CHAPTER
19

Optimizing System Performance

Definitions

performance statistics
are measurements of the total input and output operations (I/O), memory, and CPU time used to process individual DATA or PROC steps. You can obtain these statistics by using SAS system options that can help you measure your job's initial performance and to determine how to improve performance.

system performance
is measured by the overall amount of I/O, memory, and CPU time that your system uses to process SAS programs. By using the techniques discussed in the following sections, you can reduce or reallocate your usage of these three critical resources to improve system performance. While you may not be able to take advantage of every technique for every situation, you can choose the ones that are best suited for a particular situtation.

Collecting and Interpreting Performance Statistics

Using the FULLSTIMER and STIMER System Options

The FULLSTIMER and STIMER system options control the printing of performance statistics in the SAS log. These options produce different results, depending on your operating environment. See the SAS documentation for your operating environment for details about the output that SAS generates for these options.

The following output shows an example of the FULLSTIMER output in the SAS log, as produced in a UNIX operating environment.

Output 19.1 Sample Results of Using the FULLSTIMER Option in a UNIX Operating Environment

```
NOTE: DATA statement used:
      real time           0.19 seconds
      user cpu time       0.06 seconds
      system cpu time     0.01 seconds
      Memory                           460k
      Semaphores    exclusive 194 shared 9 contended 0
      SAS Task context switches         1   splits 0
```

The STIMER option reports a subset of the FULLSTIMER statistics. The following output shows an example of the STIMER output in the SAS log in a UNIX operating environment.

Output 19.2 Sample Results of Using the STIMER Option in a UNIX Operating Environment

```
NOTE: DATA statement used:
      real time           1.16 seconds
      cpu time            0.09 seconds
```

Operating Environment Information: See the documentation for your operating environment for information about how STIMER differs from FULLSTIMER in your operating environment. The information that these options display varies depending on your operating environment, so statistics that you see might differ from the ones shown. △

Interpreting FULLSTIMER and STIMER Statistics

Several types of resource usage statistics are reported by the STIMER and FULLSTIMER options, including real time (elapsed time) and CPU time. *Real time* represents the clock time it took to execute a job or step; it is heavily dependent on the capacity of the system and the current load. As more users share a particular resource, less of that resource is available to you. *CPU time* represents the actual processing time required by the CPU to execute the job, exclusive of capacity and load factors. If you must wait longer for a resource, your CPU time will not increase, but your real time will increase. It is not advisable to use real time as the only criterion for the efficiency of your program because you cannot always control the capacity and load demands on your system. A more accurate assessment of system performance is CPU time, which decreases more predictably as you modify your program to become more efficient.

The statistics reported by FULLSTIMER relate to the three critical computer resources: I/O, memory, and CPU time. Under many circumstances, reducing the use of any of these three resources usually results in better throughput of a particular job and a reduction of real time used. However, there are exceptions, as described in the following sections.

Techniques for Optimizing I/O

Overview of Techniques for Optimizing I/O

I/O is one of the most important factors for optimizing performance. Most SAS jobs consist of repeated cycles of reading a particular set of data to perform various data analysis and data manipulation tasks. To improve the performance of a SAS job, you must reduce the number of times SAS accesses disk or tape devices.

To do this, you can modify your SAS programs to process only the necessary variables and observations by:

- using WHERE processing
- using DROP and KEEP statements
- using LENGTH statements
- using the OBS= and FIRSTOBS= data set options.

You can also modify your programs to reduce the number of times it processes the data internally by:

- creating SAS data sets
- using indexes
- accessing data through views
- using engines efficiently.

You can reduce the number of data accesses by processing more data each time a device is accessed by setting the BUFNO=, BUFSIZE=, CATCACHE=, and COMPRESS= system options.

Sometimes you might be able to use more than one method, making your SAS job even more efficient.

Using WHERE-Expression Processing

You might be able to use a WHERE statement in a procedure to perform the same task as a DATA step with a subsetting IF statement. The WHERE statement can eliminate extra DATA step processing when performing certain analyses because unneeded observations are not processed.

For example, the following DATA step creates a data set SEATBELT, which contains only those observations from the AUTO.SURVEY data set for which the value of SEATBELT is YES. The new data set is then printed.

```
libname auto '/users/autodata';
data seatbelt;
   set auto.survey;
   if seatbelt='yes';
run;

proc print data=seatbelt;
run;
```

However, you can get the same output from the PROC PRINT step without creating a data set if you use a WHERE statement in the PROC PRINT step, as in the following example:

```
proc print data=auto.survey;
   where seatbelt='yes';
run;
```

The WHERE statement can save resources by eliminating the number of times you process the data. In this example, you might be able to use less time and memory by eliminating the DATA step. Also, you use less I/O because there is no intermediate data set. Note that you cannot use a WHERE statement in a DATA step that reads raw data.

The extent of savings that you can achieve depends on many factors, including the size of the data set. It is recommended that you test your programs to determine which is the most efficient solution. See "Deciding Whether to Use a WHERE Expression or a Subsetting IF Statement" on page 241 for more information.

