

How to Increase I/O Throughput with SAS for Windows' SGIO Feature

Tracy Carver
Host Performance Evaluation

Introduction

SAS users tend to do very large amounts of sequential blocked I/O. When you have several of these type jobs running on a server, SAS tends to overrun the operating system file cache. When this happens on a Windows 2000/2003 server you may be able to take advantage of the SGIO system option to increase the jobs' I/O throughput. SGIO means "Scatter-read/Gather-write Input Output". This is an advanced option that uses a special feature of the Windows operating system. SGIO can tremendously improve performance in some situations. However, SGIO can also degrade performance, so its usage requires some study and experimentation. SGIO bypasses the Windows file cache – understanding this is critical in working with SGIO – so users should learn more about how the cache works.

About the File Cache

The operating system file cache is temporary storage in main memory used to hold data being written to disk and read from disk. Normal I/O always goes through the file cache. In most situations going through file cache offers the best I/O performance because of lazy writes and fast reads. With lazy writes, disk updates are lazily written at periodic intervals. The idea is that an application writing multiple times to the same portion of a file will cause several updates to the image of the file in memory, but perhaps only 1 physical update on disk. With fast reads, the application gets much faster performance because the data comes directly from memory, bypassing the physical disk. If the operating system finds a portion of a file in memory already, this is called a cache hit, and this allows fast reads to occur. The Windows cache manager automatically does read aheads when it detects a sequential read pattern, which can increase the number of cache hits. The opposite of a cache hit is a cache miss, when the operating system cannot find the requested piece of a file in memory, and it then has to read it from the physical disk. The file cache grows as files increase in size or more files are accessed. The file cache can shrink if files are deleted, or if applications request more memory – the OS will steal memory back from the cache to give to applications. To see the size of the operating system file cache, use the "System Cache" value in the Performance Tab of Windows Task Manager. You should also know that the cache can grow to close to the system's RAM on Windows 2000 Server and Windows 2003 Server and there is no way to govern this.

Though the file cache performs well in most situations, it can't solve every problem. The performance of the file cache is best if a file is small enough to fit in the file cache (implying it is also smaller than the system's RAM) and all the users on the system access this same small file. But the file cache can be overwhelmed by:

- Large files, in particular files larger than the RAM on the system. When the file cache approaches the size of RAM paging can sometimes occur.
- Cache contention can become a performance bottleneck, especially if there are multiple users accessing multiple files where the sum of the data files is larger than RAM.
- The overhead of moving data between the disk, the cache, and an application's memory buffers.

The way to resolve this is to have the Windows application bypass the file cache. This can significantly improve I/O throughput when files are too large to cache effectively.

SGIO and the Cache

SGIO provides a way to bypass the file cache when reading and writing SAS files. Sequential access is where SGIO has been seen to be the most useful. Random file access or multiple read passes through a file typically will degrade performance substantially with SGIO, so SAS restricts the usage.

To get an idea of how SGIO can affect your performance, think about the size of your SAS files at various steps, and whether they will fit into the system's file cache. Take for example a 256 MB data set. If you are on a server with 4 gigabytes of RAM, the operating system is easily capable of caching the entire file as long as there are no other demands. On the other hand, an 8 gigabyte data set clearly will overrun the cache and operations on that data set will place more demand on the system. The 8 GB file would be a better candidate for SGIO.

SAS OPTIONS to use with SGIO

When using SGIO, use BUFNO and BUFSIZE to get maximum performance gains. These affect the size of the I/O buffer that SAS passes to the operating system for I/O. In general, the larger the buffer, the better the performance, especially with large amounts of sequential I/O. The default BUFSIZE of 0 lets SAS choose a buffer size (for data sets this would be the data set page size) based on a multiple of the observation length. For most data sets, SAS for Windows will pick a page size of 4K, 8K, 12K, or 16K. Once the observation length exceeds 16K, SAS will increase the page size by 512 byte increments to allow the observation to fit in a page. This can be overridden using BUFSIZE.

With each SAS file created, SAS allocates BUFNO I/O buffers (default is BUFNO=1). With default buffered I/O, SAS usually uses only 1 buffer. However, with SGIO, SAS will use the extra buffers to read more data per I/O request. Suppose you are writing a data set with one numeric column, using BUFNO=16; then SAS will choose a 4K page size, and allocate 16 4K buffers for I/O. With default buffered I/O, SAS will write the data to disk 4K at a time. But with SGIO, SAS will first write 4K using the first buffer, and then subsequently use the remaining 15 buffers to write 60K of data (4K * 15) in one call to the OS. This repeats until the file is finished, with the majority of data written in 60K chunks. A similar process would happen when reading the file. Using BUFSIZE=16K and BUFNO=16, then first 16K of data would be written, followed by 240K of data (16K * 16). Remember, the second write is N-1 pages, where BUFNO=N.

Keep in mind that your mileage will vary depending on your hardware.

