

SAS Programming for Efficient Processing

Clarence W. Jackson
City of Dallas Revenue and Taxation Department

This paper will explore known efficiency tips in SAS programming as defined in "SAS Programming Tips: A Guide to Efficient SAS Processing", and also outline those tips that are efficient in everyday coding. Attention will be given to the automatic background processing of SAS code, and the use of SAS programming statements to make for efficiency.

HOW IS EFFICIENCY DEFINED?

Resources that are available for SAS programs vary from site to site, but one thing is for sure, all resources are limited, some more than others. Efficiency defined, is "to get the same or more results from fewer resources". In the case of SAS programming, those resources are computer, and human. Computer efficiency seeks to reduce CPU time, input/output processes, data storage. Human efficiency seeks to reduce programming time, which includes designing, coding, testing, upkeep, and documenting SAS programs.

HOW DO YOU DECIDE WHAT IS IMPORTANT FOR EFFICIENCY?

What is important for your site depends on many factors. Availability of resources and cost are two very important factors, but not the only ones to consider. To make the efficiency decision, you must take four steps:

- * Understand efficiency at your site
(consider the hardware, host operating system, and SAS environment)
- * Know how your program will be used
(consider size of file and program, times and frequency of use)
- * Know the type of data your program will process
- * Understand the effects of tradeoffs in resources used

HOW TO VERIFY EFFICIENCY?

To verify your program's efficiency, collect:

- * performance data for resources being used
(CPU time, I/O, memory, etc)
- * SAS dataset information (use PROCs CONTENTS and DATASETS)
- * Interpret the results of the collected data

Try using different efficiency techniques and verify them by:

- * benchmarking your program
- * Account for ALL factors
(system type, number of records, etc)

As you verify your program, remember to:

- * control your code
- * control your environment
- * use multiple runs at similar and different times
- * use single-purpose runs (if possible)
- * consider the effect of overhead
- * explore other performance statistics
- * make prudent conclusions

HOW TO MAKE SAS PROGRAMS BE MORE EFFICIENT?

There are 7 general principles that are published in the book "SAS Programming Tips: A Guide to Efficient SAS Processing". The tips in the book were tested by SAS Institute's Publications Division by submitting five separate SAS jobs using SAS data sets with 1000 observations or raw data files with 1000 records. I have added one for processing of large and/or multiple files, labeled '0'. The 7 + 1 principles are:

1. Read/Write data selectively.
2. Execute only the statements needed in the order needed.
3. Use SAS PROC's wisely.
4. Know SAS defaults, and how to change them to your advantage.
5. Control sorting
6. Know your data, and test your programs.
7. Code clearly, and document your programs.
0. Use filename/libname statements instead of 'DD' for file definition, then use "filename ... clear" when processing the file is completed.

1. READ AND WRITE DATA SELECTIVELY

Reading and writing data (input/output or I/O) is the largest single component of elapsed time in most programs. Decreasing the number of I/O operations is the single most effective way to improve the execution speed of your programs. Most methods for reducing the number of I/O's fall into one of the following categories:

- * reading/writing based on a condition, rather than reading/writing all the time
- * simplifying the information you read/write, or reducing it in size
- * substituting execution time transformations for large reads/writes
- * eliminating redundant read/write steps with better logic
- * making use of SAS features and tools

In other words, if you don't need to read/write, don't!

Techniques that apply to this principle are:

- * read only the fields you need
- * read selection fields first
- * store data in SAS data sets
- * keep summaries of large SAS data sets
- * store only the variables you need
- * process only the variables you need
- * create all data subsets at one time
- * use informats for data transformations
- * shorten data using formats and informats
- * edit external files directly
- * use one buffer for external file operations
- * create indexes when appropriate
- * use binary search, not null merge

2. EXECUTE ONLY THE STATEMENTS YOU NEED, IN THE ORDER YOU NEED THEM

The number and complexity of statements executed largely control the CPU time used. By default, the SAS System executes every statement in the DATA step for each observation in the input source. Within a statement, it executes every operation in a given expression each time the statement executes. There are methods to:

- * reduce the number of statements executed
(LINK, IF-THEN-ELSE, SELETE-WHERE, DO-WHILE/UNTIL-END, etc)
- * reduce the number of operations performed

Some techniques that can be used are:

- * assign a value to a constant only once
- * use constants in expressions
- * condense constants in expressions
- * use mutually exclusive conditions
- * write conditions in order of descending probability
- * put missing values last in expressions
- * check for missing values before using a variable in multiple statements
- * put only statements affected by the loop in a loop
- * assign many values in one statement
- * shorten expressions with functions
- * edit character values with functions
- * use the IN rather than logical OR operators
- * use a series of conditions
- * check for undesirable conditions and stop processing
- * write the loop with the fewest iterations outermost
- * use temporary arrays rather than creating and dropping variables
- * use macros for repeated code
- * create macro variables only when needed
- * make array variables either all retained or all unretained
- * set the lower bound of arrays to 0
- * use multidimensional explicit arrays
- * use the Stored Program Facility

