CHAPTER

6

Performance Considerations

Hardware Considerations

To run Version 8 of the SAS System under Windows, your PC must contain a processor equivalent to the Intel 80486DX or higher.

There are several other hardware factors that might affect performance. Not all of them will apply to your particular configuration or to the way in which you use the SAS System. Before proceeding with these recommendations, investigate their merits with your system administrator.

internal memory
In general, the more memory that is available to Windows, the less swapping to disk Windows needs to do. Note that if you already have sufficient memory to load the tasks you perform, then increasing the amount of memory might not increase performance.

The minimum amount of memory to run the SAS System is 16 MB, but SAS performs best when a workstation has at least 24 MB. SAS servers should have 256 MB or more.

disk caching
Hard disk drive controllers that use their own RAM to cache data generally have better throughput than conventional controllers. As a result, both Windows and the SAS System spend less time on I/O.

hard drive space and speed
In general, large hard drives also have fast access times and fast data transfer rates. Since the SAS System is heavily I/O oriented, all of these factors (size, access time, and transfer rate) are important to system performance.

SCSI and Enhanced IDE drives generally have faster access times than MFM or IDE drives. Local bus hard drive controllers (for use with local bus motherboards) move data more efficiently than conventional controllers.

video card
PCI bus video cards (used with PCI bus motherboards) can increase the speed at which graphics are rendered, thus allowing the SAS system to continue processing data.

Windows Features that Optimize Performance

Under Windows NT

If you are using SAS under Windows NT, you can control the relative responsiveness of your SAS session by altering the application performance level. This can be done by

1. selecting

   START ▶ Settings ▶ Control Panel

   which displays Control Panel window.
2. clicking on the **System** icon within the Control Panel window. This displays the System Properties window.
3. selecting the Performance tab within the System Properties window.
4. drag the **Boost** arrow to alter the performance response time for the foreground application.

To optimize the Windows NT performance, alter the application performance boost using these guidelines:

- If you are running the SAS windowing environment and want your interactive session to have the best response time, set the performance boost to **maximum**.
- If you are running the SAS System in batch mode and you want your batch jobs to execute more quickly, set the performance boost half way between **None** and **Maximum**.

To analyze the performance of your SAS applications, you can specify SAS performance counters within the NT Performance Monitor. For more on using NT Performance Monitor, see “Overview” on page 177.

If your PC has multiple processors, SAS takes advantage of symmetric multiprocessing (SMP) with Version 8 I/O enhancements. Larger amounts of read-ahead processing is done for procedures that have large amounts of sequential data access on data stored on Windows NT Server. This occurs more on system with extra processing power to service NT and it’s disk cache.

The following is generally true in multi-processing SMP environments:

- Machines used as servers for multiple applications are best if SMP-based.
- SAS/CONNECT remote computing environments perform best if SMP-based.
- The supporting hardware in high-end SMP boxes (RAID, RAM) generally help any application perform better.

Under Windows NT Server Enterprise Edition 4.0

The SAS System for Version 8 can take advantage of Intel's Extended Server Memory Architecture (ESMA) to improve the performance of SAS programs that perform large amounts of I/O operations. Using Intel's PSE36 driver, the SAS System can process SAS libraries in the system memory to avoid multiple disk reads and disk writes. The SAS System also can use the additional memory as a SAS file system cache.

To use ESMA memory, you need these system components:

- Intel Pentium II Xeon or Pentium III Xeon processor
- Sufficient system memory. More than 4 GB is recommended, 8 GB is ideal. For systems with less than 4 GB, the boot.ini MAXMEM option reduces the amount of memory claimed by Windows NT at system startup
- Microsoft Windows NT Server Enterprise Edition 4.0 with Service Pack 3 or later
- Intel PSE36 device driver. To see if the PSE36 device driver is installed, contact your system administrator.
- SAS System for Windows, Version 8

Only one SAS session can use the PSE36 device driver as only one application can open the PSE36 device driver at a time. You can monitor the usage of ESMA memory using the NT Performance Monitor. For more information on monitoring ESMA memory usage, see "NT Performance Monitor" on page 181.

