Paper 313-2010

Thoroughly Modern SAS®: The SAS® Code Analyzer Helps Bring Programs Up to Date

Merry Rabb, SAS Institute Inc., Cary, NC

ABSTRACT

How would you like to revive and reuse your older SAS® jobs to make use of some newer capabilities? There are some easy ways to modernize your use of SAS without manually rewriting your existing SAS programs. The SAS Code Analyzer is the secret behind some new features for importing SAS code into SAS solutions such as SAS® Data Integration Studio and SAS® Enterprise Guide®. You can also use the SAS Code Analyzer to create a version of a program to run in a distributed computing environment such as with SAS® Grid Manager. This presentation will provide an overview of the SAS Code Analyzer and will include some practical examples of using it to create programs for distributed parallel processing.

INTRODUCTION

If your responsibilities include maintaining large legacy SAS programs, then you probably wish you had time to bring some of those programs up-to-date. Maybe you'd like to modify them to take advantage of some of the more recent advancements in SAS capabilities, use them in the SAS Data Integration Studio, use them in SAS Enterprise Guide, or make modifications to get them to run more efficiently. Even if the program was written a long time ago, if it still runs properly, then it's usually considered too costly and time consuming to spend time pouring through the code and making manual changes.

The SAS Code Analyzer, introduced with SAS 9.2, is a useful tool that can help you maintain and modify these legacy programs. The SAS Code Analyzer, or SCAPROC procedure, is a Base SAS® procedure. It executes an existing SAS program and while the program runs, it collects and analyzes information related to the SAS steps, input and output data, and any dependencies. It records information that can be used to enhance the manageability and efficiency of the program. In this paper, you will be introduced to the SAS Code Analyzer, understand what type of information it collects and records, and see some examples of using it to create a new SAS program that has been divided into tasks that can be run in parallel on multiple processors.

USING THE SAS CODE ANALYZER

Despite the name SAS Code Analyzer, PROC SCAPROC does not read or interpret the SAS code itself. Rather, it gathers information from an executing SAS program. This means that in order to make use of PROC SCAPROC, you must have not only your program, but data and anything else that is required for the program to fully execute. When PROC SCAPROC runs, it will record and analyze information and generate commented SAS code. The resulting code can be:

- Imported into the SAS Data Integration Studio (Code Importer feature).
- Imported into SAS Enterprise Guide.
- Executed on multiple processors if the GRID option is used.
- Manually analyzed by a programmer to look for other ways to improve the program.

There are three steps you use in your programming environment in order to use PROC SCAPROC:

1. Activate PROC SCAPROC, including specifying options and designating a location for results files.
2. Submit the existing SAS program.
3. Close the PROC SCAPROC process and write out the results files.

The results files are new versions of the program that contain special comments. There is always one results file of code with comments detailing the information gathered such as data sets read and written, dependencies, and so forth. If you are using the GRID option, then a second results file is created that includes statements to grid enable the program and to remote submit the SAS code in separate blocks to run in parallel.

AN EXAMPLE

Here is a sample program that will serve to illustrate the basics of what PROC SCAPROC does and how it works. The program consists of a DATA step to read in some data and a PROC step to perform computations:

```
data a;
  attrib x label="xnum" format = 1.;
  input x y z ;
  cards;
1 2 3
3 4 3
1 2 2
4 5 6
run;

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

In order to run PROC SCAPROC with this program, we need to add code at the beginning of the program to invoke the procedure:

```
proc scaproc;
  record 'example1.txt' ;
run;
```

Note that we are writing the results out to a file called example1.txt. After the main body of the program we will add code to close the PROC SCAPROC process:

