ABSTRACT
As SAS® software becomes increasingly more popular, guidelines for its efficient use is critical. The Base-SAS software provides SAS users with a powerful programming language for accessing, analyzing, manipulating, and presenting data. This paper addresses useful coding techniques for SAS users on all operating system platforms. Attendees can expect to learn DATA and PROC step language statements and options that help conserve CPU, I/O, data storage, and memory resources while accomplishing tasks involving processing, ordering, grouping, and summarizing data.

INTRODUCTION
When developing SAS program code, 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 by adhering to best-practice coding techniques while 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:

- CPU Time
- Data Storage
- I/O
- Memory
- Programming Time

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.

ESSENTIAL SAS CODING GUIDELINES
The difference between programs that have been optimized versus those that have not can be dramatic. By adhering to best practice guidelines, an application can achieve improved performance. Generally, the first 90% of efficiency improvements are gained relatively quickly and easily by applying simple strategies, see figure 1. It is often the final 10% that, if pursued, proves to be a challenge. Consequently, you will need to be the judge of whether your program 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 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= data set options to retain desired variables.
2. Create and use indexes with large data sets.
3. Utilize macros for redundant code.
4. Use IF-THEN/ELSE statements to process data.
5. Use the DATASETS procedure COPY statement to copy data sets 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= data set options 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. Create user-defined format libraries.
6. Use DATA _NULL_ steps for processing null data sets.
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.

**Program Code Examples**

1. Using the KEEP= 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.

   ```sas
   data pg_movies;
   set wuss.movies
       (keep=title rating category);
   if rating = 'G';
   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.

   ```sas
   proc means data=wuss.movies;
   var length;
   class rating;
   run;
   ```

3. By using IF-THEN/ELSE statements the SAS System stops processing the conditional logic once a condition holds true for any observation.
data movies;
  set wuss.movies;
  if rating = 'G' then audience = 'General';
  else
    if rating = 'PG' then audience = 'Parental Guidance';
    else
      if rating = 'PG-13' then audience = 'Parental Guidance 13';
      else
        if rating = 'R' then audience = 'Adult';
  run;

4. To avoid using default lengths for variables in a SAS data set, 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.

data _null_;
  length pageno rptdate 4;
  set wuss.movies;
  file report header=h;
  put @10 title $30.
       @45 rating $5. ;
  return;
  h:
    rptdate=today();
    pageno + 1;
    put @25 'Classic Movies Report'
       / @1 rptdate mmddyy10.
       / @44 '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=wuss.movies n noobs;
  where rating = 'PG';
  title1 'PG-Rated Movies';
run;

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

proc sql;
  title1 'PG-Rated Movies';
  select *
    from wuss.movies
    where rating = 'PG'
    order by title;
quit;

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

data wuss.movies (compress = yes);
  < additional statements >
run;

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. Here are a few suggestions to keep in mind as you develop your best-practice coding techniques:

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.

CONCLUSION
The value of implementing efficiency and performance strategies into an application cannot 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.

REFERENCES


**ACKNOWLEDGMENTS**
I would like to thank Peter Godard and Cyndi Williamson (Data Warehousing & Database Management Section Co-Chair) for accepting my abstract and paper, as well as Miriam Cisternas and Marian Oshiro (Conference Co-Chairs), and the WUSS Leadership for their support of a great Conference.

**CONTACT INFORMATION**
Kirk Paul Lafler, a SAS Certified Professional® and former SAS Alliance Partner® (1996 - 2002) with more than 25 years of SAS software experience, provides consulting services and hands-on SAS training around the world. Kirk has written four books including PROC SQL: Beyond the Basics Using SAS by SAS Institute (available October 2004), Power SAS and Power AOL by Apress, and more than one hundred articles in professional journals and SAS User Group proceedings. His popular SAS Tips column appears regularly in the BASAS, HASUG, SANDS, SAS, and SESUG Newsletters and websites. Kirk can be reached at:

Kirk Paul Lafler  
Software Intelligence Corporation  
P.O. Box 1390  
Spring Valley, California 91979-1390  
Voice: 619-277-7350  
E-mail: KirkLafler@cs.com  
Web: www.software-intel.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.