of these parameters.

Both the disk and adapter have maximum transfer parameters that can be adjusted to handle larger I/O, reduce I/O splitting and coalesce I/O as it moves up and down the stack. In addition, both have I/O queues that can be adjusted to accept additional I/O's.

If SDD is used (IBM TotalStorage® DS6000 or DS8000) the data path optimizer (dpo) device I/O queue should be evaluated. SDD provides a vpath or virtual path to the storage subsystem LUN/logical disk and provides several hdisk devices through the physical paths (such as FC adapters). So, with SDD one can issue queue_depth x number of paths to LUN.

However, when the dpo device queue is enabled (default is yes), any excess I/O's that can not be serviced in the disk queues go into the single wait queue of the dpo device. The benefit of this is the dpo device provides fault tolerant error handling. This may be desirable for high availability applications, but for other applications there are advantages to disabling the dpo device queue and utilizing multiple hdisk wait queues for each SDD vpath device. Please note this is not an exhaustive discussion and does not detail any possible AIX 5L limitations for total number of I/Os. Also the queue parameters should be carefully evaluated before implementing any changes. For tuning guides specific to a particular IBM storage system such as the IBM DS4000, DS6000 or DS8000 see the additional information section.

**Table 3: Disk and adapter I/O tuning parameters**

| Parameters | Description |
|---|---|
| max_xfer_size | • FC adapter maximum I/O that will be issued. |
| max_transfer | • Disk maximum I/O that will be issued. |
| queue_depth | • Disk maximum number of simultaneous I/Os.<br>• The default is 20 but can be set as high as 256 for ESS, DS6000 and DS8000. |
| num_cmd_elems | • FC adapter maximum number of simultaneous I/O's. The default is 200 per adapter but can be set up to 2048. |
| qdepth_enable | • Subsystem Device Driver (SDD) data path optimizer (dpo) device queuing parameter.<br>• The default is yes. A setting of no disables SDD queuing.<br>• Use this with ESS, DS6000 and DS8000 storage. |
| lg_term_dma | • Long term DMA - Memory area the FC adapter uses to store I/O commands and data. |
| LTG | • AIX 5L volume group Logical Track Group parameter.<br>• LTG specifies the largest I/O the LVM will issue to the device driver.<br>• In AIX 5L V5.3, the LTG dynamically matches the disk maximum transfer parameter. |

Note: It is important to understand the I/O characteristics of the application in order to properly tune within the I/O stack layers. If the SAS application is predominantly large I/Os, then the application performance can benefit from adjusting maximum transfer sizes, long term DMA, and the LTG. The recommended starting values for a large I/O highly sequential workload are lg_term_dma=0x800000, and max_xfer_size=0x200000.

Queue information can be monitored in AIX 5L V5.3 and later with the iostat –D command. For AIX 5L V5.1 and V5.2, SAR can be used. It is recommended that qdepth_enable=no to use the hdisk wait queue rather than the dpo device wait queue.

It is recommended to increase the num_cmd_elems for the FC adapter from the default (can start at 400). Some of these parameters require a system reboot to take effect. For additional guidelines, see the tuning guide links found in this documents additional information section.

Use the following commands to display and modify disk and adapter parameters and settings.

**Disk – max_transfer, queue_depth**
- 'lquerypv –M hdisk#' displays maximum I/O size a disk supports.
- 'lsattr –El hdisk#' displays current disk values.
- 'lsattr -Rl max_transfer hdisk#' displays allowable values.
- 'chdev –l hdisk# -a max_transfer=value –P' modify current disk values
- Note: The device should be in an offline/disabled state before changing any parameters. Then cfgmgr will need to be issued.

**Adapter – max_xfer_size, lg_term_DMA, num_cmd_elems**
- 'lsattr –El fcs#' displays current value.
- 'chdev –l fcs# -a max_xfer_size=value -P' modify current value.
- Note: The device should be in an offline/disabled state before changing any parameters. Then cfgmgr will need to be issued.

**SDD/DPO – qdepth_enable**
- 'lsattr –El dpo' displays current value.
- Use datapath command to change if at SDD 1.6 or greater. Otherwise the chdev command can be used. Example: 'datapath set qdepth disable'

## Application tuning

You can often improve system performance by tuning your applications to make the best use of the system resources. Following are a couple of tuning tips.