3. TAKE ADVANTAGE OF SAS PROCEDURES

SAS PROCedures are built-in programs that perform a wide variety of specialized tasks. Most of the techniques for taking advantage of PROC's to improve your programming efficiency fall into one of these categories:

- * using procedures instead of code you develop yourself
- * using the right procedure for the right task
- * using a procedure to perform as many different tasks as possible once you have invoked it
- * using the output of procedures for other tasks
- * knowing the full capabilities of any given procedure
- * knowing all of the SAS tools you can use for any given task

4. KNOW SAS SYSTEM DEFAULTS

The design of the SAS System gives you maximum computing power with a minimum of coding. At the same time, you need to understand its default actions so you can override them when that suits your current needs and save you CPU time, storage space, and other resources. Some of the techniques you can use are:

- * reduce the storage space for variables
- * shorten SAS date values (use 4 bytes, default is 8 bytes)
- * Know the rules for creating null data sets
- * avoid default type conversions
- * use character rather than numeric variables
- * create separate variables instead of repeating type conversions
- * use SAS configuration and autoexec files
- * compress large SAS data sets
- * Eliminate the macro facility if your program do not need it
- * store numbers a 1 byte character values (1-127 ASCII, 1-255 EBCDIC
example: oldx=100; ... newx=byte(oldx); ... oldx=rank(newx);)

5. CONTROL SORTING

Sorting is one of the most common operations performed with the SAS System. At the same time, sorting is relatively expensive in terms of CPU time and may require large amounts of work space. Knowing when and how to sort data most efficiently is a simple way to save resources in your SAS programs. You generally need to sort data in at least three cases:

- * when the DATA or PROC step uses a BY statement
- * when a procedure requires sorted data
(without BY-group processing)
- * when you want sorted results for display or printing

If you know whether the current step requires sorted data and the data is already sorted, don't sort. For BY-group processing, the SAS System checks the order of sorted data during execution of the DATA or PROC step. Any method that produces observations in the correct order is acceptable as preparation for BY-group processing. There is no sort flag indicating that the data have been processed by the SORT procedure. In cases where a PROC allows use of the CLASS statement, use CLASS, since the data does not need to be sorted.

6. TEST YOUR PROGRAMS, KNOW YOUR DATA

Programs that fail due to preventable errors waste resources and time. Preventable program failures at critical time can be disastrous. The best time to discover problems in your code or data is during testing. Techniques for finding and correcting problems fall into one of these categories:

- * use appropriate test data
- * test the right amount of code
- * use the correct SAS tools to diagnose and fix problems
- * control your test environment

Things to remember in testing include:

- * use realistic and complete test data
- * develop and test incrementally
- * request necessary messages by using the correct SAS options
- * reconcile SAS log and output
- * diagnose intermediate results
- * examine raw data before reading them
- * document your SAS data sets
- * label your variables and your SAS data sets
- * compile without executing to test syntax
- * protect existing data when you test
- * use collapsing variables to group data

7. CODE CLEARLY

One area of efficient programming that is often overlooked is the amount of time needed in interpreting programs when writing, modifying, or updating your programs. Clearly written programs save time by making it easier to debug, maintenance, modify, and transport programs. Some techniques for making your code clearer are:

- * following conventions or standards consistently
- * using blank space to enhance clarity
- * inserting explanations appropriately
- * simplifying anything confusing

Some tips for clear coding include using comments, meaningful variable names, clearly separated steps, grouped declarative statements, and using straightforward logical conditions. The main point here is to make it simple for the next person that looks at your code (it could be you next year)

8. USE FILENAME/LIBNAME STATEMENTS

In some sites, there are costs associated with files being used and the length of time those files are used. If you use 'DD' or 'ALLOC'ate file definitions, the file cannot be freed until the job or session is completed. Using FILENAME/LIBNAME statements, the file could be allocated to the program when the program needs it, and freed when the program completes use of the file.

FILENAME/LIBNAME statements offer other advantages to 'DD' and 'ALLOC' in reassigning files, portability of code to another operating system. Also, by using macro statements, you can conditionally allocate files using FILENAME/LIBNAME statements. DDnames must either be used or excluded by the programmer modifying the JCL.

SAS Programming for Efficient Processing

CONCLUSION

The SAS System is easy to learn and use. It also does a lot of house housekeeping functions for the programmer that MUST be done by the programmer in other languages. However, there comes a point in which the SAS System can become less efficient for certain tasks. Since SAS does so much housekeeping (some of which may not be needed), to become an efficient SAS programmer, one must understand what SAS is doing with the code. SAS can be a very efficient programming language if understood and used correctly.

As mentioned, the techniques and tips offered in this paper can be found in the book "SAS Programming Tips: A Guide to Efficient SAS Processing" in more detail than presented here.

Author Info:

Clarence Jackson,
Sr. Systems Analyst,
Revenue and Taxation Dept.
City of Dallas, Texas
214-670-5798

References:

SAS Programming Tips: A Guide to Efficient SAS Processing
SAS Language, Reference Version 6, 1st edition
SAS Procedures Guide, Version 6, 3rd edition
SAS Companion for MVS Environment, Version 6, 1st edition
SAS Applications Guide, 1987 edition