DRAFT - Some Ad-Hoc Tuning Guide Lines

These guidelines have been developed following visits to a number of UK customers where various performance problems and have been reported and in many cases resolved following deeper investigation. During these visits a number of common patterns have emerged that SAS Administrators can, hopefully, exploit to improve and maintain performance in their environments.

This document is designed to help SAS Administrators tune their systems however, it is not a definitive reference source, for that consult www.sas.com and the various SAS conference proceedings referenced below. Also, see http://support.sas.com/kb/42/197.html for general documents around tuning.

Finally, as a SAS Administrator you are responsible for your SAS environments and the topics discussed should be carefully reviewed before being tested or used in a production environment as environments vary depending on specific configurations.

For assistance on tuning please in the first instance contact your normal SAS Support representative. If you have additional hints and tips that you feel SAS users could benefit from then please do let me know me on the email address below.

Andrew Gadsby, SAS UK&I Customer Loyalty Team

andrew.gadsby@sas.com

Contents

Getting Started

What Should Normal Look Like?

When looking at a system it is important to determine what the system should look like and what a normal SAS day should look like. This may look something like:

- The SAS batch should complete in X hours.
- Typical SAS queries should complete in N seconds/minutes/hours.
- Response and run times should be consistent between days/weeks/months.
- Looking at the hardware the IO is being used to support SAS jobs and not for other things.
- For the size of SAS data being manipulated the amount of IO appears to be about right.
- The CPUs are busy doing user work, and not spending a high percentage of their time doing "system" tasks.
- Number of users running concurrently is Y.

Once you have defined normal then it becomes easy to spot what is abnormal. These abnormalities then become the "tuner's" hunting ground and allow you to judge when you've hit the target and solved the problem. Typically, several of the areas above may be abnormal and so it is necessary to iteratively work through the issues, identify a number of solutions and test them in isolation and together. It is very rare for one solution to fix all of the issues.

The sections below cover a number of scenarios and are generally valid for Windows, LINUX and UNIX® type systems. Refer to the documentation for each system type to determine the exact options to apply.

Things to Collect

Prior to starting diagnosis you will need to obtain the following information:

- SAS logs for jobs affected ideally with FULLSTIMER turned on.
- PROC OPTIONS output.
- Operating System configuration details (e.g. LINUX sysctl)
- Access to system logs showing the underlying operating system view of things (sar in LINUX/UNIX and Windows Performance Monitor) these should cover both good and bad days.
- Simple Diagram showing how the various systems involved are connected including details of type of storage and network connectivity to the SAS system, amount of memory and number of CPUs.

Using this data allows you to build up a picture of what is going on with SAS and its impact on the underlying Operating System and physical hardware.

System Related Settings

Memory and MEMSIZE

Memory is now very cheap for new servers it makes sense to maximise the amount of memory available to SAS. This can be done by setting MEMSIZE to a value that allows your concurrent users to still fit into real physical memory (leaving space for other processes and the operating system. For example, consider a system with 128GB of memory an expectation of 10 concurrent SAS sessions then it may be worth looking at setting MEMSIZE to 8GB thus giving SAS about 80GB of memory and leaving 48GB for the disk buffer cache etc.. However, after increasing MEMSIZE make sure that the system does not start paging active processes or performance will drop dramatically.

Also, set MAXMEMQUERY to the same size as MEMSIZE so that SAS requests memory in single chunks rather than the default 256Mbyte allocation.

Buffer Sizes

The SAS parameters BUFNO and BUFSIZE can have a big impact on performance at the expense of additional memory. Basically, these parameters control how SAS undertakes IO to its files and in general bigger values tend to deliver better performance, providing physical memory is available.

Setting	Default	Suggestions
BUFNO	1	128 to 1024
BUFSIZE	0	512K to 1M

In simple terms SAS will allocate memory for each step based on:

BUFNO * BUFSIZE * Number of Files in Step

NOTE: this is also a data step option so fine tuning is possible.

Increasing these values allows input data to be pre-fetched thus keeping the CPU fed with data and for output to be undertaken as write-behind meaning that the CPU is not idle waiting for output to complete.

A further factor is that if random operations are being undertaken many more rows of data may be held in SAS memory and so applications may not need to do slow IO to physical disks.

A useful paper on these options for Windows is contained within https://support.sas.com/resources/papers/IOthruSGIO.pdf although similar principles apply for LINUX based system where SGIO is called DirectIO.

NOTE: On Windows turning SGIO on can have a significant negative impact as it completely bypasses the Windows buffer cache, see next the section.

Operating System Buffer Cache

The disk buffer cache in LINUX, UNIX and Windows operating systems can have a huge impact on SAS performance in both a positive and negative way. On the positive side the buffer cache can save SAS from doing a lot of real IO by holding pages of data in operating system memory saving read and writes to physical disk, along with the associated disk latency. However, if the working set of a SAS program does not fit in the cache then no benefit will be provided.

A major issue arises when the working set of data no longer fits in the buffer cache due to reasons of growth in: data set size; number of users or other memory usage factors, the result being the buffer cache no longer holds sufficient pages to support the SAS program running in memory. In this

situation jobs that may have run in minutes may suddenly take 10 to 100s times longer to run. The typical symptom of this problem is users start reporting that their jobs are suddenly slower and they claim to have not made any changes, this is a good indicator that all may not be well with the system buffer cache.

