Efficiency Techniques for SAS® Software Users

Kirk Paul Lafler, Software Intelligence Corporation

ABSTRACT

As SAS® Software becomes increasingly more popular, guidelines for its efficient use grows greater. This paper presents a collection of techniques for gaining efficiency while using SAS Software. Areas deserving special consideration include program execution, I/O, disk space as well as an often overlooked item known as program maintenance. Sample code is presented to illustrate numerous practical techniques as to why some methods are better than others.

INTRODUCTION

When using SAS Software in the development of program code and/or applications, efficiency isn't always given the attention it deserves, particularly in the early phases of development. Often, system performance requirements can greatly affect the architectural characteristics and/or behavior that an application exhibits. When this is the case, active user participation is crucial to understanding the needs as well as the successful implementation of performance requirements.

Careful attention should be given to each individual program function in order to elaborate on performance criteria characteristics. User expectations, when assessed early (preferably during the early phases of the application development process) help facilitate their measurement once the application is fully operational. Consequently, by adhering to this simple rule, the difficulty associated with trying to improve efficiency as coding nears completion is often minimized.

This paper highlights several areas where a program's performance and efficiency can be improved when using SAS software.

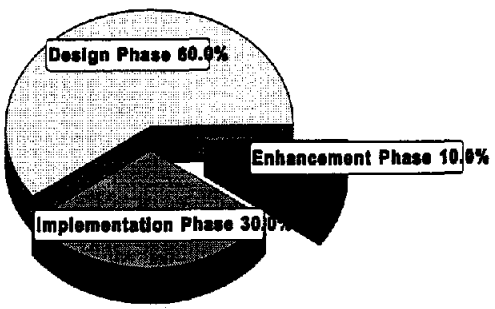
EFFICIENCY OBJECTIVES

Efficiency objectives are best achieved when implemented as early as possible, preferably during the design phase. But when this is not possible, for example when customizing or inheriting an application, efficiency and performance techniques can still be "tacked-on" as an after-thought to obtain the greatest improvements possible. Specific objectives directly achievable through the use of efficiency and performance strategies can be classified into five areas as follows:

① CPU Time
② Data Storage
③ Elapsed Time
④ I/O
⑤ Memory

Jeffrey A. Polzin of SAS Institute Inc. has this to say about measuring efficiency, *"CPU time and elapsed time are baseline measurements, since all the other measurements impact these in one way or another."* He continues by saying, *"... as one measurement is reduced or increased, it influences the others in varying degrees."*

Efficiency and Customization

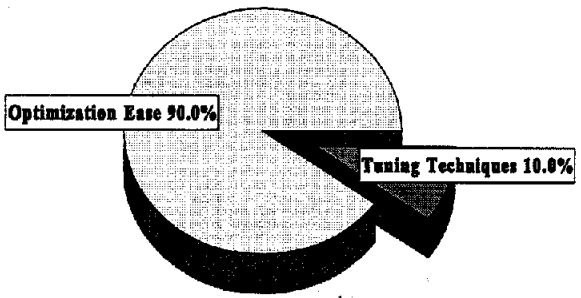


The simplest of requests can fall prey to one or more efficiency violations, such as retaining unwanted datasets in work space or not subsetting to eliminate undesirable records. Much of the efficiency loss can be avoided with better planning prior to beginning the coding process. A well known quip sums it up best, *"most people don't plan to fail - they just fail to plan."* Fortunately, efficiency gains can be realized as long as a few simple guidelines are followed.

GUIDELINES TO HOLD DEAR

The difference between an application that has been optimized versus one that has not is often dramatic. By adhering to practical guidelines, your application code will achieve efficiency in direct relationship to economies of scale. Generally, the first 90% of efficiency improvements can be gained relatively quickly and easily by applying simple strategies. It's often the final 10% that, if pursued, proves to be the challenge. Consequently, you will need to be the judge of whether your application has reached "relative" optimal efficiency while maintaining a virtual balance between time and cost.

Efficiency Scale



The following suggestions are not meant to be an exhaustive review of all known efficiency techniques, but as a sampling of proven methods that can provide some measure of efficiency in SAS programs.

Efficiency techniques are presented for the following resource areas: CPU time, data storage, I/O, memory, and programming time. Program code examples are illustrated in ***Table 1***.

CPU Time

1) Use KEEP or DROP statements to retain desired variables.
2) Create and use indexes with large datasets.
3) Utilize macros for redundant code.
4) Use IF-THEN/ELSE statements to process data.
5) Use the DATASETS procedure COPY statement to copy datasets with indexes.
6) Use the SQL procedure to consolidate the number of steps.
7) Turn off the Macro facility when not needed.
8) Avoid unnecessary sorting - plan its use.
9) Use a CLASS statement in available procedures.
10) Use the Stored Program Facility for complex DATA steps.

