Tuning the SAS® System under CMS

OBJECTIVES

The objectives of this paper are to acquaint you with the SAS system under CMS environment and with other external tuning considerations. Your choice of option settings and tuning decisions can substantially affect the resource requirements required to run SAS programs. Topics we will cover include:

- program management considerations
- SAS System I/O considerations
- macro processing options
- other SAS system options
- sort considerations
- PL/I ISASIZE.

PROGRAM MANAGEMENT CONSIDERATIONS

Program management is a very important tuning area within the SAS system under CMS. There are a several things you can do with saved segment implementation and load library management to improve performance significantly.

Saved Segments

Shared saved segments Using shared saved segments in a multiuser community reduces the real memory, CPU, and I/O resources required for program management. There are four shared segments available with release 5.16. These are (1) an 896k supervisor segment, (2) a 384k segment for SAS/FSP® software, (3) a 1344k segment for PROC FSCALC, and (4) a 704k segment for SAS/AF® software.

You may be wondering how much real memory is required by these segments. The answer depends on (1) how actively the segments are being used, (2) what procedures or SAS system features are being used, and (3) the overall contention level for memory on your machine. On SAS Institute's production CMS system, approximately two-thirds of the supervisor segment and one-third of the SAS/FSP segment are resident.

The savings resulting from installing shared segments can be dramatic. We make very heavy use of PROC FSEDIT on our production SAS system. When we moved to release 5.16 with the shared SAS/FSP segment, we noticed significant benefits to all users during peak periods as shown in the following table:

<table>
<thead>
<tr>
<th></th>
<th>response time</th>
<th>paging per sec</th>
<th>average working set</th>
</tr>
</thead>
<tbody>
<tr>
<td>before</td>
<td>0.3</td>
<td>3.8</td>
<td>20</td>
</tr>
<tr>
<td>after</td>
<td>0.2</td>
<td>2.5</td>
<td>16</td>
</tr>
</tbody>
</table>

I highly recommend that you install at least the supervisor shared segment. If you have significant use of SAS/FSP or SAS/AF software at your site, install those segments also. In Version 5, saved segments are built from the load module code, so maintenance headaches associated with segments in release 8.2.3 do not exist. You can apply maintenance to the load module libraries, test it there, and then rebuild the saved segments from the load module libraries.

Nonshared segments You can also elect to install the nonshareable portions of the SAS System in saved segments. There are no real memory savings to be had by doing this, but there could be significant I/O savings. Also, depending on how your billing algorithms work, users may incur lower charges when running with nonshared saved segments.

Load Library Management Considerations

BLDLTABLE option When looking for a program or format, the SAS system searches (1) saved segments, (2) load module libraries, (3) text files, (4) text libraries and finally, (5) module files. Even with full implementation of saved segments, there can still be a considerable amount of search activity against library directories on the system disk. The BLDLTABLE option reduces this search activity by maintaining a table of program names and locations on an LRU basis. When the option was implemented on the Institute's production SAS system, I/O to the SAS system disk was reduced by about one-third. The table size is 32 entries.

Consolidated load library As shipped, each SAS software product resides in a separate library on the SAS system disk. You can reduce directory search time by consolidating the multiple libraries into a single library. Beginning in release 5.16, the SAS System keeps the first directory block of the most recently referenced load module library in memory. If there is a load library concatenation, the effectiveness of this in-memory directory block is severely limited.

Cached DASD The SAS system disk is an excellent candidate for cached DASD (e.g. IBM 3880-13 or 3880-23 control units) should you be lucky enough to have such devices. We achieved over a 95% hit ratio on a 3880-23 with an average service time of 6 ms. Even if you are running from segments, the SAS system disk can still account for a significant I/O load. When running from segments, though, very little cache is needed to support the SAS system disk because only a very few tracks are heavily accessed.

SAS SYSTEM I/O CONSIDERATIONS

There are a number of factors that influence SAS System I/O efficiency. Chief among these are logical block size, physical block size, and library format.

