Minimizing CPU Time of SAS Software® Applications and Getting Rewarded for It

Daniel J. Leprince, Viking Freight, Inc., San Jose, CA

Abstract
Optimizing SAS applications often results in substantial savings of computer resources which can translate into significant financial gains for your institution. The successive steps taken to improve the efficiency of a SAS application in an IBM MVS® environment will be presented. Topics covered will range from obtaining performance statistics, redesigning data flow, programming tips, and relating the savings to your administration.

Introduction
Increasing performance and efficiency of SAS applications can be a very rewarding activity for programmers, end-user and the organization. It only requires an inquisitive mind and a simple testing procedure. The whole process is based on the following principle, you can improve what you can measure. This paper will outline the terminology, baseline statistics, the testing procedure, and reporting the results to your administration.

Terminology
An application is defined as one or more related SAS programs that perform a specific function for an organization on a cyclic basis (Raithel, 1995). The presentation will focus on SAS applications rather than ad-hoc programs because any savings in computer resources increases every time the application runs.

Descriptive statistics of each task (PROCEDURE or DATA steps) can be obtained when selecting MEMRPT and FULLSTATS in an OPTIONS statement.

OPTIONS MEMRPT FULLSTATS;

These two SAS system options will provide statistics on task memory, total memory, CPU time and elapsed time, EXCP count and more. EXCP count is the total number of input/output operations performed by the task. EXCP count is proportional to the size of the file and according to Raithel (1995) they take milliseconds to complete and are the slowest events in the execution of a SAS task. Central Processor Unit (CPU) time is the amount of time a central processor spends executing instructions of a SAS program. Results are expressed in HH:MM:SS.xx, where H are hours, M are minutes, S seconds and x are hundredth of seconds. Elapsed time is the most popular and the most unreliable statistics because it reports the interval taken by a task or a program to complete. It is greatly influenced by the demands on the CPU (i.e., the number of jobs running at the same time as your application, your priority code, tape jobs, etc.). Comparison of CPU time and elapsed time of the same program submitted at different times during the day revealed that CPU time will vary within a 5% margin while elapsed time will vary by several folds. I used CPU time as the criteria to fine tune my application. You will have to decide what is the most critical performance statistic for your environment.
The CONTENTS procedure provides information about temporary or permanent SAS data set (i.e. number of observations, observation length, variable name, type, format, label, etc.). The following code will provide detailed information about the temporary work file TERMINAL: engine/host dependent information, list of variables and attributes, alphabetic list of indexes and sort information.

```sas
PROC CONTENTS DATA=WORK.TERMINAL DETAILS;
RUN;
```

For a more complete description of all statistics generated by these options or procedures, please refer to your SAS® Language and SAS® Procedures books.

Methodology

Tuning an application is a three steps process:

1. obtain descriptive statistics from the program you want to improve,
2. evaluate similar statistics from the modified program running against the same data set after introducing a single modification to the code, and finally,
3. compare both to decide whether there has been an increase in performance. Because of small variation in CPU time for a given application in my environment, decreases of CPU time by 5% or more were considered significant.

The application

The Work Measurement application was the one selected for fine tuning because it took several hours of elapsed time to run biweekly on an IBM® 4381 computer. This weekly application was manipulating more than 100 input variables associated with productivity and employee performance at each terminal. A terminal is a location where freight is received and distributed. Three types of final reports were generated: the terminal report, the region report and the administration report. Each of these reports consisted in several report units. A report unit being defined as more than 100 calculated variables generated by five PRINT procedures and one PLOT procedure. A terminal report consisted of a system unit, the region unit to which the terminal belongs and the terminal unit. A region report consisted of a system unit, the respective region unit and all the terminal units belonging to that region. The administration report consisted in a system summary unit, all region units and all terminal units. At the time the application was tuned, there was 52 terminals, five regions and one system. The administration report was printed locally while the regions and terminals report were send across the nation to TANDEM® printer locations.

Fine tuning

Significant savings in computer resources are often related to a more efficient data flow than programming tips. The first fine tuning step was associated with the review of the logic of the program. The old program was creating flat file reports that were finished products: 52 terminal reports, 5 region reports and 1 system report. During this process, the print section of the program was producing 276 report units divided into 156 terminal units, 62 region units and 58 system units. In other words, each terminal unit report was generated 3 times, each region report units were generated between 3 and 25 times, and the system report was generated 58 times!
A more efficient way is to generate each unit report only once and store each unit as a flat file with the PRINTTO procedure. Then, use JCL® code to concatenate report units into the final desired report (see JCL® code for IBM and Tandem printers). Once the reprogramming was completed, the 58 report units were generated in 1480 CPU seconds instead of 6566 CPU seconds. CPU time taken by JCL programs to concatenate units into the administration, the regions or the terminals reports and send them to their printer locations took less than 6 CPU seconds.

