Paper 334-2009

Running SAS® on Windows 32-Bit Systems: Ceilings, Limitations, and Realities

Tony Brown, Margaret Crevar, SAS Institute Inc., Cary, NC

ABSTRACT

Budgetary and IT support constraints are causing many SAS® shops to deploy large applications on the Microsoft Windows environment. SAS® 9.1.3 on x64 hardware is supported only as a 32-bit application in the Microsoft Windows environment, with 64-bit support coming with SAS® 9.2. This paper is designed to help you understand the performance pitfalls and limitations involving large SAS workloads in the Windows 32-bit environment. It will outline the common performance problems encountered, what can be done to correct them, and when graduation to a 64-bit operating system running a 64-bit version of SAS is required to circumvent limitations. Best practices for Microsoft Windows server setup and tuning, related storage subsystem arrangement, and future Windows 64-bit support information will be offered in the discussion.

INTRODUCTION

There are finite limitations for SAS workloads in the Windows 32-bit environment. The low-cost aspects of the environment are discussed, leading to why unmanageable workloads are being increasingly placed there. A rundown of the typical performance issues inherent with the Windows 32-bit environment are detailed. The effects of using the /3GB and /PAE switches to increase memory utilization by individual processes are explained, along with advice concerning their usage for SAS. Performance changes and benefits of the 64-bit version of SAS 9.2 are covered that address the 32-bit system issues. The paper concludes with general optimization and tuning advice.

THE APPEAL OF THE WINDOWS PLATFORM

The appeal of using platforms based on Windows for data warehousing, analytics, and reporting is obvious: very low hardware cost, a plethora of application software, commodity-level desktop software, and a relatively low cost of administration and maintenance compared to UNIX systems. The newer generation chip architectures offer very fast CPUs, which is the vanguard performance metric those who are purchasing tend to bias toward when buying new systems. In addition, all of the legacy SAS and BI products are available in the Windows 32-bit environment. All of these things add up to quick decision making to deploy to this platform when cost becomes an issue, as in today's market. This works well until workloads are introduced to these less-expensive platforms that quickly exceed the ability of the operating system to handle it, as we discuss in the case of deploying applications to the 32-bit Windows environment. This low-cost environment works well for small or medium workloads, but can be a hindrance when the workload scales. This experience catches many users by surprise, but one must be mindful that SAS is not a random-access database, and has very different I/O throughput characteristics because of the massive amounts of data it processes in large-block, sequential I/O.

TYPICAL 32-BIT PERFORMANCE PROBLEMS ENCOUNTERED

The performance problems encountered on the Windows 32-bit operating system are generally the same bottlenecks that can be found on their larger, 64-bit UNIX cousins, but at a much lower threshold in some instances. These lower thresholds primarily concern the rate of I/O overrunning the paging pool mechanisms, file cache overruns, and the amount of memory that the Windows operating system can offer each individual process in Windows 32-bit systems. This discussion focuses on these topics.

As with other server environments (for example, UNIX or LINUX), the majority of performance problems encountered in the Windows 32-bit environment are related to I/O bottlenecks. Storage is not configured properly to surface data to the server quickly enough to service application demand; or, the system file cache becomes stressed and begins thrashing or heavy swapping. This paper focuses on performance issues that are more related to the Windows 32-bit kernel and memory subsystem. See *Best Practices for Configuring your IO Subsystem for SAS®9 Applications* in the References section of this paper for a full treatment of I/O subsystem issues and how to correct and prevent them.

Windows 32-bit low-memory ceilings are no secret. On a per-process basis, you can expect the following limits:

- Approximately 1.7 GB per-process limit using 32-bit SAS on a Windows 32-bit kernel.
  Windows 32-bit kernels have a limit of 4 GB of address space per process, 2 GB of which are reserved for the Windows kernel. The other 2 GB are what each application process is allowed to run in. The SAS process is a little under that (1.7–1.8) because of application overhead considerations.

- 3.7 GB per-process limit using 32-bit SAS on a Windows 64-bit kernel. Only 2 GB of this can be contiguous address space unless multiple threads are used (more on that later).