- A new SAS® 9 feature will allow you to direct the utility files created by SAS to a different location from the SAS WORK area. The SAS parameter is called –UTILLOC. With this parameter and the -WORK parameter, you can direct temporary files created by a SAS session to different file systems (I/O paths). Both the –UTILLOC and the –WORK parameters must be set at the invocation of the SAS session. This feature can greatly help the performance of SAS applications/programs that are I/O bound on the directory where SAS WORK is currently pointing.

- Each SAS session consists of multiple processes. If the SAS session does not have an attached X-display then there will be two processes ***sas*** and ***elssrv***. If an X-display is attached a third process ***motifxsassm*** is started.

  The SAS Memory Utilization option MEMSIZE specifies the total amount of memory available to each ***sas*** process. As SAS dynamically allocates and frees memory the amount of memory used by a ***sas*** is usually less than MEMSIZE.

Svmon shows for a V9.1 sp 2 batch ***sas*** process after initialization 11,683 4K-pages of non system memory and 10,592 of the 4K-pages are shared. The ***elssrv*** process after initialization uses 11,093 4K-pages of which 10,592 are shared with the ***sas*** process. Hence the first SAS batch session at initialization uses ~49MB of memory and each subsequent SAS session at initialization uses 7MB.

For GUI based sessions the ***sas*** process after initialization uses 12,403 4K-pages of non-system memory and 10,592 of the 4K-pages are shared. The ***elssrv*** process after initialization uses 11,093 4K-pages of which 10,592 are shared with the ***sas*** process. The ***motifxsassm*** process uses 11,371 of which 10,592 are shared with the ***sas*** and ***elssrv*** processes. Hence the first SAS batch session at initialization uses ~55MB of memory and each subsequent SAS session at initialization uses 13MB.

# Performance Monitoring Methodology

## Goals

- Help to identify the performance bottleneck and tune the performance.
- Help to build an ISV workload characteristics profile in a POWER5 environment. It can be served as a baseline profile for future ISV and IBM collaborations. Potentially the data can be used to create an ISV sizing estimator.

## Monitoring Scope

Monitoring scope includes CPU and memory utilization, disk I/O, network I/O, chip and memory subsystem, application, logical partition/logical processor and system environment/configuration. It monitors each of those areas from overall to detailed perspectives.

## When to Monitor

- When the run reflects a *representative* time slice of application workload
- When facing a performance bottleneck
- Any time you prefer to see the details of insight

## Suggested performance tools for monitoring

| Tool | Description |
|---|---|
| **vmstat** | ▪ Monitor overall system performance in the areas like CPU, Virtual memory Manager (VMM) activity, and IO. |
| **tprof** | ▪ A global and micro-profiling tool. It is used to check the “hot spots”. |
| **curt** | ▪ Produce a detailed CPU utilization for process/thread/pthread activity. |
| **trace** | ▪ The trace can be post-processed to check the events like krlock contention, workload access pattern, inode contention, etc. |
| **svmon** | ▪ Monitor the detailed memory consumption on real and virtual memory. |
| **ps** | ▪ Monitor process/thread status and memory consumption as well. |
| **iostat** | ▪ Monitor overall IO stats including disks loads or adapters, and system throughput. |
| **sar** | ▪ Report the per-processor, disk, run queue statistics. |

| | |
|---|---|
| **filemon** | ▪ A magnifying glass tool. Used for detailed file I/O activity (e.g. hot lv, pv). |
| **netstat** | ▪ Report network and adapter statistics. |
| **netpmon** | ▪ Report detailed statistics on network I/O and network-related CPU usage, data rates and response time. |
| **hpmcount** | ▪ A tool that programs the on-chip and memory subsystem's Performance Monitor facilities to count a set of events. |
| **lparstat** | ▪ Report logical partition related information. e.g. partition configuration, Hypervisor call, and CPU utilization statistics. |
| **mpstat** | ▪ Report logical processor information in logical partition. e.g. simultaneous multithreading utilization, detailed interrupts, detailed memory affinity and migration statistics for AIX 5L threads, and dispatching statistics for logical processors. |
| **topas** | ▪ Report the local system's statistics, including: CPU, network, I/O, processes, and utilization of workload management classes. |
| **nmon** | ▪ A commonly used freeware tool for capturing AIX 5L performance data.<br>▪ Use this tool together with nmon analyser which loads the **nmon** output file and automatically creates dozens of graphs reflecting key system performance characteristics.<br>▪ See Appendix A. for procedure of collecting nmon trace. |

## Monitoring example - CPU utilization monitoring

- *Suggested monitoring tools*: **vmstat**, **iostat**, **ps**, **sar**, **tprof**
- *Overall CPU utilization monitor*

  A system is probably CPU-bound if the system CPU utilization (usr+sys) is always greater than 80 percent. **Iostat**, **vmstat**, and **sar** can help determine whether a system is CPU bound. Here **vmstat** is used to demonstrate the CPU monitoring methodology.

