ABSTRACT

The focus of this paper is on ways the user can optimize Release 6.06 of the SAS® System. Emphasis is placed on describing the following concepts: bundling and SAS system options that help optimization, memory usage, SAS data libraries under MVS, sorting considerations, and other methods of optimization.

INTRODUCTION

Usually optimization means balancing the tradeoff between speed and size. Where and how this balance is done is dependent on many things such as factors associated with the user's installation, SAS job size, contention, and so on. There are a number of techniques you can use to optimize the performance of your SAS programs under MVS. Some of these techniques, such as bundling, are site decisions. Some techniques are host-specific aspects that you should be aware of, while others, such as using the performance statistics reported in the SAS log to tune your SAS program, apply to any operating system. In any case, by using a combination of these techniques as appropriate for your particular programming situation, you can optimize the performance of the SAS System.

You should also note that the breadth of application solutions encompassed by the SAS System generates a numerically and I/O-intensive system. The level of numerical or I/O intensiveness is unevenly distributed throughout the system. Areas of the system that can be numerically intensive are General Linear Models (GLM), factor analysis, logistic regression, optimal experiment design, response surface analysis, and, to a lesser extent, the data reduction procedures (MEANS, SUMMARY, CORR, UNIVARIATE, and TABULATE). Other products that can be considered numerically intensive are SAS/IML® SAS/OR®, and parts of SAS/ETS®. The data manipulation portions of the SAS System that are I/O intensive consist of the DATA step and the SORT, PRINT, FREQ, FORMAT, CHART, and COPY procedures. Depending on the system usage mixture, performance can be improved by support of attached hardware and programming practices that make use of current machine architecture.

BUNDLING OF THE BASE SAS SYSTEM

Bundling eliminates dead space between modules and saves the overhead of loading each module. Only one copy of base SAS software is loaded if the Link Pack Area (LPA)/Extended Link Pack Area (ELPA) version is used. If the appropriate non-LPA or non-ELPA version is used, it is loaded in each SAS user address space and saves valuable ELPA/LPA space, but it can cause a large increase in working size, and can possibly place a heavy burden on the paging subsystem. The four bundle configurations for base SAS software are:

- MVS/370 and non-LPA
- MVS/370 and LPA
- MVS/XA and non-ELPA/LPA
- MVS/XA and ELPA/LPA.

The ENTRY parameter of the JCL cataloged procedure or TSO CLIST determines which configuration is used. Note that the code is identical among the four versions; the only difference is in the packaging. Refer to "Customizing the Base SAS Software Product" in Customization Instructions for Base SAS Software under MVS, Release 6.06, for more details.

USE OF SYSTEM OPTIONS IN THE BASE SAS SYSTEM

The main reason system options are used is to either streamline the SAS System or to restrict use of some of the options by the user. There are three types of SAS system options: session system options, which can be changed during the SAS job or session; configuration system options, which must be specified at invocation; and host system options, which are specific to the environment in which the SAS System is running. System options help optimize SAS data libraries under MVS, sorting procedures, and memory management. You can determine which SAS system options are in effect by using the OPTIONS procedure, which lists the three types. Some options can be set or overridden at different times. Refer to Chapter 12, "Host-Specific SAS System Options," in the SAS Companion for the MVS Environment, Version 6. First Edition for more details about the precedence for options set in several places.

SYSTEM OPTIONS TO HELP OPTIMIZE SAS PROGRAMS

The combination of the STATIST and FULLSTATS options causes CPU time, elapsed time, and EXCP count to be reported if the STIMER option is on, and causes the task memory and total memory to be reported if the MEMRPT option is on. Here is an example of a SAS log listing that shows performance statistics for a short SAS session:

NOTE: Copyright © 1989 by SAS Institute Inc., Cary, NC, USA.
NOTE: Licensed to SAS Institute Inc., Site XXXXXXXXXXXX.
NOTE: Running on IBM Model 3090 Serial Number XXXXXX.

