CHAPTER

10

Optimizing Performance

Introduction

The SAS System includes many features that can help you manage CPU, memory, and I/O resources effectively. This chapter describes features that are either specific to OS/390 or that have characteristics that are specific to OS/390. The information

presented in this chapter is applicable to your site whether you run SAS interactively or in batch mode.

For additional information about optimizing SAS performance under OS/390, see *Tuning SAS Applications in the MVS Environment*, by Michael Raithel (available through SAS Institute as part of the Books by Users program).

For information about optimizing SAS performance on any host operating system, see *SAS Language Reference: Concepts* and *SAS Programming Tips: A Guide to Efficient SAS Processing*.

Collecting Performance Statistics

Several SAS system options provide information that can help you optimize your SAS programs. The STATS system option enables writes to the SAS log. The FULLSTATS, MEMRPT, and STIMER system options can be specified in combination to select the statistics that are written to the SAS log.

STATS
: specifies that statistics are to be written to the SAS log. NOSTATS specifies that no statistics are to be written to the SAS log, regardless of the values of STIMER, MEMRPT, and FULLSTATS. STATS and NOSTATS can be asserted at any time during a SAS session.

STIMER
: specifies that the CPU time statistic is to be collected and maintained throughout the SAS session. If STATS and STIMER are in effect, then the CPU time statistic will be written into the SAS log for each task. If FULLSTATS, STATS, and STIMER are in effect, the statistics listed under FULLSTATS below will be written to the SAS log. STIMER must be specified at SAS invocation.

MEMRPT
: specifies that memory usage statistics are to be written to the SAS log. If STATS and MEMRPT are in effect, then the amount of memory used by each task and the total amount of memory used for the SAS session will be written into the SAS log. If FULLSTATS, STATS, and MEMRPT are in effect, then additional statistics will be written into the SAS log, as specified below for FULLSTATS. MEMRPT and NOMEMRPT can be specified at any time during a SAS session.

FULLSTATS
: specifies that additional statistics are to be written to the SAS log. The actual statistics added are determined by the values of STIMER and MEMRPT. If STIMER is in effect, then elapsed time, vector affinity time, RSM hiperspace time, and EXCP count are displayed. If MEMRPT is in effect, then for each task, both task and total memory are displayed, including the amount of memory used for data and amount of memory used for program. FULLSTATS and NOFULLSTATS can be specified at any time during a SAS session.

Logging SMF Statistics

SMF statistics are generated by IBM's System Management Facility. If your system is configured to enter the SMF exit, and if the SAS system options SMF and SMFEXIT= are in effect, up to 20 SMF statistics can be written to the SAS log for each task.

SMF statistics are written to the SAS log only when the STATS system option is in effect.

For further information on SMF statistics, see the installation instructions for SAS in the OS/390 environment.

Optimizing I/O

To optimize SAS input and output in the OS/390 environment, consider the following suggestions.

Put catalogs and data sets into separate libraries, using the optimal block size for each

The physical block size (BLKSIZE=)of a SAS bound data library determines both the minimum page size and the minimum unit of space allocation for the library. The 6KB default is relatively efficient across a range of device types, and it leads to lower memory requirements for catalog buffers. However, when you use the 6KB default, more DASD space is needed to hold a given amount of data because smaller blocks lead to capacity losses. In one test case on a 3380, an MXG daily PDB required 8% more tracks when it was stored in 6KB physical blocks instead of in half-track blocks.

Because the optimal block sizes for SAS catalogs and SAS data sets are not necessarily the same, consider putting catalogs and data sets into separate libraries. For catalog libraries, 6KB is a good general physical block size on any device. For data sets, choose either a full-track or half-track block size, depending on whether the data library is stored on a device that supports full-track blocks.

Use the optimal buffer size and buffer number for your data

When a member of a direct access bound library is processed sequentially, the values of the SAS system options BUFSIZE= and BUFNO= are the primary factors that affect I/O performance. When a SAS data library is processed sequentially, the unit of I/O transfer, in bytes, is equal to BUFSIZE*BUFNO.

BUFSIZE is the page size for the data set. You specify BUFSIZE only when you are creating an output data set; it then becomes a permanent attribute of the data set. BUFNO is the number of page buffers to allocate for the data set. For random access, BUFNO page buffers form a least-recently-used buffer pool that can significantly reduce physical I/O depending on the data-access pattern. Of course, the greater the number of page buffers, the more memory is required. Page buffers are stored above the 16MB line.