Data Storage

1) Use KEEP or DROP statements to retain desired variables.
2) Use LENGTH statements to reduce variable size.
3) Use data compression strategies.
4) Create character variables as much as possible.
5) Use DATA _NULL_ steps for processing null datasets.

I/O

1) Read only data that is needed.
2) Use WHERE statements to subset data.
3) Use data compression for large datasets.
4) Use the DATASETS procedure COPY statement to copy datasets with indexes.
5) Use the SQL procedure to consolidate code.
6) Store data in SAS datasets, not external files.
7) Perform data subsets early and at same time.
8) Use KEEP or DROP statements to retain desired variables.

Memory

1) Read only data that is needed.
2) Use WHERE conditions when possible.
3) Use the DATASETS procedure COPY statement to copy datasets with indexes.

Programming Time

1) Use the SQL procedure for code simplification.
2) Use procedures whenever possible.
3) Document programs and routines with comments.
4) Utilize macros for redundant code.
5) Code for unknown data values.
6) Assign descriptive and meaningful variable names.
7) Store formats and labels with the SAS datasets that use them.
8) Test program code using "complete" test data.

Program Code Examples

The following program examples illustrate the application of a few popular efficiency techniques. Techniques are presented in the areas of CPU time, data storage, I/O, memory, and programming time.

① Using the KEEP statement as a data set option instructs the SAS System to load only the specified variables into the program data vector (PDV), eliminating all other variables from being loaded.

```
data af_users;
  set sands.members
      (keep=name company phone user);
  if user = 'SAS/AF';
run;
```

② The CLASS statement provides the ability to perform by-group processing without the need for data to be sorted first in a separate step. Consequently, CPU time can be saved when data is not already in the desired order. The CLASS statement can be used in the MEANS and SUMMARY procedure.

```
proc means data=mortgage;
  var prin interest;
  class state;
run;
```

③ By using IF-THEN/ELSE statements opposed to IF-THEN statements without the ELSE, the SAS System stops processing the conditional logic once a condition holds true for any observation.

```
data capitols;
  set states;
  if state='CA' then capitol = 'Sacramento';
  else
  if state='FL' then capitol = 'Tallahassee';
  else
  if state='TX' then capitol = 'Austin';
run;
```

④ To avoid using default lengths for variables in a SAS dataset, use the LENGTH statement. Significant space can be saved for numeric variables containing integers since the 8 byte default length is reduced to the specified size. Storage space can be reduced significantly.

④ (Continued)

```
data _null_;
  length pageno rptdate 4;
  set sales;
  file report header=h;
  put @10 item  $20.
      @35 sales  comma6.2;
return;
h:
  rptdate=today();
  pageno + 1;
  put  @20 'Sales Report'
     /  @1 rptdate  mmddyy10.
     / @30 'Page ' pageno 4. //;
return;
run;
```

⑤ To subset data without first running a DATA step use a WHERE statement in a procedure. I/O and memory requirements may be better for it.

```
proc print data=af_users  n  noobs;
  where user = 'SAS/AF';
  title1 'SAS/AF Programmers/Users';
run;
```

⑥ Use the SQL procedure to simplify and consolidate coding requirements. CPU, I/O, and programming time may improve.

```
proc sql;
  title1 'SAS/AF Programmers/Users';
  select *
    from sands.members
      where user = 'SAS/AF'
        order by name;
quit;
```

⑦ To improve data storage and I/O requirements, consider compressing large datasets.

```
data  sands.members (compress = yes);
  < additional statements >
run;
```

Table 1. Program Code Examples

SURVEY RESULTS

A survey was conducted to elicit responses from participants on efficiency and performance. The ***Efficiency and Performance Survey*** is illustrated in ***Table 2***. Analyzing the responses from each participant provided a better appreciation for what users and application developers look for as they apply efficiency methods and strategies.

The purpose for constructing the survey in the first place began in order to assess the general level of understanding that people have with various efficiency methods and techniques. What was found was quite interesting.

The majority of users and application developers want their applications to be as efficient as possible. Many go to great lengths to implement sound efficiency strategies and techniques achieving splendid results. Unfortunately for others, a lack of familiarity with effective techniques often results in a situation where the application works, but may not realize its true potential.