```
proc scaproc; write; run;
```

When we submit this program, the program executes within the wrapper of the PROC SCAPROC process. The data lines are read, a data set a is created, and the SUMMARY procedure executes against data set a and creates data set new1. Shown below are the contents of the results file, example1.txt, after this program has run.

The special comments prefaced with the label “JOBSPLIT:” contain the information that is captured about the executing SAS steps and the input and output data. The information in these comments can be used by other SAS applications. For example, the SAS Data Integration Studio reads this output and creates jobs, tables, and libraries in the SAS® Metadata Server as part of importing the program. Once the job is imported, you can use all of the SAS Data Integration Studio features including debugging and the ability to view performance metrics when the job is run.

Contents of the file example1.txt:

```
/* JOBSPLIT: DATASET OUTPUT SEQ WORK.A.DATA */
/* JOBSPLIT: LIBNAME WORK V9 'C:\DOCUME~1\sasmgr\LOCALS~1…'  */
/* JOBSPLIT: SYMBOL GET SYS_IOUSEEE */
/* JOBSPLIT: ELAPSED 94  */
/* JOBSPLIT: PROCNAME DATASTEP */
/* JOBSPLIT: STEP SOURCE FOLLOWS */

data a;
  attrib x label="xnum" format = 1.;
  input x y z ;
```

```
  cards;
1 2 3
3 4 3
1 2 2
4 5 6
run;


/* JOBSPLIT: DATASET INPUT SEQ WORK.A.DATA */
/* JOBSPLIT: LIBNAME WORK V9 'C:\DOCUME~1\sasmgr\LOCALS~1…' */
/* JOBSPLIT: DATASET OUTPUT SEQ WORK.NEW1.DATA */
/* JOBSPLIT: LIBNAME WORK V9 'C:\DOCUME~1\sasmgr\LOCALS~1…' */
/* JOBSPLIT: FILE OUTPUT C:\Documents and Settings\sasmgr\example1.txt */
/* JOBSPLIT: SYMBOL GET SYS_IOUSEEE */
/* JOBSPLIT: SYMBOL GET SYSSUMTRACE */
/* JOBSPLIT: ELAPSED 172  */
/* JOBSPLIT: PROCNAME SUMMARY */
/* JOBSPLIT: STEP SOURCE FOLLOWS */

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


/* JOBSPLIT: END */
```

Additional information can be created in the JOBSPLIT comments depending on the options you use. If you add the ATTR option to the RECORD statement, then PROC SCAPROC will also output additional information about the variables in data sets. Adding the OPENTIMES option to the RECORD statement will cause PROC SCAPROC to output information about when input data sets were opened and their size. For a complete explanation of the types of JOBSPLIT comments that might appear and the information that they describe, please refer to the PROC SCAPROC documentation in the *Base SAS® 9.2 Procedures Guide*.

USING THE GRID OPTION

You can manually examine an existing SAS program, identify individual steps and interdependencies, and then re-work the code so it can run on a grid. Most SAS programs are a lot longer and more complex than this simple example. The process of examining and re-working the code can be very time consuming, especially if you are not the person who created the program in the first place. The SAS Code Analyzer can be used to create a grid-enabled parallel SAS job that can save you much of this effort. PROC SCAPROC will execute the logic of the existing SAS program and output information that identifies the input and output data sets and the interdependencies of the procedure and DATA steps in the program, then use this information to output a second grid-enabled version of the program. It is highly recommended that you review, run, and test this new grid-enabled version of the application before you consider it a production application.

Here is a slightly longer program we can run to illustrate the use of the GRID option of PROC SCAPROC to generate a grid enabled version of the program. The regular output that is similar to what we saw above will be found in the file out1.txt and the grid enabled output will be found in the file out1.grid. The GRID option can be used as long as you have SAS Grid Manager or SAS/CONNECT® licensed.