Solutions can include: increasing the amount of system memory, install a faster disk technology (e.g. FLASH) for common SAS data sets including SASwork; or tuning each program using the BUFNO / BUFSIZE options described above.

LINUX transparent_hugepages

Special thanks go to Niall Boys for investigating this option and providing detailed information on its impact.

A recent issues has appeared with some LINUX releases where the operating system kernel makes use of transparent_hugepages in an attempt to make memory access more efficient. However, for SAS systems where MEMSIZE is set to reasonably large values, e.g. > 500M, the operating system can spend vast amounts of system time trying to coalesce memory pages to create huge pages transparently. Until this capability is better understood it is recommend for kernels that support transparent_hugepages that they be turned OFF.

On one customer system setting this parameter to OFF reduced the run time of the overnight batch by some 30% and dramatically improved user response times.

Some useful information on this is available at http://oracle-base.com/articles/linux/configuring-huge-pages-for-oracle-on-linux-64.php search for “Disabling Transparent HugePages”. The SAS tuning guidelines at http://support.sas.com/documentation/cdl/en/hostunx/67464/HTML/default/viewer.htm#p1fnxsqfz4rzpmn1vop744ftxgf1.htm also make the same recommendations.

The key indicator of this problem on systems supporting transparent_hugepages is high system time and slow job progress.

PROC SQL Buffersize

If PROC SQL is used to undertake complex multi-way joins the SAS query optimiser considers several ways for undertaking the work. The exact mechanism used for a given query is influenced by data sizes and values, and so the strategy adopted can change over time. In many cases adding INDEXES to key tables is sufficient to enable good performance to be maintained. However, sometimes SAS will not use a given index and will adopt a different strategy, e.g. hash-joins, where in many cases BUFFERSIZE comes into play.

The query plan generated can be seen by using

```
proc sql _method _tree;
```

To understand the output and for many real examples see the excellent description within: http://www2.sas.com/proceedings/sugi30/101-30.pdf .

With PROC SQL the default BUFFERSIZE is set to 64,000 rows meaning that up to 64,000 rows of data may be held in SAS memory, if more rows are required SAS will need to write the data into temporary tables and scan the next set of rows and so on, until the join is completed. This tends to drive very high IO levels for slow job throughput.

BUFFERSIZE can be changed using:

```
proc sql buffersize=1000000;
```

Increasing, in this case, the number of rows held in memory to 1 million. Obviously, this has a cost in terms of memory allocated so you must ensure than MEMSIZE is sufficient AND that the system has enough real memory to hold the SAS process (or you'll swap SAS paging data in and out for the operating system paging data in and out).

Reasonable values for BUFFERSIZE can be determined by looking at the query plan and seeing which tables would benefit from being held in memory and then determine the number of rows in the table.

NOTE: Do NOT set BUFFERSIZE as a global, instead find the long running PROC SQLs from your SAS logs and tune each SQL to meet the needs of the expected data.

In one case increasing BUFFERSIZE to 10000000 reduced a job run time from 5 hours down to 7 minutes.

NOTE: the very act of changing BUFFERSIZE will probably change the query plan selected so do recheck the actual plan used.

Network Issues

Whilst not directly relevant to SAS where data is being pulled or pushed to external system the external network may be a significant factor in download/upload performance. The comments in this section are not directly related to SAS rather they cover general points you're your network team and system administrators will be aware.

LINUX Network Settings – work in progress as impact not fully quantified

On modern systems networks are often connected to SAS systems using 10Gbps network technology. In order to support consistent high throughput on these networks changes should be made to the default TCP/IP parameters, which were based on 100Mbps networks! Areas to consider include, from "/etc/sysctl –a"

Network Parameter	Normal Values	Suggested Values
net.core.wmem_max	262144	16777216
net.core.rmem_max	4194304	16777216
net.core.optmem_max	20480	65536
net.ipv4.tcp_wmem	4096 16384 4194304	8192 65536 16777216
net.ipv4.tcp_rmem	4096 262144 4194304	8192 262144 16777216
net.ipv4.udp_rmem_min	4096	16384
net.ipv4.udp_wmem_min	4096	16384

NOTE: These values must be carefully co-ordinated with your LINUX System Admin team who may be able to advice on better settings

Network Jumbo Frames

Most networks now support the concept of Jumbo Frames, these provide up to 9000 bytes per transfer compared with the standard 1500 bytes. If large file transfers are being made enabling Jumbo frames on the network can make a huge difference because fewer packets are required to move the same amount of data.

The use of Jumbo frames requires both network and system support so may not be easily available.

Storage

In recent years huge advances have taken place in storage subsystems and storage connectivity. Given that SAS programs generally do a lot of IO there is huge potential in making use of the newer technology to support SASwork and potentially even the core data files being used.

Exploiting FLASH or Solid State Disk Technology

This technology delivers extremely high bandwidth and low latency transfers which can be used by SAS to improve throughput rates. A joint benchmark from early 2014

http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=SA&subtype=WH&htmlfid=TSW03263USEN#loaded

showed SAS sustained throughput of 1.5GBytes/second, peak throughput of 2.8Gbytes/sec, from a single FLASH array and since then throughput has quadrupled, illustrating the speed of progress in this area. To put that throughput into perspective to achieve 6GBYtes of second of using traditional disks would require least 50 dedicated spindles.

An easy way to make use of FLASH is to use it for SASwork. Vendors are also offering test units so you can try before you buy and measure the real throughput and how it improves your application yourself.