A 64-bit version of SAS is now available in 9.2, which allows significant memory-ceiling relief on Windows 64-bit kernel systems. This is discussed later in the paper.

Low memory per process on a Windows 32-bit system is what is causing most of the pain in SAS applications. Applications have grown tremendously, and individual SAS jobs can overrun these low-memory ceilings, let alone the aggregated workload of an entire server. Leigh Ihnen of the SAS Enterprise Excellence Center explains what causes SAS on Windows to have these low ceilings:

> "The SAS underlying microcode structure is in the C language. The C language data types of longs and pointers determine how much memory can be addressed by the application and places size limits on objects created by the application."
>
> "The size of a pointer data type, for example, determines the ***maximum total memory*** that can be addressed by a single SAS process. A 32-bit pointer can address no more than 4 GB of memory (in effect, $2^{(32)}$=4 GB). The size of a long data type determines the maximum size of an individual data ***object***. This means, for example, that a statistical procedure cannot allocate a matrix larger than 2 GB, because that is the largest size that a signed long can represent. In summary, in the Windows 32-bit environment, the maximum total memory a SAS process can use is 2 GB (actually, this is closer to 1.7 once overhead is accounted for), and SAS can create only individual data objects of 2 GB in size. In a 32-bit operating system, SAS is a $2^{(32-1)}$-bit application."

SAS 9.2 for Windows x64 operating systems (Windows Server 2003 R2 Standard x64 Edition, Windows XP Professional x64 Edition, and eventually Windows Vista 64-bit Edition) uses a 32-bit long, 64-bit pointer model in its underlying microcode implementation. Hence, SAS processes are not limited to 4 GB as the 32-bit version was. This is discussed further in the **SAS 9.2 For Windows and 64-Bit Support** section.

In addition to the low per-process memory restrictions, the Windows 32-bit kernel tends to manage 4 GB for the entire machine (all processes) effectively, and can manage up to 8 GB fairly well. Above that, things get a bit dicey. The 32-bit Windows kernel tends to spend more resources managing paging and memory operations, yielding a rapidly diminishing marginal return.

When Windows runs out of RAM, swapping occurs, which can be slow given the architecture of the I/O subsystem. In addition, the operating system can readily suffer from file-cache overruns when the memory subsystem becomes stressed. SAS, by default, does **all** of its reading and writing to disk via the system file cache. It does not do direct-I/O or scatter-gather I/O (SGIO) in Windows by default. This is done on purpose so that SAS jobs can reuse pages in file cache without having to continually fetch from disk (among other memory-management reasons). When the page-replacement mechanism gets overrun with too many requests from fast I/O, errors occur that derail processing.

When the I/O subsystem delivers data faster than the Windows page replacement algorithm can accommodate it, the algorithm can become overrun and might give up without retries, returning the Windows error code RC1450. This code is reported as "ERROR: Undetermined I/O failure" in the SAS log. In SAS 9.2, this shows up in the Windows event log with the RC1450 code. I/O processing stops for that process. This has been observed at many customer sites running SAS, using fairly large files, and involving fast I/O subsystems delivering more data to the Windows file cache than Windows can manage, thus overrunning the Windows page replacement algorithm. This has affected SAS customers who deploy many simultaneous SAS sessions on a Windows server, using large files (for example,. files greater than 50 GB using SAS 32-bit on Windows 32-bit systems, and files greater than 100 GB using SAS 32-bit on Windows 64-bit systems). For sites that cannot upgrade to SAS 9.2 (64-bit) on a 64-bit Windows operating system, the only way to solve this problem is to go to multiple servers to spread the workload, slow down the I/O flow rate to the memory subsystem affecting the file cache, or use the SGIO switch in SAS, which has ramifications of its own.

Memory fragmentation can quickly become an issue with Windows and cause a need to reboot heavily stressed systems weekly (sometimes daily). In addition, administrators can unwittingly cause additional harm by giving applications additional memory for performance by using the /3GB and /PAE switches. These fixes often have unintended consequences for SAS.