Note that the product of BUFNO and BUFSIZE is the important factor in sequential I/O performance rather than the specific value of either option. As BUFNO is increased, there is a marked reduction in I/O time and I/O count, although the cost of buffer storage increases. As a result, elapsed times can be significantly reduced. For example, when BUFNO=16 and BUFSIZE=6144, the results are very similar to BUFNO=4 and BUFSIZE=23040. Moreover, when BLKSIZE=6144, specifying BUFSIZE=24K yields performance results that are very close to those of BLKSIZE=23040 and BUFSIZE=23040.

Here are some guidelines for determining the optimal BUFSIZE and BUFNO values for your data:

- When BLKSIZE is set to a full- or half-track value, let BUFSIZE = BLKSIZE.
- Set BUFNO to at least 2.
- Do not allow BUFNO*BUFSIZE to exceed 135K (3 tracks for 3380 or 3390). Exceeding that value can be detrimental to other users of the system because long

channel programs can monopolize devices and channels. Moreover, beyond that limit, further reductions in elapsed time are negligible.

Determine whether you should compress your data

Compressing data reduces I/O and disk space but increases CPU time. Therefore, whether or not data compression is worthwhile to you depends on the resource cost-allocation policy in your data center. Often your decision must be based on which resource is more valuable or more limited, DASD space or CPU time.

You can use the portable SAS system option COMPRESS= to compress all data sets that are created during a SAS session. Or, use the SAS data set option COMPRESS= to compress an individual data set. Data sets that contain many long character variables generally are excellent candidates for compression.

The following tables illustrate the results of compressing SAS data sets under OS/390. In both cases, PROC COPY was used to copy data from an uncompressed source data set into uncompressed and compressed result data sets, using the system option values COMPRESS=NO and COMPRESS=YES, respectively.* In the following tables, the CPU row shows how much time was used by an IBM 3090-400S to copy the data, and the SPACE values show how much storage (in megabytes) was used.

For the first table, the source data set was a problem-tracking data set. This data set contained mostly long, character data values, which often contained many trailing blanks.

Table 10.1 Compressed Data Comparison 1

Resource	Uncompressed	Compressed	Change
CPU	4.27 sec	27.46 sec	+23.19 sec
Space	235 MB	54 MB	-181 MB

For the preceding table, the CPU cost per megabyte is 0.1 seconds.

For the next table, the source data set contained mostly numeric data from an MICS performance database. The results were again good, although not as good as when mostly character data were compressed.

Table 10.2 Compressed Data Comparison 2

Resource	Uncompressed	Compressed	Change
CPU	1.17 sec	14.68 sec	+13.51 sec
Space	52 MB	39 MB	-13 MB

For the preceding table, the CPU cost per megabyte is 1 second.

For more information about the pros and cons of compressing SAS data, see *SAS Programming Tips: A Guide to Efficient SAS Processing*.

* When you use PROC COPY to compress a data set, you must include the NOCLONE option in your PROC statement. Otherwise, PROC COPY propagates all the attributes of the source data set, including its compression status.

Consider using SAS software compression in addition to hardware compression

Some storage devices perform hardware data compression dynamically. Because this hardware compression is always performed, you may decide not to enable the SAS COMPRESS option when you are using these devices. However, if DASD space charges are a significant portion of your total bill for information services, you might benefit by using SAS software compression in addition to hardware compression. The hardware compression is transparent to the operating system; this means that if you use hardware compression only, space charges are assessed for uncompressed storage.

Consider placing SAS data libraries in hiperspaces

One effective method of avoiding I/O operations is to use the SAS System's HIPERSPACE engine option. This option that is specific to OS/390 enables you to place a SAS data library in a hiperspace instead of on disk.

A hiperspace overrides the specified physical data library. This means that the physical data library on disk is neither opened nor closed, and data are neither written to nor read from the data library. All data access is done in the hiperspace.

Because the specified data library is not written to, it should be a temporary data set. The only time the specified data library is used is when it is a DIV (Data-In-Virtual) data set, as explained in "Hiperspace Libraries and DIV Data Sets" on page 154.

The HIPERSPACE option is processed after the normal allocation processing is complete. The requested data set is allocated first, as it is with any LIBNAME statement or LIBNAME function. It is deallocated when you issue a LIBNAME CLEAR statement or when you terminate the SAS session. The hiperspace, in effect, overrides the data set.

Examples of Using the HIPERSPACE Engine Option

Here is an example of using the HIPERSPACE engine option to place a data library in a hiperspace:

```
libname mylib '&templib' hip;
```

(HIP is an alias for the HIPERSPACE option.)

