Paper 006-2008

# Introducing the SAS® Code Analyzer

Eric Thies and Rick Langston, SAS Institute Inc., Cary, NC

## ABSTRACT

This paper introduces the PROC SCAPROC procedure, the SAS Code Analyzer that is new in Release 9.2 of Base SAS® Software. We will examine the advantages of using the procedure, its syntax and phases of execution, and the output that the procedure can produce.

## ADVANTAGES OF THE PROC SCAPROC PROCEDURE

If you are responsible for maintaining a large legacy SAS application, you are probably interested in finding ways to improve the performance of the application. This is especially true if the application makes many passes through the same data sets, and you know that those passes could potentially be run in parallel. However, it can be difficult and tedious to convert an application to run parallel steps, even when you have a good understanding of the application.

The new SCAPROC Base SAS procedure for Release 9.2 can assist in this process. PROC SCAPROC is an instrumenting procedure that activates hooks within the SAS System core processes. These hooks allow information to be recorded and subsequently analyzed to assist in parallelizing the SAS code. Once the SAS code has been parallelized, you can use it in conjunction with SAS Grid Manager to enable various SAS steps to run in parallel. This improves the throughput execution time of the entire SAS job.

SAS jobs consist of many steps or tasks. For example, a SAS job could have a DATA step that creates data set A, then three PROC SUMMARY steps that all operate on data set A. Each step creates separate output data sets, and each of the steps executes sequentially. The DATA step is completed first, then the first PROC SUMMARY step runs to completion, followed by the second and third PROC SUMMARY steps. As these steps were running, PROC SCAPROC would gather information to determine if the three PROC SUMMARY steps could run in parallel.

## USING THE PROCEDURE

Let's see how you would run the PROC SCAPROC procedure to analyze the step described above:

```
proc scaproc; record 'out.txt' grid 'gridout.txt'; run;

data a;
     input x y z @@; cards;
1 2 3 4 5 6 7 8 9
run;
```

```
proc summary data=a;
     var x;
     output out=new1 mean=mx;
     run;
proc summary data=a;
     var y;
     output out=new2 mean=my;
     run;
proc summary data=a;
     var z;
     output out=new3 mean=mz;
     run;

proc scaproc; write; run;
```

The PROC step is inserted in front of the SAS code to be analyzed. This execution of PROC SCAPROC triggers the core components of the SAS System to provide information for activities such as opening and closing SAS data sets. The GRID statement of the SCAPROC step indicates which file to use in recording information. After all the PROC SUMMARY steps have reached completion, PROC SCAPROC is reinvoked, indicating that it is time to analyze the information gathered from the previous steps.

**OUTPUT FROM THE PROCEDURE**

Here is the output that is placed into out.txt that was created by the above step:

```
/* JOBSPLIT: DATASET OUTPUT SEQ WORK.A.DATA */
/* JOBSPLIT: ELAPSED 62  */
/* JOBSPLIT: PROCNAME DATASTEP */
/* JOBSPLIT: STEP SOURCE FOLLOWS */

data a;
     input x y z @@; cards;
1 2 3 4 5 6 7 8 9
run;


/* JOBSPLIT: DATASET INPUT SEQ WORK.A.DATA */
/* JOBSPLIT: DATASET OUTPUT SEQ WORK.NEW1.DATA */
/* JOBSPLIT: SYMBOL GET SYSSUMTRACE */
/* JOBSPLIT: ELAPSED 46  */
/* JOBSPLIT: PROCNAME SUMMARY */
/* JOBSPLIT: STEP SOURCE FOLLOWS */
```

```
proc summary data=a;
     var x;
     output out=new1 mean=mx;
     run;

/* JOBSPLIT: DATASET INPUT SEQ WORK.A.DATA */
/* JOBSPLIT: DATASET OUTPUT SEQ WORK.NEW2.DATA */
/* JOBSPLIT: SYMBOL GET SYSSUMTRACE */
/* JOBSPLIT: ELAPSED 32  */
/* JOBSPLIT: PROCNAME SUMMARY */
/* JOBSPLIT: STEP SOURCE FOLLOWS */
proc summary data=a;
     var y;
     output out=new2 mean=my;
     run;

/* JOBSPLIT: DATASET INPUT SEQ WORK.A.DATA */
/* JOBSPLIT: DATASET OUTPUT SEQ WORK.NEW3.DATA */
/* JOBSPLIT: FILE OUTPUT U:\m900\com\out.txt */
/* JOBSPLIT: SYMBOL GET SYSSUMTRACE */
/* JOBSPLIT: ELAPSED 46  */
/* JOBSPLIT: PROCNAME SUMMARY */
/* JOBSPLIT: STEP SOURCE FOLLOWS */
proc summary data=a;
     var z;
     output out=new3 mean=mz;
     run;


/* JOBSPLIT: END */
```

The output file is an executable SAS program that consists of the SAS statements that were executed, but with the special JOBSPLIT comments, which contain additional information. The information includes input and output data set activity, flat file activity, macro variable access, step names, and elapsed time. This output file can be post-processed by your own applications, but SCAPROC can post-process it, too, and produce statements for execution on a grid.

If you want grid processing, the first SCAPROC step should contain a GRID option, as is shown here:

```
  proc scaproc; record 'out.txt' grid 'grid.txt'; run;
```

When the final SCAPROC step (with the WRITE statement) is seen, the grid.txt file will contain all of the necessary code to connect to remote sessions and execute portions of

the SAS code on different nodes of the grid. In this example, the three separate PROC SUMMARY steps could be executed simultaneously on different nodes because none of the steps depends on output that is produced by any of the other PROC SUMMARY steps.

The output from the analysis phase is a set of RSUBMIT blocks that contains the steps of the original program. When particular steps of the original program are independent of each other, each of these steps will be submitted to its own remote session, and the steps will run in parallel to each other.

Please note that any global statements such as LIBNAME statements should not be part of the SCAPROC processing. Global statements should probably appear in autoexec streams for the individual nodes of the grid. You should experiment with the resultant grid stream to ensure that you will have proper access to all of the input data sets during the separate executions on the grid nodes.

Note that you can specify the ATTR option on the RECORD statement to also obtain information about every SAS data set that is read from or written to. This can produce a lot more output, but it can be worthwhile if you are trying to gather metadata about the SAS data sets that you are processing.

## RUN-TIME CHARACTERISTIC OF THE ANALYZER

It is important to note that the SAS Code Analyzer is a run-time utility. It does not examine the SAS code, but instead records information that is based on the activity during DATA steps and procedure steps. But because the SAS Code Analyzer is a run-time utility, there are no restrictions on what the SAS code is doing. Any amount of macro usage, %INCLUDEs, and so on, can appear without any adverse effects on the analyzer.

## CONCLUSION

The new SCAPROC procedure simplifies the potentially complex and time-consuming process of preparing a SAS application for use in a grid environment to allow for faster throughput.

## REFERENCES

A U.S. patent is currently pending for the technology that is used in the SCAPROC procedure.

## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at
Eric.Thies@sas.com or Rick.Langston@sas.com