Survey participants often indicated that efficiency and performance tuning is not only important, but essential to their application. Many cite response time as a critical objective and are always looking for ways to improve this benchmark. Charles Edwin Shipp of Shipp Consulting offers these comments on applying efficiency techniques, *"Efficiency shouldn't be considered as a one-time activity. It is best to treat it as a continuing process of reaching an optimal balance between competing resources and activities."*

Other universally accepted findings consist of using WHERE, LENGTH, CLASS, and KEEP/DROP statements; avoiding unnecessary sorting; using SAS functions; and constructing DATA _NULL_ steps as effective techniques to improve the efficiency in an application.

Techniques receiving "strong" (between sometimes and always), but not unanimous, support among survey participants are using system options to control resources; deleting unwanted WORK space datasets; combining (consolidating) steps; using formats and informats; using indexes; using the APPEND procedure to concatenate; constructing IF-THEN/ELSE statements for conditional processing; and saving intermediate files, especially for large multi-step jobs.

Sunil Kumar Gupta of Gupta Programming offers these suggestions on assigning informats, formats, and labels, *"Informats, formats, and labels are stored with many of our important SAS datasets to minimize processing time. An additional reason for using this technique is that many popular procedures use stored formats and labels as they produce output, eliminating the need to assign them in each individual step. This in itself provides added incentives and value for programmers and end-users, especially since reporting requirements are usually time critical."*

Techniques cited by survey participants as "sometimes" being used include using DATA step options; using data compression; conserving on memory by turning off options; using the SQL procedure to consolidate multiple operations; using the Stored Program Facility; creating and using DATA and SQL views; and using the DATASETS procedure COPY statement for indexes.

It was found that using the SQL Pass-Through Facility was virtually "never" used by survey participants.

LEARNING NECESSARY TECHNIQUES

Survey responses revealed the following concerns:

1) An insufficient level of formal training exists in the areas relevant to efficiency and performance.

2) A failure to plan in advance of the coding phase.

3) Insufficient time and inadequate budgets can often be attributed to ineffective planning and implementation of efficiency strategies.

So how do people learn about the various efficiency techniques? A small percentage learn through formal training. Others find published guidelines (e.g., book(s), manuals, articles, etc.) on the subject. But the vast majority learn valuable techniques as a result of a combination of prior experiences, through acquaintances (e.g., User Groups), and/or on the job. This *"sink or swim"* method may, at best, result in a relatively small level of improvement in efficiency, but any improvement is better than no improvement. Consequently, by adhering to a practical set of efficiency guidelines, an application can benefit significantly for many years to come.

EFFICIENCY AND PERFORMANCE SURVEY

Contact: _____ Organization: _____
Telephone: _____ Contact Date: _____

"I am conducting a survey for a regional SAS user group paper that I am writing. The topic of the paper is efficiency and how it relates to the SAS Software. Could you spare a few minutes to answer a few questions on this subject?"

1. Are efficiency and performance issues important in your environment? ❒ Yes ❒ No ❒ Sometimes

2. Have you received any training (formal or informal) in efficiency and performance strategies? ❒ Yes ❒ No

3. Do you take the time to resolve efficiency and performance issues in an application? ❒ Yes ❒ No ❒ Sometimes

4. Rate whether the following efficiency measurement categories have importance in your environment.
(Use the following rating scale: 1=Not Important, 2=Somewhat Important, 3=Very Important.)

   a. _____ CPU Time b. _____ Data Storage c. _____ Elapsed Time d. _____ I/O e. _____ Memory

5. In response to question #4, which measurement has the greatest importance in your environment? _____
Why?: _____

6. At what time(s) during the application development process do you consider using efficiency and performance techniques?

❒	Requirements Definition Phase	❒	Testing Phase
❒	Analysis Phase	❒	Implementation Phase
❒	Design Phase	❒	Maintenance/Enhancement Phase
❒	Coding Phase		

7. Rate the following techniques and/or strategies that you have used in your environment to improve a program's/application's efficiency and/or performance? *(Use the following rating scale: 1=Never, 2=Sometimes, 3=Always.)*