For a data library that was allocated externally with a DD statement or a TSO ALLOCATE command, specify a null data set name in quotes. For example, the following LIBNAME statement places a library that was allocated with the DDname "X" in a hiperspace:

```
libname x '' hip;
```

To place the WORK data library in a hiperspace, specify the HSWORK SAS system option when you invoke SAS. See "HSWORK" on page 361 for a description of the HSWORK option.

Controlling the Size of a Hiperspace Library

Just as you use the SPACE=, DISP=, and BLKSIZE= engine options to allocate a physical data set, you use the HSLXTNTS=, HSMAXPGS=, and HSMAXSPC= SAS system options to control the size of hiperspace libraries. These options are described in "HSMAXPGS=" on page 359.

The CONTENTS procedure reports all hiperspace libraries as residing on a 3380 device with a block size of 4096. These attributes may differ from the attributes of the physical data set.

Hiperspace Libraries and DIV Data Sets

The only time the allocated physical data set is actually used with the HIPERSPACE option is if the data set is a Data-In-Virtual (DIV) data set. * An empty DIV data set can be initialized by allocating it to a hiperspace library. An existing DIV data set that contains data can be read or updated, or both.

You can use the HSSAVE SAS system option to control whether the DIV pages are updated each time your application writes to the hiperspace or only when the data library is closed. See "HSSAVE" on page 361 for more information about this option.

Performance Considerations for Hiperspace SAS Data Sets

The major factor that affects hiperspace performance is the amount of expanded storage on your system. The best candidates for using hiperspace are jobs that execute on a system that has plenty of expanded storage. If expanded storage on your system is constrained, the hiperspaces are moved to auxiliary storage. This eliminates much of the potential benefit of using the hiperspaces.

For more information about using hiperspaces under OS/390, see the documentation for your operating environment. Also see *Tuning SAS Applications in the MVS Environment*.

Consider designating temporary SAS libraries as virtual I/O data sets

Treating data libraries as "virtual I/O" data sets is another effective method of avoiding I/O operations. This method works well with any temporary SAS data library–especially WORK. To use this method, specify UNIT=VIO as an engine option in the LIBNAME statement or LIBNAME function.

The VIO method is always effective for small libraries (<10 cylinders). If your installation has set up your system to allow VIO to go to expanded storage, then VIO can also be effective for large temporary libraries (up to several hundred cylinders). Using VIO is most practical during evening and night shifts when the demands on expanded storage and on the paging subsystem are typically light.

The VIO method can also save disk space because it is an effective way of putting large paging data sets to double use. During the day, these data sets can be used for their normal function of paging and swapping back storage; during the night, they become a form of temporary scratch space.

Efficient Sorting

Consider changing the values of SORTPGM= and SORTCUTP=

The SAS System includes an internal sort program that is often more efficient than host sort programs for sorting small volumes of data. Host sort programs are generally more efficient when the data volume is too high to perform the sort entirely in memory.

Under OS/390, the default value of the SAS system option SORTPGM= is BEST. This value causes SAS to use the SAS sort program for less than 4M of data; for more than 4M of data, SAS uses the host sort program. You use the SORTNAME= system option to specify the name of the host sort program.

* DIV data sets are also referred to as VSAM linear data sets.

The 4M limit is the default value that is specified by the SORTCUTP= system option, which is specific to OS/390. You may want to change the value of this option in order to optimize sorting for your particular applications.

Take advantage of the DFSORT performance booster

If your installation uses Release 13 or later of IBM's DFSORT as its host sort utility for large sorts, then you can take advantage of a DFSORT "performance booster." To do so, specify SORTBLKMODE in an OPTIONS statement, in the OPTIONS parameter list of the SAS cataloged procedure, or in a configuration file.

SORTBLKMODE causes SAS to work in conjunction with DFSORT to process your SAS sorting applications faster. SAS applications that use either PROC SORT or PROC SQL for sorting can take advantage of this "performance booster." For large sorts of approximately 100,000 observations or more, CPU usage may be reduced by up to 25%.

Some SAS System Options That Can Affect Performance

MAUTOSOURCE and IMPLMAC

These two SAS system options affect the operation of the SAS autocall macro facility, and they interact in a way that you should be aware of.

Specifying IMPLMAC enables you to use statement-style macros in your SAS programs. With IMPLMAC in effect, each SAS statement is potentially a macro, and the first word (token) in each statement must be checked to determine whether it is a macro call.

When IMPLMAC is in effect without MAUTOSOURCE, no special checking takes place until the first statement-style macro is compiled. When both IMPLMAC and MAUTOSOURCE are in effect, however, this checking is done unconditionally. The initial occurrence of a word as the first token of a SAS statement results in a search of the autocall library. There can be a significant number of directory searches, especially when a large DATA step is compiled, in addition to the CPU time that is consumed by maintaining and searching the symbol table.