NOTE: SAS system options specified are:
NEWS=W/MVSHOST.SASNEWS.W(NEWS) ALTLOG=NOTT
1 NOTE: The initialization phase used 0.30 CPU seconds and 2900K.
1 options fullstats;
1 run;
1
4 filename inn 'MISC.TEST(test1)' sh;
5 data x;
6 infile inn;
7 input;
8 run;

2 NOTE: The infile INN is:
1 Datname=0SEX10.MISC.TEST(test1),
1 Unit=3380,Volume=XXXI01,Disp=SHR,Bksize=3120,
1 Lrsize=64,Seek=40

NOTE: 16 records were read from the infile INN.
NOTE: The data set WORK.X has 16 observations and 0 variables.
NOTE: The DATA statement used the following resources:
1 CPU time = 00:00:00.09
1 Elapsed time = 00:00:00.65
1 EXCP count = 140
1 Task memory = 842K (34K data, 808K program)
1 Total memory = 357K (146K data, 211K program)
In the previous example, the performance statistics fall into three categories:

1. Initialization statistics
2. Task statistics
3. Session totals.

When you are optimizing the performance of your SAS programs, the task statistics should be used to provide the information you need to determine the performance effects caused by modifications to the SAS programs for your system. Those statistics include the following:

- CPU time, which is the time required for the CPU to perform the task.
- Elapsed time, which gives the wall clock time required for the task to be completed.
- EXCP count, which stands for execute channel program count.
- Task memory, which reports the memory used by the task.
- Total memory, which reports the total memory used by the SAS System after the task has been completed.

To interpret the performance statistics, you must remember that they relate to three critical computer resources: CPU time, I/O, and memory. Under almost all circumstances, reduction in the use of any of these three resources results in better throughput of a particular job and, in general, results in a reduction in the elapsed time. The statistics that you are probably most interested in are elapsed time and CPU time. Because elapsed time represents the actual wall-clock time, it is heavily dependent on the current load and the capacity of the system. So, as a general rule, it is not a good idea to use elapsed time as the only criterion for measuring the efficiency of your program. The CPU time statistic can tell you how much time was actually spent on your own job. Also, most data centers use CPU time to calculate data processing charges.

Another criterion for efficiency is EXCP count. The higher the number, the more system resources are being used. I/O optimization is the most important area to concentrate on when optimizing performance. In general, there is a tradeoff between conserving memory and enhancing CPU and I/O performance. With ample memory, you can use many large buffers that will usually reduce EXCP count and CPU time. This is where the use of bundling (described previously) SAS modules into the LPA (which allows more memory for buffering) can be of great value to many sites.

Also to help determine the systemwide SAS usage profile, the Institute has developed an SMF exit. The SMF exit records as user-written SMF records the system resource utilization of individual SAS steps in an MVS environment. The ability to record SMF records needs to be enabled at system installation time, and it is documented in the MVS installation guide. The SMF record type is user-definable, with a default value of 125. By matching the information reported in the SMF records to the list of vectorized procedures and other SAS options, the user can assess the potential improvement in SAS jobs at his or her site. User input is critical in analyzing the SMF records. Someone must review the results of the generated reports, which may represent a month of SAS runs, the anticipated result for the company as a whole. Turnaround can be critical only at a few specific times or with a few specific data sets, but perhaps not during the SMF-RM reporting-analysis phase chosen. Another issue to review is the opportunity cost savings of freeing machine cycles from the main processor. Utilizing vector processing can stretch the capacity of the main processor at an inexpensive cost compared to an upgrade. If the SMF-RM report indicates potential benefit with a vectorized version of the SAS System, then the SAS customer may want to enable the vector option of the SAS System.

**MEMORY CONSIDERATIONS**

SAS system options enable you to limit the amount of memory used by the SAS System or to decrease memory fragmentation so you can effectively manage memory resources used by the SAS System. The following SAS system options are used to constrain memory requirements placed on the host by the SAS System. The MEMSIZE= option specifies a limit on the total amount of memory the SAS System can use. Whenever acquiring new memory resources causes the SAS System to exceed the value of the MEMSIZE= option, the system tries to free up resources to satisfy the request. In severe memory restraints, the MINSTG system option can be used. This option releases all possible memory to MVS at the end of each DATA step or PROC step. This action keeps total memory at a minimum, but significantly increases CPU time.