Processing SAS Libraries in Extended Server Memory Architecture Memory

SAS libraries that are well suited for processing in ESMA memory have data that are referenced or updated multiple times within a SAS session. Using the WORK library in ESMA memory is beneficial for procedures that write multiple times to large temporary files, such as PROC SORT. To designate the WORK library to use ESMA memory, you start the SAS System using the MEMLIB system option. For more information on the MEMLIB system option, see "MEMLIB" on page 438.

Your SAS program needs to copy the library from disk to memory. After processing the library in ESMA memory, the library must be copied back to disk.

CAUTION:
If you do not copy the library in Extended Server Memory to disk after processing is complete, you will loose any changes made to the library when the SAS session ends. △

You designate a library to be processed in ESMA memory using the MEMLIB option in the LIBNAME statement. All librefs, including a libref to the WORK directory, must have a valid disk directory. For more information on the MEMLIB option, see "LIBNAME" on page 384

The following example shows how to use the LIBNAME statement and the PROC COPY statement to copy a library to and from Extended Server Memory.

```
/* Set up two librefs, one to the library in Extended Server
Memory and the other to the SAS library on disk.  The library on
disk contains dataset1, dataset2, dataset3 and dataset4. */

libname inmemory ''g:\memlib'' memlib;
libname ondisk ''g:\disk'';

/* Copy dataset1, dataset2, dataset3, and dataset4 to Extended
Server Memory   */

proc copy in=ondisk out=inmemory;
```

```
run;

/* ...Assume dataset1 and dataset 4 are updated   */

/* Save the updated datasets back to disk         */

proc copy in=inmemory out=ondisk;
   select dataset1 dataset4;
run;
```

You can also copy a data set to Extended Server Memory using a DATA statement, as shown in the following example:

```
data ondisk.dataset1;
   set inmemory.dataset1;
run;
```

Using Extended Server Memory Architecture as a SAS System File Cache

A SAS file cache is most useful in multiple references of data. For example, a SAS System file cache improves performance in SAS programs containing procedures that make multiple passes of the data. SAS System file caching improves performance in the following situations:

- Repeated reads of a file while other files are being written. Writing to a file clears the Windows NT file system cache.
- Repeated reads of a file when Scatter Gather I/O is active. Scatter Gather I/O operates outside of the Windows NT file system cache. Without the SAS System file cache, there is no data cache and all read operations access the disk. For more information on Scatter Gather I/O, see “SAS System Features that Optimize Performance” on page 153 and “SGIO” on page 461.
- Repeated reads when Windows NT is low on memory. In systems with high memory usage, the performance of the Windows NT file system cache is degraded.

To use Extended Server Memory as a SAS System file cache, specify the MEMCACHE system option when the SAS System is started or in an OPTIONS statement. When you use the MEMCACHE system option in the OPTIONS statement, you can control which data sets use the SAS System file cache, as shown in the following example.

```
/* Example of controlling cached files with the options statement   */

/* Assume cachelib contains 2 data sets, ds1 and ds2.               */
/* Also assume ds1 and ds2 are large enough that they can not exist */
/* in the cache together. ds1 is read many times, so caching is     */
/* desired. ds2 is accessed only once, so cacheing is of no         */
/* benefit.  By using the memcache option, ds1 is cached, and ds2   */
/* not cached.                                                      */

libname cachelib ''e:\tmp'';

/* Turn on full caching */

options memcache = 4;

/* Read ds1 and place the data in the cache.  This read could be a  */
/* more useful read of the file in a real world case.               */
```

```
data _null_;
  set cachelib.ds1;
run;

/* Change memcache setting to use the cache only for files that   */
/* already exist in the cache.                                    */

options memcache = 1;

/* Data from ds1 will come from the cache and ds2 will not be     */
/* cached.                                                        */

proc sort data=cachelib.ds1 out=cachelib.ds2;
  by j;
run;

/* Other access of ds1...                                         */

/* All use of the cache can be ended with a memcache system       */
/* option value of 0.                                             */

options memcache = 0;

/* Neither ds1 or ds2 will access the cache.                      */

proc sort data=cachelib.ds1 out=cachelib.ds2;
  by j;
run;
```

For more information on the MEMCACHE system option, see "MEMCACHE" on page 437.