  The four CPU utilization group such as *us, sys, wa, idle* in **vmstat** report indicates CPU spent in user mode, system mode, idle or I/O wait. The first group "kernel thread" of two columns "r" and "b" represents statistics about thread queues. It is suggested to check these two columns first.

  *% us*: Percentage of CPU time spent in the application code (i.e. SAS). In order to maximize the throughput, ideally this value should be as high as possible.

  *% sys*: Percentage of CPU time spent in the system calls and kernel code. Ideally system time should be as low as possible. High % system time needs to be investigated.

  % wa: Percentage of CPU time spent waiting for an I/O (disk read/write, network etc.) to be completed. Ideally this value should be zero. If not, it means there is some opportunity to improve system throughput by either tuning disk or network or memory configuration.

  "r": Average number of runnable kernel threads during the sampling interval. The run queue is used to display the number of active tasks that are currently waiting for CPU resources. The higher the value in "r", the more CPU work there is to do, which is an indication of CPU bottleneck.

  "b": Average number of kernel threads in the wait queue during the sampling interval. If threads are consistently being forced to wait, CPU performance will get degraded.

- *Detailed CPU utilization analysis*
  If we decide that the system is CPU bound, then **tprof** can be used to check which process or program is dominating the CPU usage. **ps** can also be used but profiler is a better method. Once the culprit is identified, we can decide if this behavior is normal then tune as needed. Conducting further analysis before just adding more CPU processing power to the server is always recommended.
  - *Step 1- Profiling entire system*: In order to have a better understanding of SAS workload characteristics on POWER5, establishing a baseline profile is the first step. We get familiar with the workload pattern by checking if there are outstanding routines based on the profiling data. An outstanding routine means the CPU spends quite amount of the time in this routine comparing to others (e.g. 25% vs. 3%). Further evaluation is required for the outstanding routine.
  - *Step 2- Micro-profiling SAS user application*: Micro-profiling can focus on where CPU spent the most time in application. For instance, vmstat reports that CPU utilization mainly spent on user, micro-profiling of the application is reasonable.

# Example of Performance Monitoring and Tuning

- **Problem scenario:** SAS® 9 application runs on POWER5 processor-based server with AIX 5L V5.3 installed. The application environment including database is on JFS2. *pi*/*po* columns in **vmstat** output are constantly non-zero and the value of *wa* columns in **vmstat** output is high.
- **Diagnosis:** Often, an *iowait* time in excess of 20% (due to paging) can indicate a memory problem. If there is a high *pi* and *po,* it is likely that the system has memory constraints. In this case, the high I/O wait is due to paging. Excessive paging is because database file pages are causing database buffer cache pages to get paged out.
- **Tuning description:** Disable the use of repage counters and reduce *minperm%* to ensure that *numclient* is above *minperm*. The following tuning are applied**:**
  lru_file_repage=0
  minperm%=5
  strict_maxclient=0
  lru_poll_interval=5
- **Tuning results:** Paging condition was eliminated.

## Before the tuning

**Example 1:** vmstat output before tuning

```
System Configuration: lcpu=24 mem=92160MB

kthr    memory              page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy   cs       us sy id wa
10 82 7922525  3045   0  39 3472 40727  55082   0 2951 217026 54604   36 34  0 30
21 70 7922873  3057   0   9 2076 42936  53779   0 2858 219840 53694   38 34  0 28
47 43 7921948  2982   0  23  756 40141  80523   0 2636 224916 49550   39 34  0 27
46 44 7921431  2978   0  12  252 40710 114595   0 2622 224561 43536   38 33  0 29
66 24 7921400  3176   0   5   48 35612  62075   0 2127 190123 41881   37 29  0 35
```

## After the tuning

**Example 2:** vmstat output after tuning

```
System Configuration: lcpu=24 mem=92160MB
```

```
kthr    memory             page              faults          cpu
----- ----------- ------------------------ ------------     -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs     us sy id wa
92  0 1916538  3035   0   0   0 20489 87001   0 2726  85863 6550 92  8  0  0
94  1 1922958  3033   0   0   0 22013 27652   0 1407  89295 6365 92  8  0  0
94  0 1931803  3231   0   0   0 26657 27718   0 2904  81565 6428 92  8  0  0
93  0 1935122  7781   0   0   0 26463 27336   0 4141  79394 7425 92  8  0  0
93  0 1940973  2926   0   0   0 24519 25409   0 1812 101753 7186 91  9  0  0
```