GRABBING MORE MEMORY IN 32-BIT—/3GB, /PAE, AND AWE

There are at least two popular ways to increase the memory an application can use under the restrictive 32-bit Windows memory subsystem: using the /3GB switch or using the /PAE switch. The /3GB switch is used in the lower-level 32-bit kernel to allow applications to use more than 2 (1.7 for SAS) GB of virtual address space. The /PAE switch is a kernel-level switch that allows the kernel to be booted in PAE kernel mode (NTKRNLPA.EXE) to support greater than 4 GB of physical memory for most IA-32 platforms. We discuss both individually (they work differently), and their effect together on SAS operations.

THE /3GB SWITCH

This switch is set in the BOOT.INI file. It allows applications to use 3 GB of virtual address space for the application process, instead of the default 2 GB. However, this is done at a cost. When using the /3GB switch, the kernel space is reduced to 1 GB from 2, to allow the user-mode address space the extra 1GB. This effectively reduces the memory available for the pools of non-paged pool space, paged pool space, and system page table entries (PTEs). The effect can be that the memory reduction in these pools on a heavily used system can cause the server or application to generate an error or appear to hang up. The /USERVA=*XXXX* switch can be used to fine-tune, allowing a smaller amount to the user-mode address space, instead of the full 1 GB the /3GB switch allows. For more information, see http://support.microsoft.com/kb/316739.

The /3GB switch worked well for programs, such as database applications, and servers, such as Microsoft Exchange, which did not use a lot of kernel activity. It allowed less memory fragmentation, and relative data values were kept in a single large block for efficient processing. It borrowed from the kernel space to do so. The process memory utilization in question did not exceed 3 GB in most instances, and this allowed the /3GB switch to offer the performance improvement described. SAS, on the other hand, does use a lot of kernel activity on behalf of the user. **Therefore, SAS does not recommend the use of the /3GB switch**. It limits the memory available to the Windows kernel and can result in some device drivers not working properly or working as well. In addition, a reduction in file cache performance can occur. It can limit some system resources—because the size of the kernel is reduced, many system-wide resources can be affected. We have seen system-wide out-of-resource situations pop up with this. SAS already uses Address Windowing Extensions (AWE), and physical pages of memory allocated for use by AWE do not have a 2-GB constraint, even when the /3GB switch is being used. Only a small subset of the physical pages (governed by the SAS option MEMMAXSZ) is mapped into the address space at any given time. The size of the MEMMAXSZ option is important. It needs to be large enough to support all the memory-based libraries you intend to use. But, setting it excessively high can make less memory available for other processes, which degrades the system. In addition, no systems with over 16 GB of RAM installed can use the memory over 16 GB with the /3GB switch enabled. For more information, see http://support.microsoft.com/kb/283037.

THE /PAE SWITCH

The /PAE switch is a switch provided by Intel that enables the kernel to be booted in the PAE kernel mode (NTKRNLPA.EXE). This allows the kernel to provide virtual address space support greater than 4 GB per process for most IA-32 platforms. This is a 32-bit kernel option only—it is not supported or required in the 64-bit kernel. This option is invoked from the BOOT.INI file, and is not enabled by default unless you are using hot-add memory devices in your system. For more information, see http://www.microsoft.com/whdc/system/pnppwr/hotadd/hotaddmem.mspx or check with your administrator.

In general, it is not recommended to use the /PAE switch in concurrence with the /3GB switch if there is more than 4 GB of RAM in the total system. Doing so can cause system performance to suffer when a high demand for memory is placed and large allocations of system PTEs are required. When the /PAE switch is enabled, the operating system requires two PTEs to address a single page in memory, effectively halving the number of available PTEs. When coupled with the /3GB switch, the PAEs might be getting consumed at a higher rate. This might cause some operations to have performance issues, to even fail intermittently. The /PAE switch is handled by the memory manager independently of the programs that run on the system.

The /PAE switch allows the application to allocate the RAM directly with the help of AWE configuration, so when the application needs more than 2 GB of space, it moves to AWE memory blocks and performance remains best. **The PTE drawback remains, and on a heavily used system. it is contraindicated to enable PAE.**

In summary, for SAS usage, we do not recommend the use of the /3GB switch in 32-bit Windows environments at all, and caution judicious use of the /PAE switch to systems with memory subsystems that are not too heavily loaded.