The combination of MAUTOSOURCE and IMPLMAC can add 20% to CPU time and 5% to I/O for a non-trivial job. Therefore, for best performance, leave NOIMPLMAC as the installation default.

REXXMAC

When SAS encounters an apparent SAS statement that it does not recognize, it typically generates a "statement is not valid" error message in the SAS log. However, when the REXXMAC system option is in effect, SAS passes the first word in the apparent statement to the OS/390 REXX processor, which looks for a member by that name in the SASREXX library. Hence, a mistyped statement could have unintended results and could have a negative impact on performance. For more information, see “REXXMAC” on page 390 and “REXXLOC=” on page 389.

SPOOL/NOSPOOL

The SPOOL system option is appropriate when you are running SAS interactively, without using the windowing environment. When SPOOL is in effect, SAS input

statements are stored in a WORK library utility file; they are retrieved later by %INCLUDE and %LIST commands. SAS is shipped with SPOOL as the default setting for interactive sessions, but you may want to consider resetting it to NOSPOOL for batch jobs. In a batch job that has a large number of input lines, NOSPOOL can reduce I/O by as much as 9%.

VECTOR/NOVECTOR

The SAS system option VECTOR enables you to use the IBM 3090 Vector Facility for certain SAS procedures–most notably, PROC GLM. This option is in effect in both of the configuration files that are shipped with the SAS System. However, your site administrator may have reset this option to NOVECTOR if your data center limits the use of the vector facility. Therefore, you should check with your site administrator before enabling this option.

Managing Memory

In the OS/390 operating environment, you can use two options to limit the amount of memory used by SAS. The MEMSIZE= option sets a hard limit on the amount of memory used by SAS. The MEMLEAVE= option limits SAS memory relative to the user's available region of memory. SAS Institute recommends that you specify a value for the MEMLEAVE= option and let SAS determine the value of MEMSIZE= based on your REGION size and on the value of MEMLEAVE=.

The following sections provide details on available memory management techniques.

Specify a value for MEMSIZE= when you invoke SAS

For most programs and data, SAS uses memory (storage) above the 16MB line. This is sometimes referred to as 31-bit addressing. The few exceptions include data areas that require 24-bit addressability and a small part of the host supervisor. Because the OS/390 JCL parameter REGION is ineffective at controlling 31-bit memory usage, the SAS MEMSIZE= option has been implemented for this purpose.

One way to limit the amount of virtual memory that SAS can use is to specify a value for the MEMSIZE= system option when you invoke SAS. Under OS/390, MEMSIZE= has a default value of 0, which means that SAS can use memory up to the maximum amount that is available. This enables you to invoke multiple procedures consecutively without reloading the program each time. However, if there is no upper limit on virtual memory usage, then memory is not freed for reuse even when the programs that are stored there are no longer needed.

The default value for the MEMSIZE= option is calculated internally using the following formula:

```
MEMSIZE_default = size_of_user_memory_region --- MEMLEAVE_value
```

For interactive applications that use multiple SAS System components such as SAS/AF and SAS/GRAPH, specify MEMSIZE=24M. If you prefer to set the value of the MEMSIZE= option yourself, a value of 12M is a reasonable choice for most batch applications.

Specify a value for MEMLEAVE=

As with MEMSIZE=, the MEMLEAVE= system option limits the amount of memory used by SAS, but in a different way. Instead of setting an absolute limit on the amount of memory SAS can use, MEMLEAVE= specifies a value that is subtracted from the total amount of memory available in the user's REGION. The amout of memory specified by MEMLEAVE= is reserved for the use of the operating environment. The remainder of the user's REGION remains available to SAS. The advantage provided by MEMLEAVE= is that it applies equally well to all SAS sessions, regardless of the size of the REGION. Choosing an appropriate value for MEMSIZE=, on the other hand, can be problematic in environments where REGION sizes vary.

The default value of MEMLEAVE= is 512K. You might need to increase this value depending on memory demands expected for host programs running in the same REGION, to prevent SAS from using too much of that REGION. For example, you might want to increase the value of MEMLEAVE= if you plan to run a memory-intensive host sort routine while also running a large SAS session.

If you specify a value for the MEMLEAVE= option, either do not specify a value for the MEMSIZE= option or set the value of the MEMSIZE= option to 0.

Consider using superblocking options to control memory fragmentation

Superblocking options are SAS system options that set aside large blocks of memory for different classes of use. In most cases, the default values for these options are appropriate and should not be altered. However, if you receive a superblock-overflow warning message in the SAS log, you may want to use these options to adjust the memory allocation for your job.