Performance Tunings to Try First Before Trying SGIO

- Verify there is no I/O bottleneck on the disks associated with the various file systems on the server.
- Look into splitting I/O loads onto different drives, concentrating first on drives that use different controllers.
- Think about the data set page size, which you can check with PROC CONTENTS. A small page size like 4K or 8K may perform poorly on some systems. To increase the page size you will have to recreate/copy the existing file with an increased BUFSIZE, either as a system option or a data set option. In addition, multiply the "Max Obs per Page" value by the observation length, and subtract the answer from the data set page size. This will tell you the amount of unused space in each data set page – you should pick a BUFSIZE that minimizes the unused space. In most cases, 64K for BUFSIZE is as high as you need to go with default buffered I/O.
- Use PROC DATASETS or some other means to delete unneeded data sets throughout your job, especially temporary ones created in WORK. Though WORK data sets will be deleted automatically when SAS ends, they will tend to take up file cache space in the meantime.

Examples

Below, we show some examples of the effects of SGIO and how I/O throughput improves as larger files are used.

Example 1 – SGIO on a 32-Bit Xeon Server

Table 1 shows the elapsed times in seconds for I/O intensive SAS Steps with and without SGIO for variously sized SAS data sets, on a Xeon 4-way server with 3.5 Gigabytes of RAM. Each test begins with a DATA step to write a large data set with 127 numeric variables and a data set page size of 16 K. In the step "DATA Read and Write", the data set was copied to a new one on a different disk drive. The "MEANS" step generates summary statistics on the variables. The "SORT" step sorts by the first two columns. The "SQL" step calculates the sum of each column. Each of these steps will have to read the entire data set at least once. The two large data sets created are then deleted, and each step is repeated for a larger file.

The SGIO results were run with a BUFNO of 128, which combined with the 16K page size means the I/O will be in approximately 2 MB-sized chunks. Remember that it may be useful to try different BUFNO and BUFSIZE values for each individual step.

Table 1. Real Time in Seconds for SAS Steps on a Xeon Windows Server With and Without SGIO.

SAS Step/Data Size	16 MB	256 MB	2048 MB	4096 MB	8192 MB	16384 MB
DATA Write	0.4	6.4	44.8	96.3	198.5	411.5
DATA Write SGIO	0.6	5.5	48.2	94.6	169.4	346.3
DATA Read and Write	0.1	6.0	74.6	235.0	460.1	884.3
DATA Read and Write SGIO	0.6	10.7	88.7	174.8	340.6	679.1
MEANS	0.2	1.5	31.8	133.9	204.0	512.7
MEANS SGIO	0.3	5.8	47.0	93.5	180.4	361.4
SORT	0.8	6.5	229.0	504.0	1,035.5	2,115.4
SORT SGIO	0.7	11.8	128.4	343.0	756.6	1,568.5
SQL	0.7	7.7	61.8	130.0	256.9	520.9
SQL SGIO	0.6	10.2	79.4	156.2	313.3	624.1
Total	1.8	21.7	397.2	1,002.9	1,956.5	4,033.3
Total SGIO	2.2	38.4	343.6	767.5	1,590.9	3,233.1

For the datasets that are 2048 MB through 16384 MB sizes the overall performance improves with SGIO, even though the SQL step is slower with SGIO. This should give you an idea of how SGIO can improve performance with larger files. (Note: The SQL case was measured as faster with SGIO on other systems).

The 16 MB and 256 MB sizes demonstrate the high performance of the file cache when files easily fit. It's useful to think in terms of I/O throughput (megabytes per second) instead of just runtimes. For the non-SGIO case, PROC MEANS took just 1.5 seconds to read the 256 MB file. (Note that with this test, MEANS only needs to make one pass through the data.) 256 MB / 1.5 seconds = 171 MB per second. However, with SGIO, PROC MEANS took 5.8 seconds to read the 256 MB file, which works out to 44 MB per second – this more closely reflects the true performance of the physical disk. The disk used was measured with an independent disk speed test that showed a maximum throughput of 47 MB/sec for sequential reads with a 16 MB buffer size.

If you contrast the 256 MB case with the 16384 MB case, you can see that the non-SGIO case of PROC MEANS took 512.7 seconds to read the 16384 MB file, which works out to a throughput of 32 MB/sec. However with SGIO, PROC MEANS ran with an I/O throughput of 45 MB/sec to process the 16384 MB file.

Example 2 – The Effects of Changing BUFNO

Table 2 shows the effects of different values of BUFNO with SGIO on the test program used in Example 1. BUFNO values of 1, 32, 128, and 256 were used. These are the total times of each test.

Table 2. Real Time in Seconds for I/O Intensive SAS Tests on a Xeon Windows Server.