SAS BLKSIZE Option

SAS System performance can be greatly affected by logical block size. The following table shows the results obtained when creating and then reading a SAS data set containing thirty-thousand 200-byte observations on an IBM 4381-R14:

<table>
<thead>
<tr>
<th>Logical Block Size Comparisons</th>
<th>Write</th>
<th>Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write</td>
<td></td>
<td></td>
</tr>
<tr>
<td>block size</td>
<td>elapsed</td>
<td>vtime</td>
</tr>
<tr>
<td>block size</td>
<td>sec</td>
<td>sec</td>
</tr>
<tr>
<td>2k</td>
<td>33</td>
<td>3.2</td>
</tr>
<tr>
<td>4k</td>
<td>31</td>
<td>2.6</td>
</tr>
<tr>
<td>8k</td>
<td>30</td>
<td>2.5</td>
</tr>
<tr>
<td>16k</td>
<td>18</td>
<td>2.2</td>
</tr>
<tr>
<td>32k</td>
<td>9</td>
<td>2.1</td>
</tr>
</tbody>
</table>

217
In general, I recommend setting your system BLKSIZE default to 16k or 32k. Please note that when using the new Version 5 library format, specifying a large block size will not cause small data sets to require extra disk space (see Library Format Considerations).

Minidisk Physical Block Size

Specify a 4K block size for your minidisks that are to contain SAS libraries. This specification gives you the best space utilization in release 82.3 and requires substantially less resources. Further­
ences observed when creating data sets in bOth formats with different minidisk block sizes:

The following table illustrates the performance differences observed when creating data sets in both formats with release 5.16:

<table>
<thead>
<tr>
<th>physical block size</th>
<th>elapsed logical block size</th>
<th>virtual logical block size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1k</td>
<td>19 sec</td>
<td>2.3 sec</td>
</tr>
<tr>
<td></td>
<td>2.3 sec</td>
<td>3.0 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2k</td>
<td>12 sec</td>
<td>2.1 sec</td>
</tr>
<tr>
<td></td>
<td>2.1 sec</td>
<td>2.5 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4k</td>
<td>9 sec</td>
<td>2.1 sec</td>
</tr>
<tr>
<td></td>
<td>2.1 sec</td>
<td>2.4 sec</td>
</tr>
</tbody>
</table>

Library Format Considerations

SAS data library I/O is markedly faster in Version 5 than it was in release 82.3 and requires substantially less resources. Further­
more, the new format requires less DASD space, especially for small files. The minimum space required for a file under the old format is two times logical block size. Therefore, there was a defi­
inite disadvantage to using a large block size as the default under that format. The following table illustrates the performance differ­
ences observed when creating data sets in both formats with release 5.16:

<table>
<thead>
<tr>
<th>physical block size</th>
<th>release number</th>
<th>elapsed logical block size</th>
<th>virtual logical block size</th>
</tr>
</thead>
<tbody>
<tr>
<td>2k</td>
<td>82.3</td>
<td>573 sec</td>
<td>13.8 sec</td>
</tr>
<tr>
<td></td>
<td>5.16</td>
<td>33 sec</td>
<td>3.2 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4k</td>
<td>82.3</td>
<td>280 sec</td>
<td>7.8 sec</td>
</tr>
<tr>
<td></td>
<td>5.16</td>
<td>31 sec</td>
<td>2.8 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8k</td>
<td>82.3</td>
<td>128 sec</td>
<td>4.9 sec</td>
</tr>
<tr>
<td></td>
<td>5.16</td>
<td>30 sec</td>
<td>2.5 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16k</td>
<td>82.3</td>
<td>73 sec</td>
<td>3.4 sec</td>
</tr>
<tr>
<td></td>
<td>5.16</td>
<td>18 sec</td>
<td>2.2 sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32k</td>
<td>82.3</td>
<td>44 sec</td>
<td>2.8 sec</td>
</tr>
<tr>
<td></td>
<td>5.16</td>
<td>9 sec</td>
<td>2.1 sec</td>
</tr>
</tbody>
</table>

There is virtually no difference in the READ performance of the two formats under Version 5. Version 5 does, however, read old format data sets more efficiently than release 82.3 did.

Version 5 of the SAS System will create new libraries in the new format; however, libraries created under release 82.3 will continue to be maintained in the old format. You will clearly bene­
fit by converting libraries created under release 82.3 to the new format.

Putting it all together...