```
proc scaproc;
 record 'out1.txt' grid 'out1.grid';
run;

data a;
  attrib x label="xnum" format = 1.;
  input x y z ;
  cards;
1 2 3
```

```
3 4 3
1 2 2
4 5 6
run;

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

proc summary data=a;
  var y;
  output out=new2 mean=meany;
run;

proc summary data=a;
  var z;
  output out=new3 mean=meanz;
run;

proc scaproc; write; run;
```

After running this program, there will be two results files. The one named out1.txt will contain the code and the "JOBSPLIT:" comments similar to what was shown in the first example. The contents of the second results file named out1.grid are shown below. The following lists several important things to note about this output:

- The initial comments say that the job has four tasks, but none of them can be RSUBMITed (run in parallel on remote processors) because they use WORK data sets. This is a very important thing to remember about grid enabling a SAS program – any data used by the program will need to be permanently available to all processors or nodes on the grid. WORK data sets present a problem because they are temporary – they cannot be created in a SAS job on one processor then be used by a SAS job on another processor. In a moment we will look at how to deal with this issue.
- Code has been inserted to enable grid processing (a call to a function gridsvc_enable) and macros are used to start up the remote SAS sessions (%scagrid_sessions) and to close the sessions when the tasks are complete (%scagrid_waitfors).
- There is a macro variable called SCAGRID_SESSIONS that defaults to 3. This defines the number of SAS/CONNECT sessions that will be initiated. For example, you can change this value to correspond to how many processors you actually have or how many SAS sessions you want started as part of running on a grid.
- This is an executable program. As the comments indicate, if you submit this program, then the tasks will actually run locally. In our next example we will look at a version of this program that can make use of multiple grid nodes.

Contents of the file out1.grid:

```
/*-------------------------------------------------------------------------*/
/* There are 4 tasks in this job.                                          */
/* 0 of these tasks can be RSUBMITed.                                      */
/* 4 of the tasks cannot be RSUBMITed because they use data sets in WORK.  */
/* These 4 tasks used 61 units of time.                                    */
/* The longest task took 32 units of time, 52.5% of total time.            */
/*-------------------------------------------------------------------------*/
/*-------------------------------------------------------------------------*/
/* This is the user-modifiable number of connect sessions                  */
/* Numbers of sessions should be between 1 and 20 inclusive.               */
/*-------------------------------------------------------------------------*/
%let SCAGRID_SESSIONS=3;

/*-------------------------------------------------------------------------*/
/* *** Please don't edit anything below this line.                         */
/* *** Regenerate the file if you need to make changes.                    */
```

```
/*-------------------------------------------------------------------------*/

/*-------------------------------------------------------------------------*/
/* Enable grid service                                                     */
/*-------------------------------------------------------------------------*/
%let rc=%sysfunc(grdsvc_enable(_all_, resource=SASMain));

/*-------------------------------------------------------------------------*/
/* This macro starts up the connect sessions                               */
/*-------------------------------------------------------------------------*/
%macro scagrid_sessions(count);
  %do i = 1 %to &count;
    signon sess&i connectwait=no;
    %put Session started on grid node %sysfunc(grdsvc_getname(sess&i));
  %end;
%mend scagrid_sessions;

/*-------------------------------------------------------------------------*/
/* Start up our connect sessions.                                          */
/*-------------------------------------------------------------------------*/
%scagrid_sessions(&SCAGRID_SESSIONS);

/*-------------------------------------------------------------------------*/
/* This function call initializes data structures for our SCAGRID functions.  */
/* We pass in the number of sessions and the number tasks in this job.     */
/*-------------------------------------------------------------------------*/
proc scaproc; startup 4 &SCAGRID_SESSIONS; run;

/*-------------------------------------------------------------------------*/
/* TASK 1 run locally                                                      */
/*-------------------------------------------------------------------------*/
/* I/O Activity                                                            */
/*-------------------------------------------------------------------------*/
/* DATASET    OUTPUT SEQ   created in task 1 WORK.A.DATA                    */
/*-------------------------------------------------------------------------*/
/* Symbol activity                                                         */
/*-------------------------------------------------------------------------*/
/* SYMBOL GET task:0 used by subsequent task:no  Name:SYS_IOUSEEE          */
/*-------------------------------------------------------------------------*/
/* ELAPSED 0                                                               */
/*-------------------------------------------------------------------------*/

data a;
  attrib x label="xnum" format = 1.;
  input x y z ;
  cards;
1 2 3
3 4 3
1 2 2
4 5 6
run;