The second tuning step dealt with format and label statements that were loaded for each of the five PRINT procedures, a total of 290 times, respectively. By incorporating labels and formats as statements to the three temporary data sets prior to the print section of the program, it was no longer necessary to bring labels and formats into each PRINT procedure. Please note that you need to add a "label" data set option or a 'split=' option to the PRINT procedure to insure that labels will be associated to the printed variables. The average CPU time dropped from 1480 to 660 CPU seconds.

When using the CONTENTS procedure to ascertain that labels and formats were incorporated to temporary data sets, it became obvious that more than half of the variables were not used in the PRINT procedures. The KEEP= data set option was incorporated to the three temporary data sets (terminal, regions and system SAS work files) to keep only the variables needed in the print section. As a result, the number of variables from each temporary data set dropped from 230 to 106 variables resulting in similar reduction in record length and number of blocks used. The average CPU time dropped by 6%, from 660 to 618 CPU seconds.

The terminal work file was holding the data of 52 terminals. Terminal values were indexed with the DATASETS procedure to speed up the retrieval of the proper terminal by the five PRINT procedures. For a given terminal unit report, a WHERE statement was used to subset the terminal data set to the proper terminal value. The average CPU time dropped by 6%, from 618 to 580 CPU seconds.
/* SAMPLE OF KEEP= DATA SET OPTION, */
/* AND FORMAT & LABEL STATEMENTS */
...
DATA TERMINAL (KEEP=VAR1 VAR2 ... VAR106);
SET TERMINAL;
FORMAT VAR1 9.2
VAR2 8.
VAR3 10.

...;
LABELS VAR1 = '##DAYS';
VAR2 = 'LABOR\?ACTUAL\?COST';
VAR3 = 'FUEL\?MILES';
...;
RUN;
...
/* INDEXING VARIABLE TERMINAL */
/* LOCATION (TERMLOC) */
/* FROM TEMPORARY TERMINAL WORK FILE */
...
PROC DATASETS LIBRARY=WORK;
MODIFY TERMINAL;
INDEX CREATE TERMLOC;
RUN;
QUIT;
...
/* SAMPLE OF PRINT PROCEDURE */
/* NOTE: FORMATS & LABELS ALREADY */
/* INCORPORATED IN THE TEMPORARY */
/* TERMINAL FILE */
PROC PRINT DATA=TERMINAL SPLIT='?'; /*LABEL*/;
WHERE TERMLOC='TERM1';
TITLE1 'TERMINAL TERM1 WEEKLY REPORT';
VAR VAR1 VAR2 ... VAR20;
RUN;
PROC PRINT DATA=TERMINAL SPLIT='?'; /*LABEL*/;
WHERE TERMLOC='TERM1';
TITLE1 'TERMINAL TERM1 WEEKLY REPORT';
VAR VAR21 VAR22 ... VAR40;
RUN;
...;

From the original baseline, the report was now running 11.3 times faster. It was time to deliver the information to the administration in a format that executives are used to reviewing. Although the application ran twice a week, it made more sense to present the results by period (four working weeks, 13 periods per year) and year. The Management Information System department provided an average time of 330 CPU seconds for SAS production on the IBM 4381 computer. Although we were not billed for CPU time, it became obvious that some members of the administration would better relate to money figures than CPU time or other computer performance statistics. A conservative estimate of $0.15 per CPU second was obtained from another company. It was now time to present the savings to the administration.

A total of 5986 CPU seconds would be saved for each run (47,888 CPU per period). It also meant that 145 SAS production programs could run every period in the time saved by this increase in performance. This could relate to $7200.00 in savings per period or $93,000.00 per year if we had to contract the work to another company and pay for CPU time. From the company's Profit Improvement program, the project was awarded Profit Improvement Champion for January 1996.

Obviously, most application performances cannot be improved by an order of magnitude. More often than not, small gains in efficiency are insignificant to be reported on their own. In this case, regrouping several improvements into one report to be reviewed by the administration may be the solution. These small increments in performance can add up to significant savings to your organization and your recognition as a savvy programmer.

Conclusion

Fine tuning applications requires the programmer to compare baseline statistics (control) to statistics from a modified program (test) that process the same data set. It is preferable to test one type of modification at a time and keep record of each change. When tunings efforts no longer generate savings in computer resources, it is time to summarize significant changes for future reference. Regrouping several fine tuning efforts on different applications may be required
when individual savings in computer resources are too small to be reported on their own. Savings need to be reported to your administration in terms that they can understand (i.e. computer performance statistics, average application execution time, money, etc.).

References


Author’s Address
Daniel J. Leprince
Viking Freight, Inc., Corporate Planning,
6411 Guadalupe Mines Road,
San Jose CA 95120

SAS is a registered trademark of SAS Institute Inc. in the USA and other countries. IBM, MVS/TSO and JCL are registered trademarks or trademarks of International Machines Corporation. ® indicates USA registration.