Using DROP and KEEP Statements

Another way to improve efficiency is to use DROP and KEEP statements to reduce the size of your observations. When you create a temporary data set and include only the variables that you need, you can reduce the number of I/O operations that are required to process the data. See *SAS Language Reference: Dictionary* for more information on the DROP and KEEP statements.

Using LENGTH Statements

You can also use LENGTH statements to reduce the size of your observations. When you include only the necessary storage space for each variable, you can reduce the number of I/O operations that are required to process the data. Before you change the length of a numeric variable, however, see "Specifying Variable Lengths" on page 250. See *SAS Language Reference: Dictionary* for more information on the LENGTH statement.

Using the OBS= and FIRSTOBS= Data Set Options

You can also use the OBS= and FIRSTOBS= options to reduce the number of observations processed. When you create a temporary data set and include only the necessary observations, you can reduce the number of I/O operations that are required to process the data. See *SAS Language Reference: Dictionary* for more information on the OBS= and FIRSTOBS= data set options.

Creating SAS Data Sets

If you process the same raw data repeatedly, it is usually more efficient to create a SAS data set. SAS can process SAS data sets more efficiently than it can process raw data files.

Another consideration involves whether you are using data sets created with previous releases of SAS. If you frequently process data sets created with previous releases, it is sometimes more efficient to convert that data set to a new one by creating it in the most recent version of SAS. See Chapter 34, “Compatibility of Version 8 with Earlier Releases,” on page 493 for more information.

Using Indexes

An index is an auxiliary data structure that is used in conjunction with WHERE-expression processing, BY-group processing, or a MODIFY or SET statement with the KEY= option to locate and select specific observations by the value of the indexed variable. By creating and using an index, you can access an observation faster. Without the index, SAS must start at the top of the data set and read the observations sequentially to the end of the data set, applying the WHERE clause or BY statement to each observation. In contrast, an index returns observations in sorted order.

Note: Indexing might or might not, however, improve the performance of an application. If you are continually rewriting a data set, indexing its variables would be wasteful because an index must be recreated each time the data set is rewritten. △

See Chapter 28, “SAS Data Files,” on page 411 for more information about indexes.

Accessing Data Through Views

You can use the SQL procedure or a DATA step to create views of your data. A view is a stored set of instructions that subsets your data with fewer statements. Also, you can use a view to group data from several data sets without creating a new one, saving both processing time and disk space. See Chapter 29, “SAS Data Views,” on page 455 and the *SAS Procedures Guide* for more details.

Using Engines Efficiently

If you do not specify an engine on a LIBNAME statement, SAS must perform extra processing steps to determine which engine to associate with the data library. SAS must look at all of the files in the directory until it has enough information to determine which engine to use. For example, the following statement is efficient because it explicitly tells SAS to use a specific engine with the libref FRUITS:

```
/* Engine specified. */

libname fruits v8 '/users/myid/mydir';
```

The following statement does not explicitly specify the V8 engine; notice the NOTE about mixed engine types that is generated:

```
/* Engine not specified. */

libname fruits '/users/myid/mydir';

NOTE: Directory for library FRUITS contains
      files of mixed engine types.
NOTE: Libref FRUITS was successfully assigned
      as follows:
      Engine:        V8
      Physical Name: /users/myid/mydir
```

Operating Environment Information: In the OS/390 operating environment, you do not need to specify an engine for certain types of libraries. △

See Chapter 36, "SAS I/O Engines," on page 511 for more information about SAS engines.

Setting the BUFNO=, BUFSIZE=, CATCACHE=, and COMPRESS= System Options

The following SAS system options can help you reduce the number of disk accesses that are needed for SAS files, though they might increase memory usage.

BUFNO=

SAS uses the BUFNO= option to adjust the number of open page buffers when it processes a SAS data set. Increasing this option's value can improve your application's performance by allowing SAS to read more data with fewer passes; however, your memory usage increases. Experiment with different values for this option to determine the optimal value for your needs.

Note: You can also use the CBUFNO= system option to control the number of extra page buffers to allocate for each open SAS catalog. △

See "System Options" in *SAS Language Reference: Dictionary* and the SAS documentation for your operating environment for more details on this option.

BUFSIZE=

When the V8 engine creates a data set, it uses the BUFSIZE= buffer size option to determine the page size of the data set. In each subsequent I/O operation, SAS moves the number of pages that is set by the BUFNO= option. Whether you use your operating environment's default value or specify a value, the engine always writes complete pages regardless of how full or empty those pages are.

If you know that the total amount of data is going to be small, you can enforce a small page size with the BUFSIZE= option, so that the total data set size remains small and you minimize the amount of wasted space on a page. In contrast, if you know that you are going to have many observations in a data set, you should optimize BUFSIZE= so that as little overhead as possible is needed. Note that each page requires some additional overhead.

Large data sets that are accessed sequentially benefit from larger page sizes because sequential access reduces the number of system calls that are required to read the data set. Note that because observations cannot span pages, typically there is unused space on a page.