/*-------------------------------------------------------------------------*/
/* TASK 2 run locally                                                      */
/*-------------------------------------------------------------------------*/
/* I/O Activity                                                            */
/*-------------------------------------------------------------------------*/
/* DATASET    INPUT  SEQ   created in task 1 WORK.A.DATA                    */
/* DATASET    OUTPUT SEQ   created in task 2 WORK.NEW1.DATA                 */
/*-------------------------------------------------------------------------*/
/* Symbol activity                                                         */
/*-------------------------------------------------------------------------*/
```

```
/* SYMBOL GET task:0 used by subsequent task:no  Name:SYS_IOUSEEE             */
/* SYMBOL GET task:0 used by subsequent task:no  Name:SYSSUMTRACE             */
/*----------------------------------------------------------------------------*/
/* ELAPSED 14                                                                 */
/*----------------------------------------------------------------------------*/
proc summary data=a;
  var x;
  output out=new1 mean=meanx;
run;


/*----------------------------------------------------------------------------*/
/* TASK 3 run locally                                                         */
/*----------------------------------------------------------------------------*/
/* I/O Activity                                                               */
/*----------------------------------------------------------------------------*/
/* DATASET    INPUT  SEQ   created in task 1 WORK.A.DATA                      */
/* DATASET    OUTPUT SEQ   created in task 3 WORK.NEW2.DATA                   */
/*----------------------------------------------------------------------------*/
/* Symbol activity                                                            */
/*----------------------------------------------------------------------------*/
/* SYMBOL GET task:0 used by subsequent task:no  Name:SYS_IOUSEEE             */
/* SYMBOL GET task:0 used by subsequent task:no  Name:SYSSUMTRACE             */
/*----------------------------------------------------------------------------*/
/* ELAPSED 32                                                                 */
/*----------------------------------------------------------------------------*/
proc summary data=a;
  var y;
  output out=new2 mean=meany;
run;


/*----------------------------------------------------------------------------*/
/* TASK 4 run locally                                                         */
/*----------------------------------------------------------------------------*/
/* I/O Activity                                                               */
/*----------------------------------------------------------------------------*/
/* DATASET    INPUT  SEQ   created in task 1 WORK.A.DATA                      */
/* DATASET    OUTPUT SEQ   created in task 4 WORK.NEW3.DATA                   */
/* FILE       OUTPUT       created in task 4 C:\Documents and
Settings\sasmgr\out1.txt  */
/*----------------------------------------------------------------------------*/
/* Symbol activity                                                            */
/*----------------------------------------------------------------------------*/
/* SYMBOL GET task:0 used by subsequent task:no  Name:SYS_IOUSEEE             */
/* SYMBOL GET task:0 used by subsequent task:no  Name:SYSSUMTRACE             */
/*----------------------------------------------------------------------------*/
/* ELAPSED 15                                                                 */
/*----------------------------------------------------------------------------*/
proc summary data=a;
  var z;
  output out=new3 mean=meanz;
run;


/*----------------------------------------------------------------------------*/
/* This macro issues waitfors and signoffs for our sessions.                  */
/*----------------------------------------------------------------------------*/
%macro scagrid_waitfors(count);
  %do i = 1 %to &count;
    waitfor sess&i;
    signoff sess&i;
  %end;
```

```
%mend scagrid_waitfors;

/*-------------------------------------------------------------------------*/
/* Wait for and sign off all sessions.                                     */
/*-------------------------------------------------------------------------*/
%scagrid_waitfors(&SCAGRID_SESSIONS);