WHEN THE WORKLOAD BECOMES AN OVERLOAD

The following are some general guidelines to help determine whether a 32-bit Windows system is overloaded to the point of having to add additional servers or having to migrate to a 64-bit system. First, here are some cases in which moving to a 64-bit version of SAS and Windows ***will not*** help your performance issues:

- **You are CPU bound in a single process, or on your system as a whole.**
  Be careful in determining CPU binding. You might have excess CPU capacity on your system, but the individual job you are running might be constrained in a single threaded process. If this single process is CPU bound, but the system is not, your choices of improving this job are to buy a faster CPU or parallelize the job operations if the job is amenable to that. If you are CPU bound on your system as a whole, you can add more CPUs if the system frame accepts them, and if you have adequate memory and I/O resources to sustain them.

- **You are I/O bound from the I/O storage subsystem.**
  This is a storage subsystem problem and has little or nothing to do with the server kernel. You can determine whether you have I/O storage subsystem issues if the SAS log FULLSTIMER statistics report that real-time versus user plus system CPU time is disparate (at least 25%), and the memory and file cache are performing adequately. On a PERFMON monitor, disk queues are chronically above .009, or Read or Write bytes per second are at your spindle capacity for the physical drive arrangement. See *Solving SAS Performance Problems: Employing Host Based Tools* in the **References** section.

Moving to a 64-bit version of SAS and a Windows 64-bit kernel ***will help*** your performance issues if your circumstance is as follows:

- **Jobs take much longer than they used to and you are neither CPU constrained nor I/O constrained from the storage subsystem.**
  If you are constrained from the I/O storage subsystem, moving from 32-bit to 64-bit might actually cause more I/O for the same task because you are now reading 64-bit record headers for each SAS observation.

And, the following is true:

- **You already have 8 GB of RAM or more in your Windows 32-bit system.**

And, the following is true:

- **Your Windows file cache is getting overrun.**
  In PERFMON, if your disk Read bytes per second chronically equal the I/O Read bytes per second, your cache page replacement algorithm might be working hard, thrashing the file cache. Highly random I/O accesses speed this degradation up. When the system-cache-resident bytes plus committed bytes come close to or equal RAM size, system file cache throughput can slow dramatically. High pages per second and cache faults per second are corollary figures to look at (chronically, over 100 pages per second indicates cache stress and page faulting.)

Or, the following is true, or also true:

- **You are experiencing heavy page faulting.**
  In PERFMON, if the pages per second is chronically over 100, you are likely stressing the system file cache and page faulting. Look at your page reads per second, which indicates hard page faults for corroboration.

Or, the following is true, or also true:

- **You are chronically exhausting free memory to the point of hurting performance.**
  In PERFMON, if the available megabytes are less than 4 MB, then the process working set is already constrained. If it goes to 0, you are memory bound. When the committed bytes per RAM attains a ration approaching 2:1, you are likely experiencing memory shortfalls. You should keep an eye on the situation when the percentage of committed bytes in use is chronically greater than 70% of the available RAM.

Or, the following is true, or also true:

- **Moving to two separate Windows servers and sharing the data via clustered file system is not a good option for you.**

Or, the following is true, or also true:

- **You can upgrade your SAS 9.1.3 applications and processes to SAS 9.2.**

More information about monitoring SAS processes can be found in the papers listed in the **References** section.

SAS 9.2 FOR WINDOWS AND 64-BIT SUPPORT

SAS 9.2 for Windows x64 operating systems (Windows Server 2003 R2 Standard x64 Edition, Windows XP Professional x64 Edition, and eventually Windows Vista 64-bit Edition) initially uses a 32-bit long, 64-bit pointer model in its underlying microcode implementation. Let's briefly discuss what this means.

By using 32-bit long data types and 64-bit pointer data types for the 64-bit operating systems, SAS no longer is constrained to the 4-GB process limit. This means that individual processes have a much higher memory ceiling (nominally restricted to the available memory in the server). This should alleviate many of the memory-related performance problems that can exist in the 32-bit-only environment. The 32-bit long implementation in the initial release of SAS 9.2 still limits the object size to 2 GB. (*This should be changed in the SAS 9.2 TS1M0 Release to a 64-bit long implementation, removing this limitation.)*

The initial release of SAS 9.2 (32-bit long, 64-bit pointer) included only the SAS Foundation products.

