

## Efficient SAS® Programming Techniques

Kirk Paul Lafler, Software Intelligence Corporation, Spring Valley, CA

### ABSTRACT

As SAS® Software becomes increasingly more popular, guidelines for its efficient use are critical. Areas deserving special consideration include program execution, I/O, disk space, and program maintenance. A collection of techniques and sample code are presented to illustrate numerous practical techniques for gaining efficiency while using SAS Software.

### INTRODUCTION

When developing SAS program code and/or applications, efficiency is not always given the attention it deserves, particularly in the early phases of development. System performance requirements can greatly affect the behavior an application exhibits. Active user participation is crucial to understanding application and performance requirements.

Attention should be given to each individual program function to assess performance criteria. Understanding user expectations, (preferably during the early phases of the application development process) often results in a more efficient application. Consequently, the difficulty associated with improving efficiency as coding nears completion is often minimized.

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

### EFFICIENCY OBJECTIVES

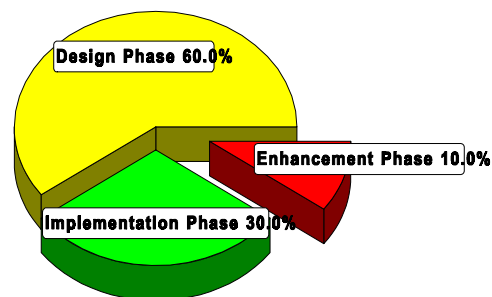
Efficiency objectives are best achieved when implemented as early as possible, preferably during the design phase. But when this is not possible, for example when customizing or inheriting an application, efficiency and performance techniques can still be "applied" to obtain some degree of improvement. 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."*

The simplest of requests can fall prey to one or more efficiency violations, such as retaining unwanted datasets in work space, not subsetting early to eliminate undesirable records, or reading wanted as well as unwanted variables. Much of an application's inefficiency can be avoided with better planning and knowing what works and what does not prior to beginning the coding process. Most people do not plan to fail - they just fail to plan. Fortunately, efficiency gains can be realized by following a few guidelines.

## Efficiency and Customization

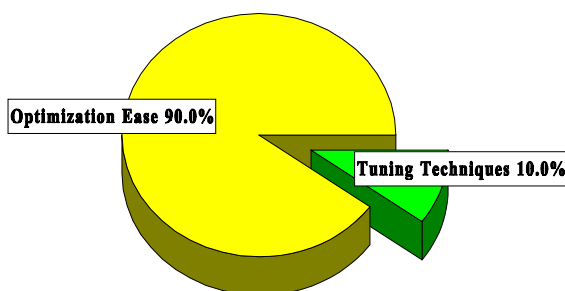


### GUIDELINES TO HOLD DEAR

The difference between an application that has been optimized versus one that has not is often dramatic. By adhering to practical guidelines, an application can achieve efficiency in direct relationship to economies of scale. Generally, the first 90% of efficiency improvements are gained relatively quickly and easily by applying simple strategies. It is 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.

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

## Efficiency Scale



Efficiency techniques are presented for the following resource areas: CPU time, data storage, I/O, memory, and programming time. Coding 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 CLASS statements in 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.

## 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."*

## 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

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 of an application.

Several techniques receive "strong" (between "Sometimes" and "Always"), but not unanimous, support among survey participants. Using system options to control resources; deleting unwanted WORK datasets; combining 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. A 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 provides added incentives and value for programmers and end-users, especially since reporting requirements are usually time critical."*

A very interesting new development is that more people are achieving greater efficiency by using the SQL Pass-Through Facility to access data stored in one or more database environments.

Techniques cited by survey participants as "Sometimes" being used to achieve efficiency. Using DATA step options, using data compression, conserving 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.

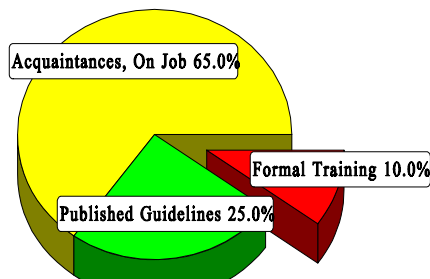
### LEARNING NECESSARY TECHNIQUES

So how do people learn about efficiency techniques? A small number learn through formal training. Others find published guidelines (e.g., book(s), manuals, articles, etc.) on the subject. The majority indicated they learn techniques as a result of a combination of prior experiences, through acquaintances (e.g., User Groups), and/or on the job.

Any improvement is better than no improvement. Consequently, adhering to a practical set of guidelines can benefit significantly for many years to come. Survey responses revealed the following concerns:

- 1) An insufficient level of formal training exists on 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.

## 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. Many learn valuable techniques through formal classroom instruction, while others find value in published guidelines such as books, manuals, articles, and videotapes. But the greatest value comes from 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

- Fournier, Roger, 1991. Practical Guide to Structured System Development and Maintenance. Yourdon Press Series. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 136-143.
- Hardy, Jean E. (1992), "Efficient SAS Software Programming: A Version 6 Update", Proceedings of the Seventeenth Annual SAS Users Group International Conference, 207-212.
- Lafier, Kirk Paul (1985), "Optimization Techniques for SAS Applications", Proceedings of the Tenth Annual SAS Users Group International Conference, 530-532.

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

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, 266-270.

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

#### TRADEMARKS

SAS is the registered trademark of SAS Institute Inc., Cary, NC, USA. SAS Institute Quality Partner is the trademark of SAS Institute Inc., Cary, NC, USA.

#### AUTHOR'S BIOGRAPHY

Kirk Paul Lafler is a SAS Institute Quality Partner™ and senior consultant with twenty-three years experience working with the SAS System. His expertise includes application design and development, training, and programming using base-SAS, SAS/SQL, ODS, SAS/FSP, SAS/AF, SCL, FRAME, and SAS/EIS software. Comments and suggestions can be sent to:

**Kirk Paul Lafler**  
**Software Intelligence Corporation**  
**P. O. Box 1390**  
**Spring Valley, California 91979-1390**  
**Voice: (619) 660-2400**  
**E-mail: [KirkLafler@CompuServe.com](mailto:KirkLafler@CompuServe.com)**  
**Web Site: [www.software-intelligence.com](http://www.software-intelligence.com)**

## 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?
 

<input type="checkbox"/> Requirements Definition Phase	<input type="checkbox"/> Testing Phase
<input type="checkbox"/> Analysis Phase	<input type="checkbox"/> Implementation Phase
<input type="checkbox"/> Design Phase	<input type="checkbox"/> Maintenance/Enhancement Phase
<input type="checkbox"/> 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 subset datasets.
  - \_\_\_\_\_ Use WHERE statements to subset datasets.
  - \_\_\_\_\_ 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 concatenate datasets.
  - \_\_\_\_\_ 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 copy datasets with built-in indexes.
  - \_\_\_\_\_ Use the DATA \_NULL\_ step to avoid creating a dataset 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**