# Summary

The recommendations covered in this paper have been proven effective on many systems under various SAS applications. They include using the AIX 5L 64-bit kernel with JFS2, configuring kernel options like *maxuproc*, tunning AIX 5L file memory, file system cache limit, page read-ahead, and release-behind feature to enable file system to perform efficient prefetching and caching for sequential reads and writes. They also include tuning disk and adapter parameters such as *max_xfer_size*, *lg_term_dma*, LTG, and syncing up SAS BUFSIZE. In addition to performance tuning, regularly monitoring environment and collecting data for analysis is recommended as well. These are the suggested techniques to enhance SAS performance and create a winning environment for SAS workload.

# Additional Information

- IBM Publications Center
  www.elink.ibmlink.ibm.com/public/applications/publications/cgibin/pbi.cgi?CTY=US

- AIX 5L V5.3 Performance Management Guide and
  AIX 5L V5.3 Performance Tools Guide and Reference
  http://publib.boulder.ibm.com/infocenter/pseries/index.jsp

- AIX 5L Performance Tools Handbook
  http://www.redbooks.ibm.com/pubs/pdfs/redbooks/sg246039.pdf

- Information about using IBM SSA Storage for SAS:
  http://www.sas.com/partners/directory/ibm/storagessa.pdf
  http://www.sas.com/partners/directory/ibm/storageess.pdf
  http://www.sas.com/partners/directory/ibm/sasstorage.pdf

- I/O subsystem Tuning Recommendations:
  DS4000 Best Practices and Performance Tuning Guide, SG24-6363-01
  IBM TotalStorage DS6000 Series: Performance Monitoring and Tuning, SG24-7145-00
  IBM TotalStorage DS8000 Series: Performance Monitoring and Tuning, SG24-7146-0
  Hints and Tips for Running SAS Software on an IBM eServer pSeries Server Running AIX 5L
  http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/TD102517

# Appendix A. Procedure for Collecting an nmon Trace

1. Point your browser to the following link to download NMON at the most current release: http://www-941.haw.ibm.com/collaboration/wiki/display/WikiPtype/nmon

2. Acknowledge the freeware disclaimers:

   **------>  International License Agreement for Non-Warranted Programs**

3. Download to PC client, and ftp this file in binary mode to target systems;
   nmon ftpserv ----> PC ----> AIX
   or, direct access from AIX browser ;
   nmon ftpserv ----> AIX

4. Uncompress (uncompress -v <filename.tar.Z) and extract using tar **(tar -xvf <filename.tar>.** nmon file contents into any directory, I suggest an nmon directory in sysadm $HOME workspace. Estimated file size for download time: **nmon9a.tar.Z is ~700KB. Extracted is ~5MB**
   To collect data, execute either nmon, or nmon64, depending on the version of the OS kernel running on target system a/o partition.
   To check the OS version and service levels, use following commands:
   - **oslevel –r** returns "5300-02" where 5300 is AIX Release, 02 is Maintenance Level
     or
   - **uname -v**

5. To check the OS kernel for 64bit vs. 32bit:
   - **bootinfo -K** returns "64" (64-bit kernel enabled) or "32" (32-bit kernel enabled)
6. The following command sequence will create a file in whatever directory **nmon** is executed from (ie; $HOME/nmon will create an output file, default fn is <hostid_date_time>.nmon, at 30 sec intervals, for 300 intervals, or precisely 2.5 hrs, including the top resource consuming processes during measured intervals. The -s, and -c flags should be adjusted to accommodate for workload timing profile for trace activity. If we are only monitoring a 15m time slice, maybe 5 or 10 sec intervals are more appropriate. Have to flex based on the circumstances. Produce and deliver results to IBM for trace analysis)
   - 32 bit AIX kernel : **nmon -f -s 30 -c 300 -t**
   - 64 bit AIX kernel : nmon64 -f -s 30 -c 300 –t

   NOTE: To use NMON in interactive mode, type nmon, or nmon64, using a uid with system level reporting authority, and an interactive console is presented, with on-screen help mode. Reporting is by toggle mode, hitting c reports CPU utilization, hitting it again turns it off. Hit m to report memory, d for disk, etc. Detailed help is also available in read.me in distribution pkg, and from the nmon interface by hitting the h key.

# References


IBM Corporation
Marketing Communications
Systems Group
Route 100
Somers, New York 10589

Produced in the United States

April 2006