What do the extended memory ceilings mean for SAS? The following table shows the increase in kernel pool sizes when between SAS on a Windows 32-bit system and a Windows 64-bit system. In the long run, the increase in these sizes should significantly cut down the number of performance-related issues, such as overrunning the system file cache, swapping to disk, and running out of virtual address space for individual processes.

When comparing SAS using 32-bit longs and pointers on the Windows 32-bit operating system, to the initial release of SAS 9.2 using a 32-bit long, 64-bit pointer implementation on the Windows 64-bit operating system, the increase in kernel memory pool sizes and larger physical memories could reduce errors because of resource constraints, and could improve I/O performance by using more memory for file caching. See the table for increases in the maximum pool sizes.

Architectural Component	Windows Server 2003 for Itanium, EM64T, and AMD64 Platforms	32-bit Windows (x86)
Virtual memory	16 terabytes	4 GB
Paging file size	512 terabytes	16 terabytes
Hyperspace	8 GB	4 MB
Paged pool	128 GB	470 MB
Non-paged pool	128 GB	256 MB
System cache	1 terabyte	1 GB
System PTEs	128 GB	660 MB

Table 1. Maximum Pool Sizes

I/O Changes Moving to 64-Bit

In addition, Windows 2008 is anticipated to eliminate some of the file cache I/O issues that Windows 2003 exhibits. It is unclear whether the actual paging pool space overrun (for example, generating RC1450) is corrected. More information will be disclosed once we have an official port of SAS 9.2 for Windows 2008.

The move to 64-bit might have some unanticipated changes for the same data and data operations that existed in the 32-bit environment. Because each observation now has a 64-bit record header instead of a 32-bit record header (for example. they are larger), the data sets themselves can be 15% to 100% larger, depending on the makeup of the data set). This additional size generates additional I/O to read the data set. Comparing 32-bit data set I/O characteristics to the same data set stored in a 64-bit system shows the impact of the additional I/O and storage space required. Do not move to a 64-bit operating system if your only goal is to improve I/O. The I/O artifacts are inconsequential compared to the benefits of moving to a 64-bit system to gain the additional memory address space for SAS applications.

Improvements in Math Kernel Library Usage
SAS 9.2 uses the math libraries provided by the Intel compiler for optimal performance of SAS applications on Intel processor architectures. This yields a noticeable performance improvement on floating-point operations.

MISCELLANEOUS PRACTICE AND PROVISIONING TIPS FOR WINDOWS

SIZING

If you are deploying a new Windows server, and you are unsure of the capacity and throughput needed for deployment, please contact the SAS Enterprise Excellence Center for sizing guidance. Most servers tend to be under-provisioned from the initial setup, causing performance problems later.

MEMORY

The general recommendation from SAS for memory in a server is 4 GB per CPU. Given the restrictions of memory utilization on the Windows 32-bit kernel, that recommendation is mollified to 8 GB for the entire server. In 64-bit Windows kernels, the recommendation of 4 GB per CPU as a minimum holds. We do not recommend the use of the /3GB and /PAE switches in the Windows 32-bit environment. If you add physical memory to your system, and you cannot see it or address it via the operating system, check the BIOS to determine whether it is redundantly set up or mirrored. This causes the new memory size not to show up as you would expect on the kernel.

Swap space or Windows paging file space should be set at least 1.5 times the amount of total memory on the server. The paging file location should be directed to its own disk, or at least directed to a striped device that is not overly busy. Users already pay a performance penalty when paging occurs, and having the paging file on a slow I/O device exacerbates the issue. If you make your page file static on a defragged device, it helps keep the file contiguous, yielding better performance (for example. make the beginning and ending page file size the same on setup, and direct it to one location).