SAS Mode	Default SAS	SGIO BUFNO=1	SGIO BUFNO=32	SGIO BUFNO=128	SGIO BUFNO=256
Real Time in Seconds	8,177.4	11,920.8	6,808.3	6,645.1	6,656.0

From examining Table 2, we can see that on the 32-bit system the fastest results were with BUFNO=128. The BUFNO of 1 means that each I/O, which goes directly to the physical disk, will have a size of 16K, because the data set page size is 16K. With the BUFNO of 32, the I/O will be in sizes of 512K, which will be much more efficient. Though the 128 value of BUFNO seems best in this case, remember that each step within your job should be evaluated for perhaps a smaller or larger value of BUFNO. The runtimes for the 32, 128, and 256 BUFNO results only differ from each other by several percent, so to minimize the resources consumed by SGIO, the BUFNO=32 value may be sufficient for most users in this example. Also, remember to think about explicitly using a different BUFSIZE in addition to setting BUFNO in order to achieve the best throughput.

Example 3 – Multi User Scenario

SGIO can have a benefit in a multi user scenario. Here, the idea is to look for SAS jobs that are doing sequential I/O on the largest files and try them with SGIO. This should reduce cache contention. The below scenario is constructed from 8 different I/O intensive tests launched simultaneously. Some of the tests are launched twice, so there are a total of 14 SAS jobs running concurrently. The biggest test, "DATA_SORT_INDEXCREATE", creates 4.4 GB of data sets and indexes. Since it is launched twice it will create a total of 8.8 GB of data files. By applying SGIO to both instances of this test, the cache contention is reduced considerably, increasing overall I/O throughput for the 14 concurrent jobs. To evaluate this, we look at the average runtime of the tests, and the time it takes for the longest test to complete.

Table 3. Multiple Concurrent SAS Tests on a Xeon Windows Server. SGIO was applied only to the "DATA_SORT_INDEXCREATE" tests.

Test	Default SAS	SGIO BUFNO=1	SGIO BUFNO=32	SGIO BUFNO=128	SGIO BUFNO=256
DATA_APPEND 1	960	727	1,151	1,132	795
DATA_APPEND 2	1,310	717	1,044	1,138	830
DATA_BASEPROCS	2,300	1,048	1,255	1,068	1,170
DATA_MODIFY_INDEX 1	692	512	657	663	657
DATA_MODIFY_INDEX 2	683	521	723	695	460
DATA_RANDOM 1	577	530	598	571	536
DATA_RANDOM 2	577	534	605	550	521
DATA_RANDOM_COMPRESSED 1	1,779	763	1,129	1,155	892
DATA_RANDOM_COMPRESSED 2	1,781	763	1,130	1,162	935
DATA_SORT_INDEXCREATE 1	2,923	3,543	2,399	2,372	2,413
DATA_SORT_INDEXCREATE 2	2,984	3,578	2,400	2,395	2,321
DATA_WHERE_INDEXED 1	386	358	400	378	458
DATA_WHERE_INDEXED 2	386	355	403	388	328
DATAMINING	1,230	1,168	1,256	1,294	1,176
Average	1,326	1,080	1,082	1,069	964
Completion	2,984	3,578	2,400	2,395	2,413

It can be seen that the average test time drops significantly with SGIO where the BUFNO is greater than 1. With these results, all SASWORK locations were on the same drive. Though the completion time doesn't change much with the different values of BUFNO, the average completion time is best with BUFNO=256.

Conclusion

- Availability: Windows 2000, Windows 2003, Windows XP, Windows NT SP3 on SAS 9 and SAS 8 data files. 32-bit SAS requires the page size to be a multiple of 4 K, and 64-bit SAS requires the page size to be a multiple of 8K.
- Do the size of DATA sets and temporary utility files for all concurrent SAS sessions exceed the size of RAM? SGIO is most effective for sequentially accessed SAS files that are significantly larger than the maximum size of the file cache.
- Start on simple jobs – do not use SGIO arbitrarily on entire SAS jobs, especially if they are complicated jobs with many steps.
- You don't have to use BUFNO as a global system option – you can set it using the OPTIONS statement within your job. Try to find a BUFNO that performs well at each step. Executing the SAS job with different BUFNO values will be required. We suggest you start with 32, 128, or 256. Avoid the default BUFNO value of 1.
- The combination of BUFNO and data set page size (or BUFSIZE) cannot exceed 64 MB, as that is Windows' limit for the size of an I/O buffer.
- Analytical procedures such as GLM, DMREG, NEURAL, and PHREG may not work well with SGIO; the reason is these procedures may need to make multiple passes through the various SAS data files associated with the procedures (either the permanent datasets and catalogs or temporary utility files).
- We have observed that SGIO may reduce I/O bottlenecks caused by disk fragmentation.
- When you find a significant decrease in real time with SGIO, you may also see a significant decrease in the system CPU time reported with FULLSTIMER.
- In a multiple user scenario, look for the user operating on the largest SAS files and think about whether SGIO might help.