C H A P T E R

6

Performance Considerations

Hardware Considerations

To run Version 8 of the SAS System under OS/2, your PC must contain a processor that is 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. Consult your system administrator for details.

internal memory
    In general, the more memory that is available to OS/2, the less swapping to disk OS/2 needs to do. Note that if you already have sufficient memory to load the tasks that 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 OS/2 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.

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.

□ 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 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 is 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 copy of the SAS System that occupies the least amount of hard disk space necessary. For more information, see "Creating a Scaled-Down Configuration of the SAS System for Distribution" on page 189.

Network Performance Considerations

Under OS/2, 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 OS/2, run the SAS System from the local disk. Alternatively, you can put the most frequently used modules on the local disk.

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 that SAS is configured on your machine.

Improving Performance of the SORT Procedure

Consider using two options for the PROC SORT statement, the SORTSIZE= and TAGSORT options. These two options control the amount of memory that the SORT procedure uses during a sort and are discussed in the next two sections. For more information about the SORT procedure, see "SORT" on page 285.

SORTSIZE= Option

The SORTSIZE= option specifies the amount of memory that is available for PROC SORT to use and can reduce the amount of swapping that the SAS System must do to sort the data set. A value of 2 M is optimal for all memory configurations. 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 2 MB and 8 MB may improve performance. If PROC SORT needs more memory than you specify, it creates a temporary utility file in your WORK folder to complete the sort.

If you do not use the SORTSIZE= option, PROC SORT uses the value of the SORTSIZE system option. The default value of the SORTSIZE system option is 2 M.

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 that are 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

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

When you sort a SAS data set, the data set is sorted in a temporary utility file that has a file extension of .sasv7butl or .su7 (the shorter file extension is used in libraries that support only three characters in a file extension). If there is not enough memory to hold the utility file during the sort, the utility file is created in the WORK data library. If the sort completes successfully, the utility file is renamed with an extension of .sasv7bdat and the original data set is deleted. This ensures data integrity. Be sure that you have space for the utility file. Use the following rules to determine where the utility file and the resulting sorted data sets are created:

☐ If you omit the OUT= option in the PROC SORT statement, the data set is sorted in the folder where it is located. For example, if you submit the following statements, the .SU7 file is created on the C: drive in the MYDATA folder. Note the two-level data set name.

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

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

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

```
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 that is needed for a SAS data set:

1 create a dummy SAS data set that contains one observation and the variables that 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 125.

Output 6.1   Example for Calculating Data Set Size with PROC CONTENTS

```
                    Example for Calculating Data Set Size                        1

                       The CONTENTS Procedure

Data Set Name: WORK.ORANGES                  Observations:       1
Member Type:   DATA                          Variables:          5
Engine:        V8                            Indexes:            0
Created:       x:xx Tuesday February 2, xxxx  Observation Length: 40
Last Modified: x:xx Tuesday February 2, xxxx  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:                 E:\SASV8\SASWORK\_TD301\oranges.sas7bdat
     Release Created:           8.00.00B
     Host Created:              OS2


         -----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. You can use your PROC CONTENTS output and the following formula to estimate the data set size:

number of data pages = 1 + (*floor*(**Observations** / **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 that is shown in Output 6.1 on page 125, 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 by 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 the command `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.)

SAS® Companion for the OS/2® Environment, Version 8

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