For complete information on superblocking system options, see the installation instructions for the SAS System in the OS/390 environment. You can also submit the following SAS statement to list the superblocking system options:

```
proc options group=memory;
run;
```

Use SYSLEAVE= and PROCLEAVE= to handle out-of-memory conditions

Sometimes a job runs out of memory in spite of additional memory allocations. To ensure that the job ends "gracefully" under that condition, you may want to increase the values of the SAS system options SYSLEAVE= and PROCLEAVE=.

- The SYSLEAVE= option reserves a specified amount of memory to ensure that, when a SAS task ends, enough memory is available to close data sets and to "clean up" other resources. For details, including the SAS default value, see “SYSLEAVE=” on page 412.
- The PROCLEAVE= option serves a similar function for SAS procedures. For example, some procedures are designed to use memory until no more is available; they then continue by opening and using work files. PROCLEAVE= ensures that there will be enough memory left to open these work files and to allocate I/O buffers for them so that the procedure can continue. For details, including the SAS default value, see “PROCLEAVE=” on page 388.

Loading SAS Modules Efficiently

Use a bundled configuration of SAS

The SAS System has three possible program configurations:

- unbundled
- bundled (LPA/ELPA version).
- bundled (non-LPA version)

In an unbundled configuration, all modules are loaded individually from the SAS System load library. Running in this manner is not generally recommended because it significantly increases library-directory searches and I/O. However, SAS is shipped with this setting by default because some of the installation tasks must invoke SAS before the installer has had the opportunity to select a bundled version.

In the two bundled configurations of SAS, many individual modules are combined into one large executable file. Invoking a bundled version of SAS eliminates both wasted space between modules and the overhead of loading each module individually. Performance is also improved slightly.

In a multiuser SAS environment, the most effective way to reduce memory requirements is to use the LPA/ELPA bundled configuration. This configuration dramatically reduces each user's working-set size. *

The non-LPA bundled configuration is intended for sites that do not want to place SAS modules in the Link Pack Area. In this configuration, the bundle is loaded into each user's address space. Although this decreases library-directory searches and I/O, it has the unfortunate side-effect of increasing individual working-set sizes. Therefore, this method is not recommended if you have many SAS users at your site.

For detailed information about the bundled configurations and how to install them, see the installation instructions for the SAS System in the OS/390 environment.

Other Considerations for Improving Performance

Leave AUTOSCROLL 0 in effect for the LOG and OUTPUT windows

The AUTOSCROLL command controls how information is scrolled as it is written to the LOG and OUTPUT windows. Specifying small scrolling increments is very expensive in terms of response time, network data traffic, and CPU time.

Under OS/390, AUTOSCROLL is preset to 0 for the LOG window. AUTOSCROLL 0 suppresses automatic scrolling and positions the LOG window at the bottom of the most recent output when a DATA step or procedure is completed. At that time, of course, you can scroll up to view the contents of the log.

To see the effect of this command, enter AUTOSCROLL 1 on the command line of the LOG window and then run PROC OPTIONS. Then enter AUTOSCROLL 0 and run PROC OPTIONS again. The CPU time ratio is more than 30 to 1.

* Working-set size is the amount of real system memory that is required to contain a) the programs that consume most of system execution time, and b) the data areas that these programs reference.

Use the EM3179 device driver when appropriate

If you are running Attachmate or any other full-functioned 3270 emulator over a slow connection, specify the SAS system option FSDEVICE=EM3179 when you invoke SAS. Menus in applications such as SAS/ASSIST are then displayed as text menus instead of icon menus. The text menus require much less network data transfer and are considerably faster across slow lines.

Consider using the direct log-on procedure to invoke SAS

When you use the direct log-on procedure to invoke SAS instead of the TSO log-on procedure, SAS acts as your terminal monitor program. The direct log-on procedure has three potential advantages for your installation:

- It eliminates the need for SAS users to know anything about TSO.
- It saves a small amount of memory (approximately 50KB per user) in working-set size.
- If you license TSO/E as a measured usage product, then you may be able to reduce your TSO charges significantly because CPU time for SAS applications will no longer be accumulated as TSO/E usage.

For a sample log-on procedure and other information about configuring it into the environment at your site, see the installation instructions for the SAS System in the OS/390 environment.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS® Companion for the OS/390 Environment, Version 8*, Cary, NC: SAS Institute Inc., 1999.

SAS® Companion for the OS/390® Environment, Verison 8

1–58025–523–X

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, November 1999