I/O THROUGHPUT

SAS typically recommends 100- to 125-MB-per-second throughput on the SAS WORK file system and on heavily used data file systems. Because NTFS file systems become fragmented easily, it is recommended to defragment them at least weekly to alleviate performance issues. Striped file systems across multiple disks are preferred to obtain the high bandwidth requested. For more information, see http://support.sas.com/rnd/papers/sgf07/sgf2007-iosubsystem.pdf. If you are using NAS storage via a network, understand that the packetized communication methods used by one of the most commonly used protocols—Common Internet File System (CIFS)—can make heavy I/O painful, especially on heavy Write requests to the NAS.

BACKGROUND PROCESSES

Be aware of background processes that are running on your server that might interfere with I/O on the SAS WORK file system. A good example is antivirus software or backup and restore software that is executing during user working hours. These processes can interfere with the large block, sequential I/O that SAS typically performs, resulting in noticeable performance degradations.

VIRTUAL ENVIRONMENTS

Virtual work environments are very popular on today's Windows servers. SAS works well within the virtualized environment. An acute awareness must be maintained from a holistic server perspective of the accumulated workload engendered on the individual virtual domains. Many performance issues are arising because of server resource bottlenecks in response to overloaded accumulated activity from the virtual domains.

SGIO

In the **Typical 32-Bit Performance Problems Encountered** section, SGIO is mentioned as a workaround for Windows kernels that are experiencing I/O errors because of the page replacement algorithm getting overrun. SGIO enables SAS to read directly from and write directly to storage, bypassing the system file cache. This circumvents the problem described. SGIO can be selectively turned off and on in a single job at the step level, or SAS can be invoked using an SGIO option. It is important to note that SGIO is not a silver bullet; it can have performance issues of its own and some SAS procedures cannot use it. If you are on a 32-bit environment, and you are experiencing frequent I/O errors in your SAS logs, please contact your Technical Support representative for expert help.

See the **References** section for additional papers on performance monitoring, optimal server and storage setup, and tuning.

CONCLUSION

By moving to SAS 9.2 on Windows 64-bit platforms, significant memory resource overhead can be taken advantage of, allowing your SAS processes better performance and room to run. In addition, floating-point operations are improved by exploiting the Intel math libraries. For Windows users who use heavy loads on their servers, these changes significantly reduce performance issues encountered on the limited Windows 32-bit platform.

By being aware of the tips in **Miscellaneous Practice and Provisioning Tips for Windows** section, SAS administrators and users can avoid many of the common performance pitfalls that are encountered in heavily used Windows environments.

RECOMMENDED READING

Crevar, M., T. Brown, and L. Ihnen. 2007. "Best Practices for Configuring your IO Subsystem for SAS®9 Applications." *Proceedings of the SAS Global Forum 2007 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/rnd/papers/sgf07/sgf2007-iosubsystem.pdf.

Crevar, M. 2007. "How to Maintain Happy SAS® Users." *Proceedings of the 2007 Northeast SAS Users Group*. Cary, NC: SAS Institute Inc. Available at http://www.nesug.org/proceedings/nesug07/as/as04.pdf.

Brown, T. 2007 "A Practical Approach To Solving Performance Problems with SAS®." Cary, NC: SAS Institute Inc. Available at http://support.sas.com/rnd/scalability/papers/solve_perf.pdf.

Brown, T. 2006. "Solving SAS Performance Problems: Employing Host Based Tools." *Proceedings of the Thirty-First SAS Users Group International Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/kb/17/759.html.

Brown, T. 2008. "SAS® Performance Monitoring: A Deeper Discussion" *Proceedings of the SAS Global Forum 2008 Conference*. Cary, NC: SAS Institute Inc. Available at http://www2.sas.com/proceedings/forum2008/387-2008.pdf.

RESOURCES

System requirements for SAS 9.2 are documented at http://support.sas.com/resources/sysreq/92/index.html.

A tutorial on the Windows NT Performance Monitor can be found at http://www.microsoft.com/technet/archive/mcis/*perfmon*.mspx.

ACKNOWLEDGMENTS

The author would like to acknowledge the assistance of Margaret Crevar and Leigh Ihnen of SAS for their collaboration, contribution to, and review of this paper.

CONTACT INFORMATION

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

Tony Brown
SAS Institute Inc.
Work Phone: 214-977-3916
E-mail: Tony.Brown@sas.com