Under Windows

Ensure that Windows is configured for optimal performance:

1. With the *right* mouse button, click on the **My Computer** icon on the desktop, and select **Properties...**.
2. Select the **Performance** tab. The performance status should report that your system is configured for optimal performance. If it is not, use the Advanced Settings to adjust the configuration accordingly.

SAS System Features that Optimize Performance

The following are some additional features of the SAS System that you can control to improve system performance and make efficient use of your computer's resources. For additional information about optimizing SAS performance, see the chapter on optimizing system performance in *SAS Language Reference: Concepts*.

- Create SAS data sets instead of accessing flat ASCII files. The SAS System can access a SAS data set more efficiently than it can read flat files.

  Also, you should convert existing data sets that you use frequently to Version 8 format.

- In your SAS code, use IF-THEN-ELSE conditional structures instead of multiple IF-THEN structures. When one condition in the IF-THEN-ELSE structure is met, control returns to the top of the structure (skipping the ELSE clause, which might contain subsequent IF-THEN structures). With multiple IF-THEN structures, each condition must be checked.
- When using arrays, make them _TEMPORARY_ if possible. This speeds retrieval time.
- Use programming structures that reduce file I/O, the most time-intensive aspect of SAS processing. Some ideas for reducing file I/O are:
  - Use the WHERE statement in a procedure to reduce extra data processing.
  - Use indexed data sets to speed access to the desired observations.
  - Use the SQL procedure to subset and group your data.
- Experiment with the value of the CATCACHE system option, which specifies the number of SAS catalogs to keep open at one time. By default, no catalogs are cached in memory (and CATCACHE is set to 0). Caching catalogs is an advantage if one SAS application uses catalogs that are subsequently needed by another SAS application. The second SAS application can access the cached catalog more efficiently.

  Note: Storing catalogs in memory can consume considerable resources. Use this technique only if memory issues are not a concern. △
- Store your data sets in a compressed format (using the COMPRESS data set option). This can improve the performance of your SAS application, though it might require more CPU time to decompress observations as SAS needs them. The COMPRESS data set option is described in the data set options section of *SAS Language Reference: Dictionary*.
- You can use the resource file tracking facility to create a scaled-down version of the SAS System that occupies the least amount of hard disk space necessary. For more information, see “Creating a Scaled-Down Version of the SAS System for Distribution” on page 263.
- If you are using Windows NT and you specify the Scatter-read/Gather-write system option, SGIO, the SAS System bypasses intermediate buffer transfers when reading or writing data. The SAS System will read ahead the number of pages specified by the BUFNO system option and place the data in memory before it is needed. When the data is needed it is already in memory, and is in effect a direct memory access. As the value of the BUFNO system option increases, I/O performance improves. Scatter–read / gather–write is active only for SAS I/O opened in INPUT or OUTPUT mode. If any SAS I/O files are opened in UPDATE or RANDOM mode, SGIO is inactive for that process.Compressed and encrypted files can also be read ahead using Scatter-read/Gather-write. For more information on the SGIO system option, see “SGIO” on page 461.

  Note: Windows NT 4 Service Pack 4 is required to use the SGIO system option. △

Network Performance Considerations

Under Windows, loading application DLL (dynamic link library) files from a network drive can result in slower performance than loading the DLL files from a local drive.

To achieve optimum SAS System performance under Windows, run the SAS System from the local disk. Alternatively, you can put the most frequently used modules on the local disk.