/*-------------------------------------------------------------------------*/
/* Termination for our  SCAGRID functions.                                 */
/*-------------------------------------------------------------------------*/
proc scaproc; shutdown; run;

/*-------------------------------------------------------------------------*/
/* All done.                                                               */
/*-------------------------------------------------------------------------*/
```

GRID ENABLING PROGRAMS THAT USE TEMPORARY DATA SETS

You cannot take a program that passes temporary data sets from step to step and divide it up over multiple processors. In order to grid enable such a program, you need to make the data that is shared among the different steps into permanently stored data sets. Obviously, this could be done by manually modifying the program, but another way is to make use of the "USER=" system option. By specifying this option, you can use one-level names to reference permanent SAS files in SAS statements. For example, if we add the following statement to the above example, then we will get a different result in the results file from PROC SCAPROC:

```
options user='c:\temp';
```

When we add that option to the program, the resulting out1.grid file is shown below. The following lists some differences to note between this version of the results file and the one above:

- All four tasks can be RSUBMITed. None of the tasks need to be run locally.
- Before the first proc summary step we see comments that indicate that this step (Task 2) is dependent on Task 1 (the DATA step) and the following code is inserted to force a synchronization with Task 1:

  ```
  proc scaproc; taskwait 1; run;
  ```

  There are similar comments about dependency on Task 1 for the subsequent two PROC SUMMARY steps.

```
/*-------------------------------------------------------------------------*/
/* There are 4 tasks in this job.                                          */
/* 4 of these tasks can be RSUBMITed.                                      */
/* These 4 tasks used 170 units of time.                                   */
/* The longest task took 63 units of time, 37.1% of total time.            */
/*-------------------------------------------------------------------------*/
/*-------------------------------------------------------------------------*/
/* This is the user-modifiable number of connect sessions                  */
/* Numbers of sessions should be between 1 and 20 inclusive.               */
/*-------------------------------------------------------------------------*/
%let SCAGRID_SESSIONS=3;

/*-------------------------------------------------------------------------*/
/* *** Please don't edit anything below this line.                         */
/* *** Regenerate the file if you need to make changes.                    */
/*-------------------------------------------------------------------------*/
/*-------------------------------------------------------------------------*/
/* This is the USER option from the original run of the job.               */
/*-------------------------------------------------------------------------*/
libname USER "c:\temp";

/*-------------------------------------------------------------------------*/
```

```
/* Enable grid service                                                          */
/*------------------------------------------------------------------------------*/
%let rc=%sysfunc(grdsvc_enable(_all_, resource=SASMain));

/*------------------------------------------------------------------------------*/
/* This macro starts up the connect sessions                                    */
/*------------------------------------------------------------------------------*/
%macro scagrid_sessions(count);
  %do i = 1 %to &count;
    signon sess&i connectwait=no inheritlib=(user) ;
    %put Session started on grid node %sysfunc(grdsvc_getname(sess&i));
  %end;
%mend scagrid_sessions;

/*------------------------------------------------------------------------------*/
/* Start up our connect sessions.                                               */
/*------------------------------------------------------------------------------*/
%scagrid_sessions(&SCAGRID_SESSIONS);

/*------------------------------------------------------------------------------*/
/* This function call initializes data structures for our SCAGRID functions.  */
/* We pass in the number of sessions and the number tasks in this job.        */
/*------------------------------------------------------------------------------*/
proc scaproc; startup 4 &SCAGRID_SESSIONS; run;