To decrease memory fragmentation, the SAS System is able to obtain large blocks of memory from the operating system, out of which smaller memory requests are satisfied. This scheme, called superblocking, not only reduces fragmentation but also reduces the number of system GETMAIN calls that are issued. Superblocks can be composed of different types of memory, such as:

- I/O memory, which is located below the 16-megabyte line.
- Non-I/O memory, which is located above the 16-megabyte line.
- Permanent memory, which can be used for the duration of the SAS session.
- Temporary memory, which is reusable for another SAS task.
- SAS System portable supervisor memory, which is used by the portable supervisor.
- SAS System memory management control memory, which is used for control blocks to manage SAS memory.

Memory is acquired from MVS and is maintained in system control areas. The acquisition of memory is made by primary allocation, called initial storage allocation (ISA) and secondary allocation, called overflow storage allocation (OSA). The following SAS system options can be used to specify the size of the superblocks:

- PSUPISA/OSA=, which specifies the size of the ISA/OSA for the portable supervisor.
- VMPSISA/OSA=, which specifies the size of the ISA/OSA for permanent memory above the 16-megabyte line.
- VMPSISA/OSA=, which specifies the size of the ISA/OSA for permanent memory below the 16-megabyte line.
- VMPSISA/OSA=, which specifies the size of the ISA/OSA for temporary memory above the 16-megabyte line.
- VMPSISA/OSA=, which specifies the size of the ISA/OSA for temporary memory below the 16-megabyte line.
- VMPSISA/OSA=, which specifies the size of the ISA for SAS System memory management control blocks.

Each default configuration file supplied with the SAS System includes the defaults recommended for the operating system and...
method of processing. These default values can be changed to determine settings that are most efficient for your site.

You should take the following factors into consideration when using SAS system options to control superblocking:

- An ISA is acquired for each system control area when the SAS System is initialized.
- An OSA is acquired only when the ISA and previous OSAs cannot satisfy the current request.
- An ISA that is too large can tie up memory unnecessarily.
- Multiple OSAs can increase memory fragmentation.
- Temporary memory is freed at the end of the task.
- The SAS host supervisor frees all unused OSA units reclaiming that memory when memory is in short supply.
- ISAs are not freed, even in a memory shortage situation.

Refer to Chapter 17, “Optimizing Performance,” in the SAS Companion for the MVS Environment, for more details.

SAS SYSTEMS OPTIONS THAT CONTROL SORTING

There are new SAS options that enable the user to take advantage of more recent host sort utility programs and other host features such as MVSO/A and MVSO/ESA's 31-bit mode. The SORT procedure can access either the SAS sort utility, which can be provided as part of the SAS System, or a host sort utility, which can be provided by either IBM or another software company. In some cases, a host sort utility may be more appropriate for an application with a large number of observations than the SAS sort utility; therefore, the SAS System enables you to control how sort options that are used to select a sort utility and identify options can be used by that utility. For example, the SAS system option SORTPGM = specifies which sort utility is to be invoked. SORTPGM=SAS specifies that the SAS sort utility is to be invoked. SORTPGM=HOST specifies that the host sort utility is to be invoked. SORTPGM=BEST specifies that the SAS System is to choose either the SAS sort utility or the host sort utility, depending on the data to be sorted. The switch from the internal SAS sort utility to a host sort utility is made automatically at 500 observations.

Also the SORT31PL option specifies that a 31-bit extended parameter list be used to invoke the host sort utility; otherwise, a 24-bit parameter list is used. On MVSO/370 systems this makes little difference; however, using a 31-bit parameter list on MVSO/A or MVSO/ESA systems can boost performance, principally depending on whether the SORTEOOP option (specifying if the host sort utility supports the EQUALS option) is set. Most sites probably have more recent host sort utility programs available that can take advantage of some of the new options such as SORTOPTS, SORTEOOP, SORT31PL, and SORTSUMF. Refer to Chapter 17 in the SAS Companion for the MVS Environment for more details.

SAS DATA LIBRARIES UNDER MVS

The SAS System provides four options (BLKSIZE=, BUFNO=, BUFSIZE=, and COMPRESS=) that can help you to optimize performance and make efficient use of disk space for SAS data libraries under MVS. Each option can be specified as a SAS system option or as a data set option. In many cases a combination of more than one option is necessary to achieve the optimal performance for a given application.