SAS Institute's MIS department applied the above I/O tuning recommendations to some of the CMS batch applications that run at night and subsequently reported the following improve­
ments in elapsed times:

<table>
<thead>
<tr>
<th>application</th>
<th>before changes</th>
<th>after changes</th>
<th>improvement ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;A&quot;</td>
<td>3.5</td>
<td>1.2</td>
<td>2.9</td>
</tr>
<tr>
<td>&quot;B&quot;</td>
<td>2.0</td>
<td>1.2</td>
<td>1.7</td>
</tr>
<tr>
<td>&quot;C&quot;</td>
<td>3.0</td>
<td>2.0</td>
<td>1.5</td>
</tr>
<tr>
<td>&quot;D&quot;</td>
<td>4.0</td>
<td>1.5</td>
<td>2.7</td>
</tr>
</tbody>
</table>

Other I/O considerations

OS read-only SAS libraries There is special channel program level code in the CMS SAS System to read SAS data libraries in OS format. Since these libraries can have large physical block sizes, they can be read very efficiently. Therefore, if you have both MVS and VM systems that can conveniently share DASD, there are both performance and maintenance advantages to using a common OS copy of such things as SAS/AF menus, CBT libraries, and SAS/GRAPH® map data sets.

Virtual I/O You can use the VIOBUF=nK option to specify that up to nK of work data sets will reside in virtual storage rather than on disk. This can be a very effective way of eliminating much work data set I/O provided your working sets can be increased enough to support the virtual I/O. It does not do much good to replace work I/O with paging I/O! The default shipped with release 5.08 was 256k; however, it appears that 128k is a better general default and that value is the default in CMS releases 5.12 and above.

Please note that the VIOBUF storage does not hold a dynamic pool of work data set buffers, but rather contains up to nK of work data sets. To use virtual I/O most effectively, you should scratch work data sets as you finish with them so that the virtual I/O buffer can be allocated to active data sets. If you are using SAS Display Manager, clear your log window periodically because it, too, is a work data set. (The CLEAR command is available in releases 5.12 and above.)

MACRO PROCESSING OPTIONS

IMPLMAC and MAUTOSOURCE Options

One of the significant enhancements included in Version 5 SAS software is an autocall facility for macros. It is no longer neces­
sary to explicitly include each macro you want to use. Macros may instead be pulled in and compiled on demand from a macro
library. This facility can save resources in that macros will not be retrieved and compiled unless required. The macro autocall facil­
ity is turned on by specifying the MAUTOSOURCE option.

More than one macro may be stored in a single library member provided that one macro in the group matches the library member name. This can be a handy way to bring in a group of related mac­
ros with one directory search.

There is an Interaction of MAUTOSOURCE with another option, IMPLMAC, that you should be aware of. Specification of IMPLMAC allows you to use statement-style macros in your SAS programs. With IMPLMAC in effect, each SAS statement is poten­
tially a macro and the first word (token) in each statement must be checked to see if 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
MAUTOSOURCE is on, however, this checking must be 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, espe-
cially during the compilation of a large DATA step, in addition to
the CPU time necessary to maintain and search the symbol table.
The combination of MAUTOSOURCE and IMPLMAC can add
10% or more to the CPU and I/O resources required for a job.
As an example, the compilation of a SAS program containing
about 420 statements required 95 autocall library directory
searches and an additional 25% CPU time. For best performance,
you should set NOIMPLMAC as the installation default.

MSIZE and MLEAVE options
When macros are compiled, they are stored in the work data
library in 2048-byte execution frames. These frames are brought
into memory during macro execution. If insufficient storage has
been set aside to hold the execution frames, extra work data set
I/O is incurred during macro execution to support frame paging.
(MSIZE-MLEAVE/2k) gives the number of memory frames avail-
able for macro processing. The default is three. The cost of hav-
ing too few macro execution frames can be significant while the
cost of overallocation is trivial. A good set of installation defaults
would be MSIZE=128k and MLEAVE=64k.

OTHER SAS SYSTEM OPTIONS

Number of history generations The GEN=n option specifies
how many generations of history data are to be kept for SAS data
sets. When GEN is nonzero, the SAS source statements used to
create the data set are stored in the directory. This storage of
history information can take up considerable library space and
add significantly to the number of I/O operations required to run
a SAS program. If you specify GEN=0, you can expect to reduce
I/O by 10% or more. The amount of space saved by GEN=0 will
vary over a wide range. I have seen extreme cases in which the
data occupied less than 2% of the space required for a data set;
the rest of the space contained history information.

There are times, of course, when keeping history information is
entirely appropriate, perhaps in studies where it is necessary to
keep an audit trail of how the data was processed. Ideally, it
would be best to set GEN=0 as your installation default and have
no history information. The rest of the space contained history
information.