Note: For information on monitoring Windows NT performance, see "Overview" on page 177. △

Advanced Performance Tuning Methods

This section presents some advanced performance topics, such as improving the performance of the SORT procedure and calculating data set size. Use these methods only if you are an experienced SAS user and you are familiar with the way SAS is configured on your machine.

Improving Performance of the SORT Procedure

Two options for the PROC SORT statement are available under Windows, the SORTSIZE= and TAGSORT options. These two options control the amount of memory the SORT procedure uses during a sort and are discussed in the next two sections. Also included is a discussion of determining where the sorting process occurs for a given data set and determining how much disk space you need for the sort. For more information about the SORT procedure, see "SORT" on page 367.

SORTSIZE= Option

The PROC SORT statement supports the SORTSIZE= option, which limits the amount of memory available for PROC SORT to use.

If you do not use the SORTSIZE= option in the PROC SORT statement, PROC SORT uses the value of the SORTSIZE= system option. If the system option is not set, PROC SORT uses all available memory and causes unnecessary amounts of swapping. If you use the SORTSIZE= option to limit the amount of available memory to about 1 or 2 megabytes, most of the unneeded SAS files and operating system files are swapped out, and the 1 to 2 megabytes of sort buffers stay in memory for an optimum sort. If PROC SORT needs more memory than you specify, it creates a temporary utility file in your SASWORK directory to complete the sort.

The default value of this option is 2 megabytes (MB), which is optimal. If your machine has more than 12 MB of physical memory and you are sorting large data sets, setting this option to a value between 4 MB and 8 MB may improve performance.

Note: You can also use the SORTSIZE system option, which has the same effect as the SORTSIZE= option in the PROC SORT statement. △

TAGSORT Option

The TAGSORT option is useful in situations where there may not be enough disk space to sort a large SAS data set. When you specify the TAGSORT option, only sort keys (that is, the variables specified in the BY statement) and the observation number for each observation are stored in the temporary files. The sort keys, together with the observation number, are referred to as tags. At the completion of the sorting process, the tags are used to retrieve the records from the input data set in sorted order. Thus, in cases where the total number of bytes of the sort keys is small compared with the length of the record, temporary disk use is reduced considerably. However, you should have enough disk space to hold another copy of the data (the output data set) or two copies of the tags, whichever is greater. Note that although using the TAGSORT option can reduce temporary disk use, the processing time may be much higher.

Determining Where the Sort Occurs

Where the physical sort occurs for a given data set depends on how you reference the data set name and whether you use the OUT= option in the PROC SORT statement. You may want to know where the sort occurs if you think there may not be enough disk space available for the sort. You always need free disk space that equals 3 to 4 times the SAS data set size. For example, if your SAS data set takes up 1MB of disk space, you need 3 to 4MB of disk space to complete the sort.

When you sort a SAS data set, a temporary utility file is opened in the WORK data library (that is, in a subdirectory of the SASWORK directory) if there is not enough memory to hold the data set during the sort. This file has a .sasv7butl file extension. This file can be several times as large as your data set. Before you sort, be sure your WORK data library has room for this temporary utility file.

Note: If you work with especially large data sets, and you use a Windows NT NTFS disk volume, you should redirect your WORK data library to that volume. Windows NT with NTFS is not restricted by the 2 gigabyte file size limit you might encounter under other Windows systems. For more information, see "Using Large Data Sets with Windows NT and NTFS" on page 82. △

A second file with a .SU7 file extension is also created, which, if the sort completes successfully, is renamed to the data set name of the file being sorted (with a .sasv7bdat file extension). The original data set is then deleted. This technique ensures data integrity. Be sure that you have space for this .SU7 file. Use the following rules to determine where the .SU7 file and the resulting sorted data set are created:

- If you omit the OUT= option in the PROC SORT statement, the data set is sorted on the drive and in the directory or subdirectory where it is located. For example, if you submit the following statements (note the two-level data set name), the .SU7 file is created on the C: drive in the MYDATA subdirectory:

```
libname mylib 'c:\sas\mydata';
proc sort data=mylib.report;
   by name;
run;
```

  Similarly, if you specify a one-level data set name, the .SU7 file is created in your WORK data library.