The BLKSIZE = Option

The BLKSIZE = option defines the default block size of data libraries. You should consider the tradeoffs when choosing a block size. For example, if your members have a large number of large observations, choose the largest optimum block size that your device supports, for example, 2048 observations per 3380, or 5120 observations per 3350. If your members have a small number of small observations, or if you have members whose size and number of observations are widely varied, choose a smaller block size, for example, 6144.

The BUFSIZE = Option

The BUFSIZE = option is used to set the page buffer size for a SAS data set. A page consists of one or more blocks. The BUFNO = option is used to specify how many page buffers to allocate for an open SAS data set. Some tradeoffs need to be taken into consideration. More data is transferred per I/O operation with large pages; therefore, the EXCP count will be lower. The disadvantage is that more memory is required for large pages, which is reflected in the task and total memory statistics. Additional buffers can mean a lower EXCP count when accessing pages more than once, such as scrolling backwards in FSEDIT, or processing very small SAS data sets. They can also substantially reduce the elapsed time for a SAS job that processes mostly sequential data.

The COMPRESS = Option

The COMPRESS = option, a new, optional observation format that uses variable-length observations, is used to set the SAS data set observation type, either uncomplemented or compressed. In general, a SAS data set with compressed observations requires less mass storage than the same data set with uncompressed observations. Observation compression and the additional bookkeeping needed to manage the reuse of deleted observation space requires more CPU time.

Effect of the BLKSIZE = and BUFNO = Options on Performance

Review of user-supplied SMF information from sites running an experimental SMF exit on Release 5.18 shows the SAS usage profile displayed in Table 1. The data is normalized within site usage and then averaged over sites.

<table>
<thead>
<tr>
<th>Usage Data CPU Time</th>
<th>Usage Data I/O Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC</td>
<td>Sites</td>
</tr>
<tr>
<td>SASDATA</td>
<td>9</td>
</tr>
<tr>
<td>SORT</td>
<td>5</td>
</tr>
<tr>
<td>COPY</td>
<td>6</td>
</tr>
<tr>
<td>TABLES</td>
<td>3</td>
</tr>
<tr>
<td>FREQ</td>
<td>6</td>
</tr>
<tr>
<td>NAMES</td>
<td>6</td>
</tr>
<tr>
<td>RDATA</td>
<td>6</td>
</tr>
<tr>
<td>POINT</td>
<td>6</td>
</tr>
<tr>
<td>SUMMARY</td>
<td>9</td>
</tr>
<tr>
<td>DISK</td>
<td>9</td>
</tr>
<tr>
<td>SQL</td>
<td>9</td>
</tr>
<tr>
<td>APPLID</td>
<td>8</td>
</tr>
<tr>
<td>REC</td>
<td>8</td>
</tr>
<tr>
<td>FORMAT</td>
<td>9</td>
</tr>
<tr>
<td>CHART</td>
<td>9</td>
</tr>
<tr>
<td>DETECT</td>
<td>9</td>
</tr>
<tr>
<td>TASKLIST</td>
<td>9</td>
</tr>
<tr>
<td>DISPLAY</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 1 SAS Usage Data Averaged Across Nine Sites
Table 1 shows the effect of the SORT procedure, significant performance improvement can be experienced systemwide. Table 2 shows the effect of the BUFNO= and BUFSIZE= options in conjunction with the DCB parameter BLKSIZE. Resource utilization for two SAS steps is reported. The first step is a DATA step in which the first 50,000 observations are selected from a data set containing 105,434 observations. The second step is to sort the 50,000 observations in one numeric field. The SAS data set contains 110 variables and has an observation length of 1631 bytes.

**Table 2: Effect of the BLKSIZE=, BUFNO=, and BUFSIZE= Options on CPU Usage (One Variable)**

Running the same SAS steps on the same data set, selecting only 4 variables and all 105,434 observations, results in a data set with an observation length of 90 bytes. The impact of the BUFNO=, the BUFSIZE=, and the BLKSIZE= options is shown in Table 3.