"Calculating Data Set Size" on page 250 discusses how to estimate data set size.

See "System Options" in *SAS Language Reference: Dictionary* and the SAS documentation for your operating environment for more details on this option.

CATCACHE=

SAS uses this option to determine the number of SAS catalogs to keep open at one time. Increasing its value can use more memory, although this may be warranted if your application uses catalogs that will be needed relatively soon by other applications. (The catalogs closed by the first application are cached and can be accessed more efficiently by subsequent applications.)

See "System Options" in *SAS Language Reference: Dictionary* and the SAS documentation for your operating environment for more details on this option.

COMPRESS=

One further technique that can reduce I/O processing is to store your data as compressed data sets by using the COMPRESS= data set option. However, storing your data this way means that more CPU time is needed to decompress the observations as they are made available to SAS. But if your concern is I/O, and not CPU usage, compressing your data may improve the I/O performance of your application.

See *SAS Language Reference: Dictionary* for more details on this option.

Techniques for Optimizing Memory Usage

If memory is a critical resource, several techniques can reduce your dependence on increased memory. However, most of them also increase I/O processing or CPU usage.

However, by *increasing* memory available to SAS by increasing the value of the MEMSIZE= system option (or by using the MEMLEAVE= option, in some operating environments), you can decrease processing time because the amount of time that is spent on paging, or reading pages of data into memory, is reduced. The SORTSIZE= and SUMSIZE= system options enable you to limit the amount of memory that is available to sorting and summarization procedures.

You can also make tradeoffs between memory and other resources, as discussed in "Reducing CPU Time by Using More Memory or Reducing I/O" on page 249. To make the most of the I/O subsystem, you must use more and larger buffers. However, these buffers share space with the other memory demands of your SAS session.

Operating Environment Information: The MEMSIZE= system option is not available in some operating environments. If MEMSIZE= is available in your operating environment, it might not increase memory. See the documentation for your operating environment for more information. △

Techniques for Optimizing CPU Performance

Reducing CPU Time by Using More Memory or Reducing I/O

Executing a single stream of code takes approximately the same amount of CPU time each time that code is executed. Optimizing CPU performance in these instances is usually a tradeoff. For example, you might be able to reduce CPU time by using more memory, because more information can be read and stored in one operation, but less memory is available to other processes.

Also, because the CPU performs all the processing that is needed to perform an I/O operation, an option or technique that reduces the number of I/O operations can also have a positive effect on CPU usage.

Storing a Compiled Program for Computation-Intensive DATA Steps

Another technique that can improve CPU performance is to store a DATA step that is executed repeatedly as a compiled program rather than as SAS statements. This is especially true for large DATA step jobs that are not I/O-intensive. For more information on storing compiled DATA steps, see Chapter 30, "Creating and Executing Stored Compiled DATA Step Programs," on page 465.

Reducing Search Time for SAS Executable Files

The PATH= system option specifies the list of directories (or libraries, in some operating environments) that contain SAS executable files. Your default configuration file specifies a certain order for these directories. You can rearrange the directory specifications in the PATH= option so that the most commonly accessed directories are listed first. Place the least commonly accessed directories last.

Operating Environment Information: The PATH= system option is not available in some operating environments. See the documentation for your operating environment for more information. △

Specifying Variable Lengths

When SAS processes the program data vector, it typically moves the data in one large operation rather than by individual variables. When data is properly aligned (in 8-byte boundaries), data movement can occur in as little as 2 clock cycles (a single load followed by a single store). SAS moves unaligned data by more complex means, at worst, a single byte at a time. This would be at least eight times slower for an 8-byte variable.

Many high performance RISC (Reduced Instruction Set Computer) processors pay a very large performance penalty for movement of unaligned data. When possible, leave numeric data at full width (eight bytes). Note that SAS must widen short numeric data for any arithmetic operation. On the other hand, short numeric data can save both memory and I/O. You must determine which method is most advantageous for your operating environment and situtation.

Note: Alignment can be especially important when you process a data set by selecting only specific variables or when you use WHERE-expression processing. △

Calculating Data Set Size

If you have already applied optimization techniques but still experience lengthy processing times or excessive memory usage, the size of your data sets might be very large, in which case, further improvement might not be possible.

You can estimate the size of a data set by creating a dummy data set that contains the same variables as your data set. Run the CONTENTS procedure, which shows the size of each observation. Multiply the size by the number of observations in your data set to obtain the total number of bytes that must be processed. You can compare processing statistics with smaller data sets to determine if the performance of the large data sets is in proportion to their size. If not, further optimization might still be possible.

Note: When you use this technique to calculate the size of a data set, you obtain only an estimate. Internal requirements, such as the storage of variable names, might cause the actual data set size to be slightly different. △

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS Language Reference: Concepts*, Cary, NC: SAS Institute Inc., 1999. 554 pages.

SAS Language Reference: Concepts

Copyright © 1999 SAS Institute Inc., Cary, NC, USA.

ISBN 1–58025–441–1

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

1st printing, November 1999