/*------------------------------------------------------------------------------*/
/* TASK 1 run in rsubmit                                                        */
/*------------------------------------------------------------------------------*/
/* I/O Activity                                                                 */
/*------------------------------------------------------------------------------*/
/* DATASET    OUTPUT SEQ   created in task 1 USER.A.DATA                        */
/*------------------------------------------------------------------------------*/
/* Symbol activity                                                              */
/*------------------------------------------------------------------------------*/
/* SYMBOL GET task:0 used by subsequent task:no  Name:SYS_IOUSEEE               */
/*------------------------------------------------------------------------------*/
/* ELAPSED 63                                                                   */
/*------------------------------------------------------------------------------*/
/*------------------------------------------------------------------------------*/
/* Get an available session with the scagrid_gs function                        */
/*------------------------------------------------------------------------------*/
proc scaproc; getsession 1 "sess"; run;
%put sess=&sess;
/*------------------------------------------------------------------------------*/
/* rsubmit for task 1                                                           */
/*------------------------------------------------------------------------------*/
rsubmit &sess sysrputsync=yes cmacvar=scagrid_task_1;

data a;
  attrib x label="xnum" format = 1.;
  input x y z ;
  cards;
1 2 3
3 4 3
1 2 2
4 5 6
run;


endrsubmit;


/*------------------------------------------------------------------------------*/
```

```
/* TASK 2 run in rsubmit                                                      */
/*----------------------------------------------------------------------------*/
/* I/O Activity                                                               */
/*----------------------------------------------------------------------------*/
/* DATASET    INPUT  SEQ   created in task 1 USER.A.DATA                      */
/* DATASET    OUTPUT SEQ   created in task 2 USER.NEW1.DATA                   */
/*----------------------------------------------------------------------------*/
/* Symbol activity                                                            */
/*----------------------------------------------------------------------------*/
/* SYMBOL GET task:0 used by subsequent task:no  Name:SYS_IOUSEEE             */
/* SYMBOL GET task:0 used by subsequent task:no  Name:SYSSUMTRACE             */
/*----------------------------------------------------------------------------*/
/* ELAPSED 30                                                                 */
/*----------------------------------------------------------------------------*/
/*----------------------------------------------------------------------------*/
/* Dependencies                                                               */
/* Depends on task 1 DATASET USER.A.DATA                                      */
/* Highest task depended on: 1                                                */
/*----------------------------------------------------------------------------*/
/*----------------------------------------------------------------------------*/
/* Sync with task 1 (USER.A.DATA)                                             */
/*----------------------------------------------------------------------------*/
proc scaproc; taskwait 1; run;
/*----------------------------------------------------------------------------*/
/*----------------------------------------------------------------------------*/
/* Get an available session with the scagrid_gs function                      */
/*----------------------------------------------------------------------------*/
proc scaproc; getsession 2 "sess"; run;
%put sess=&sess;
/*----------------------------------------------------------------------------*/
/* rsubmit for task 2                                                         */
/*----------------------------------------------------------------------------*/
rsubmit &sess sysrputsync=yes cmacvar=scagrid_task_2;
proc summary data=a;
  var x;
  output out=new1 mean=meanx;
run;


endrsubmit;


/*----------------------------------------------------------------------------*/
/* TASK 3 run in rsubmit                                                      */
/*----------------------------------------------------------------------------*/
/* I/O Activity                                                               */
/*----------------------------------------------------------------------------*/
/* DATASET    INPUT  SEQ   created in task 1 USER.A.DATA                      */
/* DATASET    OUTPUT SEQ   created in task 3 USER.NEW2.DATA                   */
/* DATASET    OUTPUT MULTI created in task 3 WORK._TF0005.UTILITY             */
/*----------------------------------------------------------------------------*/
/* Symbol activity                                                            */
/*----------------------------------------------------------------------------*/
/* SYMBOL GET task:0 used by subsequent task:no  Name:SYS_IOUSEEE             */
/* SYMBOL GET task:0 used by subsequent task:no  Name:SYSSUMTRACE             */
/*----------------------------------------------------------------------------*/
/* ELAPSED 62                                                                 */
/*----------------------------------------------------------------------------*/
/*----------------------------------------------------------------------------*/
/* Dependencies                                                               */
/* Depends on task 1 DATASET USER.A.DATA                                      */
/* Highest task depended on: 1                                                */
/*----------------------------------------------------------------------------*/
```