**Table 3: Effect of the BLKSIZE=, BUFNO=, and BUFSIZE= Options on CPU Usage (Four Variables)**

Several problems are associated with the choice of a large block size or buffer size. For small data sets, a complete block or buffer must be written to the DASD. Even more serious is that for each code stream that SAS generates, a temporary data set is opened. Hence a large block size can cause large amounts of data to be transferred needlessly. The following DATA step generates three code streams:

```sas
DATA A;
SET B;
RUN;
```

If BY-processing is used, then two code streams are generated for each input data set and one code stream for each output data set, as well as a code stream for the body of the DATA step. The problem of opening multiple utility data sets for code generation has been solved in Release 6.07's family 3. Performance improvements of 15-20 percent with small DATA steps opening multiple code streams are possible. Because each member of a SAS data library uses at least one block, it is recommended that catalogs and temporary data sets be separated into different data libraries. If it is impractical to separate catalogs and data sets, then use the BUFNO= and BUFSIZE= data set options to improve performance of large data sets with BLKSIZE=6144.

**USING THE STORED DATA STEP FEATURE**

Another area where a new feature in Release 6.06 provides performance gains is the stored DATA step. With the move to a ninety percent portable system, several areas of Version 5 performance were severely impacted. One is the lexing and tokenization of DATA step code. The scanner and parser used in Release 6.06 is completely portable, but is much slower than Version 5. One worst-case example has a factor of four times slower. In this example, 2,500 lines of DATA step code are used to generate a report reading one record from a VSAM file. The SAS step is dominated by the lexing and tokenization. If a stored DATA step program is used, the execution of the stored program is approximately as fast as Version 5. For Release 6.07's family 3, work is under way to use host-specific features to improve performance of the SAS parser.

**MVS/ESA CONSIDERATIONS**

MVS/ESA is exploited by using hyperspace for work data sets. Work is in progress to determine performance benefits. Support for PDSEs is also in progress and may give additional performance benefits over PDSs.

**USE OF THE IBM VECTOR FACILITY**

Additional vector support is scheduled to be incorporated into Release 6.07's family 3. Candidate procedures are REG, SYSLIN, MEANS, SUMMARY, and IML. Preliminary data from an experimental version of the MEANS procedure indicate that you receive a ten percent performance improvement around 500 observations with one variable. When the number of variables is increased to 20, the ten percent improvement begins around 25 observations. Estimates of maximum performance improvement for number of variables are shown in the following table:

<table>
<thead>
<tr>
<th>Variables</th>
<th>Maximum CPU Improvement Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12%</td>
</tr>
<tr>
<td>2</td>
<td>20%</td>
</tr>
<tr>
<td>5</td>
<td>30%</td>
</tr>
<tr>
<td>10</td>
<td>40%</td>
</tr>
<tr>
<td>20</td>
<td>50%</td>
</tr>
<tr>
<td>100</td>
<td>55%</td>
</tr>
</tbody>
</table>
Two types of performance improvement are offered in the IML procedure. The following functions are vectorized:

- matrix multiplication
- EIGEN, EIGVAL, EIGVEC
- INV (inverse)
- SOLVE \((Ax=b)\)
- DET (determinant)
- matrix power
- ROOT (Cholesky decomposition)
- HALF (Cholesky decomposition)
- SVD (singular value)
- SWEEP
- GSORTH (Gram-Schmidt orth)
- ECHelon
- HERMITE
- INVUPD

For those sites that have IBM's Engineering and Scientific Subroutine Library (ESSL), the following routines can be transparently used by SAS/IML software after local link editing of one image:

- matrix multiplication - OGEMULT subroutine
- EIGEN, EIGVAL, EIGVEC - OSLEY subroutine
- INV (inverse) - DGEICD subroutine
- SOLVE \((Ax=b)\) - DGEF, DGES subroutine
- DET (determinant) - DGEICD subroutine
- matrix power - DGEMULT subroutine
- ROOT (Cholesky decomposition) - DPPF subroutine
- HALF (Cholesky decomposition) - DPPF subroutine
- SVD (singular value) - DGESVF subroutine

Maximum performance improvement for large matrices can be up to a factor of 2.

CONCLUSION

In conclusion, Version 6 SAS software contains many features that can be used to tune the SAS System in different environments. Some of these features are bundling and SAS system options that help optimization, memory usage, SAS data libraries under MVS, sorting considerations, and other methods of optimization.

REFERENCES


SAS, SAS/ETS, SAS/IML, and SAS/OR are registered trademarks of SAS Institute Inc., Cary, NC, USA.

IBM is a registered trademark and MVS/ESA is a trademark of International Business Machines Corporation, Armonk, NY.