- If you use the OUT= option in the PROC SORT statement, the .SU7 file is created in the directory associated with the libref used in the OUT= option. If you use a one-level name (that is, no libref), the .SU7 file is created in the WORK data library. For example, in the following SAS program, the first sort occurs in the SASWORK subdirectory, while the second occurs on the F: drive in the JANDATA directory:

```
proc sort data=report out=newrpt;
   by name;
run;
libname january 'f:\jandata';
proc sort data=report out=january.newrpt;
   by name;
run;
```

Calculating Data Set Size

To estimate the amount of disk space needed for a SAS data set:

1 create a dummy SAS data set containing one observation and the variables you need

2 run the CONTENTS procedure using the dummy data set
3 determine the data set size by performing simple math using information from the CONTENTS procedure output

For example, for a data set with one character variable and four numeric variables, you would submit the following statements:

```
data oranges;
  input variety $ flavor texture looks;
  total=flavor+texture+looks;
  datalines;
navel 9 8 6
;
proc contents data=oranges;
  title 'Example for Calculating Data Set Size';
run;
```

These statements generate the output shown in Output 6.1 on page 157.

Output 6.1 Example for Calculating Data Set Size with PROC CONTENTS

```
                  Example for Calculating Data Set Size                          1
                                  07:44 Tuesday, February 2, 1999

                       The CONTENTS Procedure

Data Set Name: WORK.ORANGES                       Observations:         1
Member Type:   DATA                               Variables:            5
Engine:        V8                                 Indexes:              0
Created:       7:45 Tuesday February 2, 1999      Observation Length:   40
Last Modified: 7:45 Tuesday February 2, 1999      Deleted Observations: 0
Protection:                                       Compressed:           NO
Data Set Type:                                    Sorted:               NO
Label:

               -----Engine/Host Dependent Information-----

      Data Set Page Size:         4096
      Number of Data Set Pages:   1
      First Data Page:            1
      Max Obs per Page:           101
      Obs in First Data Page:     1
      Number of Data Set Repairs: 0
      File Name:                  C:\TEMP\SAS Temporary Files\_Td200\oranges.sas7bdat
      Release Created:            8.00.008
      Host Created:               WIN_NT


            -----Alphabetic List of Variables and Attributes-----

                #    Variable    Type    Len    Pos
                $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
                2    flavor      Num       8      0
                4    looks       Num       8     16
                3    texture     Num       8      8
                5    total       Num       8     24
                1    variety     Char      8     32
```

The size of the resulting data set depends on the data set page size and the number of observations. The following formula can be used to estimate the data set size:

number of data pages = 1 + (floor(number of obs / **`Max Obs per Page`**))

size = 256 + (**`Data Set Page Size`** * number of data pages)

(*floor* represents a function that rounds the value down to the nearest integer.)

Taking the information shown in Output 6.1 on page 157, you can calculate the size of the example data set:

number of data pages = 1 + (floor(1/101))

size = 256 + (4096 * 1) = 4352

Thus, the example data set uses 4,352 bytes of storage space.

Increasing the Efficiency of Interactive Processing

If you are running a SAS job using the SAS System interactively and the job generates numerous log messages or extensive output, consider using the AUTOSCROLL command to suppress the scrolling of windows. This makes your job run faster because the SAS System does not have to use resources to update the display of the LOG and OUTPUT windows during the job. For example, issuing **`autoscroll 0`** in the LOG window causes the LOG window not to scroll until your job is finished. (For the OUTPUT window, AUTOSCROLL is set to 0 by default.)

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS Companion for the Microsoft Windows Environment, Version 8,* Cary, NC: SAS Institute Inc., 1999. pp.555.

SAS Companion for the Microsoft Windows Environment, Version 8

ISBN 1–58025–524–8

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

1st printing, September 1999

The Institute is a private company devoted to the support and further development of its software and related services.