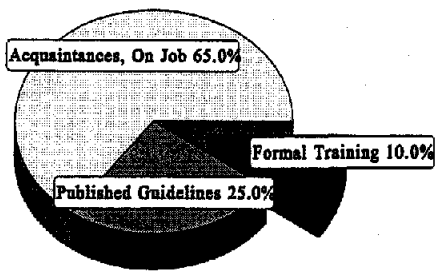
_____ Use System Options such as BUFNO=, BUFOBS=, BUFSIZE= COMPRESS=, etc.
_____ Use DATA Step Options such as NOMISS or NOSTMTID.
_____ Use the LENGTH Statement to reduce the size of numeric variables and storage space.
_____ Use numeric variables for analysis, otherwise create character variables - less CPU intensive.
_____ Use the KEEP or DROP statements to control only variables desired.
_____ Delete Unwanted Datasets in the WORK area.
_____ Combine Steps to minimize the number of DATA and/or PROC steps.
_____ Use Data Compression.
_____ Conserve on Memory (e.g., turning off NOMACRO, array processing)
_____ Use Formats and Informats to save CPU during complex logic assignments.
_____ Avoid unnecessary sorting with PROC SORT.
_____ Control sorting by combining two or more variables at a time when sorting is necessary.
_____ Use Subsetting IF statements to subsetdatasets.
_____ Use WHERE statements to subsetdatasets.
_____ Use indexes to optimize the retrieval of data.
_____ Construct IF-THEN/ELSE statements to process condition(s) with greatest frequency first.
_____ Save intermediate files in multi-step applications.
_____ Use PROC APPEND versus SET statement to concatenatedatasets.
_____ Use PROC SQL to consolidate multiple operations into one step.
_____ Use the PROC SQL Pass-Through Facility to pass logic to target database for processing.
_____ Use the Stored Program Facility to store SAS DATA steps in a compiled format.
_____ Use DATA Views and SQL Views to create "virtual" tables.
_____ Use SAS Functions to perform common tasks.
_____ Use the DATASETS Procedure COPY statement to copydatasets with built-in indexes.
_____ Use the DATA _NULL_ step to avoid creating adataset when one is not needed but processing is.
_____ Use a CLASS statement in procedures that support it to avoid having to sort data.
Other: _____
_____
_____

8. Would you like a copy of the completed paper? ❒ Yes ❒ No

Thank you for participating in this survey!

Table 2. Efficiency and Performance Survey

Where Techniques are Learned



CONCLUSION

The value of implementing efficiency and performance strategies into an application can not be over-emphasized. Careful attention should be given to individual program functions, since one or more efficiency techniques can often affect the architectural characteristics and/or behavior an application exhibits.

Efficiency techniques are learned in a variety of ways. Some learn through formal classroom instruction, while others find published guidelines such as books, manuals, articles, and video tapes to acquire the necessary understanding associated with efficiency. But the vast majority learn through other's experiences, as well as their own, by word-of-mouth, and on the job. Whatever the means, a little efficiency goes along way.

ACKNOWLEDGMENTS

The author would like to thank Sunil Kumar Gupta, Gupta Programming; Charles Edwin Shipp, Shipp Consulting; and James H. Sorenson, Sorenson Systems for participating in the efficiency survey and offering valuable comments, suggestions, and for their support during the development of this paper.

REFERENCES

Babutzka, Dieter, (January/February 1995), "An OS/2 Performance Primer", OS/2 Developer, pps. 65-69.

Fournier, Roger, 1991. Practical Guide to Structured System Development and Maintenance. Yourdon Press Series. Englewood Cliffs, N.J.: Prentice-Hall, Inc., pps. 136-143.

Hardy, Jean E. (1992), "Efficient SAS Software Programming: A Version 6 Update", Proceedings of the Seventeenth Annual SAS Users Group International Conference, pps. 207-212.

Lafler, Kirk Paul (1985), "Optimization Techniques for SAS Applications", Proceedings of the Tenth Annual SAS Users Group International Conference, pps. 530-532.

Polzin, Jeffrey A. (1994), "DATA Step Efficiency and Performance", Proceedings of the Nineteenth Annual SAS Users Group International Conference, pps. 1574-1580.

Rabb, Merry G. (1991), "Using the SQL Procedure in Data Management Applications", Proceedings of the Sixteenth Annual SAS Users Group International Conference, pps. 600-604.

SAS Institute Inc. (1990), SAS Programming Tips: A Guide to Efficient SAS Processing, Cary, NC, USA.

Valentine-Query, Paige (1991), "Introduction to Efficient Programming Techniques", Proceedings of the Sixteenth Annual SAS Users Group International Conference, pps. 266-270.

Wilson, Steven A. (1994), "Techniques for Efficiently Accessing and Managing Data", Proceedings of the Nineteenth Annual SAS Users Group International Conference, pps. 207-212.

AUTHOR'S BIOGRAPHY

Kirk Paul Lafler is a SAS Institute Quality Partner™ and senior consultant for Software Intelligence Corporation. He has twenty two years experience using the SAS System. His areas of expertise include information-based systems analysis, design, and development; computer training; and programming using base-SAS, SAS/SQL, SAS/FSP, SAS/AF, Screen Control Language, and SAS/EIS software.

The author welcomes comments and suggestions. He can be reached at:

Kirk Paul Lafler
Software Intelligence Corporation
P.O. Box 1390
Spring Valley, California 91979-1390
(619) 660-2400
Compuserve: KirkLafler@CompuServe.com
Web Site: www.software-intelligence.com