When a PL/I program is started, PL/I allocates an Initial Storage
Area (ISA) to be used for dynamic storage requests. The size of
this area is determined by ISASIZE as coded in the program or
as passed by an execution-time parameter. If the ISA is too small,
oddly enough, this option does have some performance impact
because one extra GETMAIN and two extra FREEMAINs are
done by memory management for each storage request when the
LEAVE option is nonzero. Even when using procedures that
issue many memory management requests, the impact of this
overhead appears to be small. Some tests were run and the CPU
time differences (total and virtual) were very small. However, the
overhead may show up more significantly in other situations.
Therefore you should set LEAVE to 0 unless you have a demon-
strated reason to do otherwise.

PROC FSEDIT AUTOSAVE command The FSEDIT procedure
provides the AUTOSAVE command to allow you to control how
often checkpointing is done by closing and reopening the data
set. The system default is to do this every 25 observations;
hence, you will lose no more than 25 observations if the system
crashes during an editing session. There is a definite tradeoff
between overhead and data loss (or re-entry) potential in setting
this option. Setting AUTOSAVE to 1 costs about five I/Os per
observation with a single variable data set in a small directory.

Display manager AUTOROLL LOG ON/OFF In release 5.08,
each line written to the log window caused a separate I/O to the
terminal. The most noticeable effect of this was heavy data traffic
and slow response on remote lines. Local users did not usually
notice a response problem due to the many terminal I/Os. How-
ever, the CPU time cost was heavy in both cases. A new option,
AUTOROLL LOG ON or OFF, is available in releases 5.12 and
above. ON causes the system to behave as it does in release
5.08, while OFF causes the log window to be updated only when
you are prompted for input. At that time, of course, you may scroll
the log backwards to see what transpired since your last input.
The system default is AUTOROLL LOG OFF. To convince your-
selves this should be your installation default, run PROC OPTIONS
both ways and compare the results. The CPU time ratio is about
10:1.

SPOOL/NOSPOOL option The SPOOL option is intended to be
used when running SAS in batch and little reason
is stored in the WORK library for later retrieval by the %IN-
CLUDE command. You should set NOSPOOL when running
under the display manager because the RECALL command pro-
vides similar function.

SORT CONSIDERATIONS

FILSZ parameter If your system sort utility supports the FILSZ
parameter, set FILSZ in your SAS system options. This option
can improve sorting efficiency because it will cause the sort to
be given more accurate information about the number of records
to be sorted.

SORT versus SORTT You can use either your own system sort
utility or one supplied with the SAS System to sort SAS data sets.
Your system sort is used by setting SORTPGM = NAME and invok-
ing PROC SORT. The SAS System sort may be used by setting
SORTPGM = SAS or by invoking PROC SORTT. PROC SORTT
is efficient when sorting small data sets. However whether it is
more efficient than your system sort depends on what utility it is
and how it is installed.

SORTPGM=SAS versus SORTPGM=NAME The system
sort utility in the SAS System is a version of %SORT, the
SAS System sort utility. When you use your own system sort
utility, you must specify SORTPGM=SAS or SORTPGM=NAME.
SORTPGM=SAS can be used only with SAS 6.05 and later.
SORTPGM=NAME can be used with SAS 6.04 and later.

Comparisons run on the SAS Institute production CMS system
against SyncSort™ release 6.0c installed in a shared segment
show the crossover point comes at about 300 observations (ob-
server size was 232 bytes). The same tests run last year
against an earlier SyncSort release reached a crossover point at
about 2000 observations. If you have CMS users who are sorting
a large number of small data sets, it may be worthwhile to set
SORTPGM=SAS in the interactive default.

PL/I ISASIZE When a PL/I program is started, PL/I allocates an Initial Storage
Area (ISA) to be used for dynamic storage requests. The size of
this area is determined by ISASIZE as coded in the program or
as passed by an execution-time parameter. If the ISA is too small,
additional operating system storage management overhead is
incurred. ISASIZE may be specified using the SAS PARM option
as follows: PARM(ISASIZE=E(nK)). In most cases, the ISA size
internally specified in SAS procedures should be sufficient, but there may be some cases where it is not. If you want to be on the safe side, specify an ISA of 64k. A report on ISA usage may also be gathered by coding REPORT in the PARM option and including a PUDUMP FILEDEF. See the appropriate IBM PL/I Optimizing Compiler Programmer’s Guide for details.

SUMMARY

The SAS system option settings you choose can substantially affect the performance of your SAS applications. Furthermore, an understanding of those options is necessary for you to make the choices and tradeoffs appropriate for your installation.

SAS, SAS/GRAPH, SAS/FSP, and SAS/AF are registered trademarks of SAS Institute Inc. Cary, NC. USA.

SyncSort is a trademark of SYNCSORT INC. Englewood Cliffs, NJ. USA.