```
/*-------------------------------------------------------------------------*/
/* Get an available session with the scagrid_gs function                   */
/*-------------------------------------------------------------------------*/
proc scaproc; getsession 3 "sess"; run;
%put sess=&sess;
/*-------------------------------------------------------------------------*/
/* rsubmit for task 3                                                      */
/*-------------------------------------------------------------------------*/
rsubmit &sess sysrputsync=yes cmacvar=scagrid_task_3;
proc summary data=a;
  var y;
  output out=new2 mean=meany;
run;


endrsubmit;


/*-------------------------------------------------------------------------*/
/* TASK 4 run in rsubmit                                                   */
/*-------------------------------------------------------------------------*/
/* I/O Activity                                                            */
/*-------------------------------------------------------------------------*/
/* DATASET    INPUT  SEQ   created in task 1 USER.A.DATA                   */
/* DATASET    OUTPUT SEQ   created in task 4 USER.NEW3.DATA                */
/* FILE       OUTPUT       created in task 4 C:\Documents...\sasmgr\out1.txt  */
/*-------------------------------------------------------------------------*/
/* Symbol activity                                                         */
/*-------------------------------------------------------------------------*/
/* SYMBOL GET task:0 used by subsequent task:no  Name:SYS_IOUSEEE           */
/* SYMBOL GET task:0 used by subsequent task:no  Name:SYSSUMTRACE           */
/*-------------------------------------------------------------------------*/
/* ELAPSED 15                                                              */
/*-------------------------------------------------------------------------*/
/*-------------------------------------------------------------------------*/
/* Dependencies                                                            */
/* Depends on task 1 DATASET USER.A.DATA                                   */
/* Highest task depended on: 1                                             */
/*-------------------------------------------------------------------------*/
/*-------------------------------------------------------------------------*/
/* Get an available session with the scagrid_gs function                   */
/*-------------------------------------------------------------------------*/
proc scaproc; getsession 4 "sess"; run;
%put sess=&sess;
/*-------------------------------------------------------------------------*/
/* rsubmit for task 4                                                      */
/*-------------------------------------------------------------------------*/
rsubmit &sess sysrputsync=yes cmacvar=scagrid_task_4;
proc summary data=a;
  var z;
  output out=new3 mean=meanz;
run;


endrsubmit;


/*-------------------------------------------------------------------------*/
/* This macro issues waitfors and signoffs for our sessions.               */
/*-------------------------------------------------------------------------*/
%macro scagrid_waitfors(count);
  %do i = 1 %to &count;
    waitfor sess&i;
```

```
    signoff sess&i;
  %end;
%mend scagrid_waitfors;

/*-------------------------------------------------------------------------------*/
/* Wait for and sign off all sessions.                                           */
/*-------------------------------------------------------------------------------*/
%scagrid_waitfors(&SCAGRID_SESSIONS);

/*-------------------------------------------------------------------------------*/
/* Termination for our  SCAGRID functions.                                       */
/*-------------------------------------------------------------------------------*/
proc scaproc; shutdown; run;

/*-------------------------------------------------------------------------------*/
/* All done.                                                                     */
/*-------------------------------------------------------------------------------*/
```

MORE HINTS AND TIPS FOR GRID ENABLING PROGRAMS WITH PROC SCAPROC

PROC SCAPROC is a great way to get started on grid enabling a SAS program, but it's not always going to give you 100% of the benefit that a manual rewrite could give you. If a good SAS programmer takes the time to review a complex SAS program and re-works it to run on the grid, the results are often going to be better and more efficient than the results you can obtain with this procedure alone. PROC SCAPROC is new in SAS 9.2 and more features will be added to it in subsequent releases. Even with enhanced capabilities, an automated procedure like this simply cannot anticipate and account for every single thing a SAS program can do. What we have tried to accomplish with PROC SCAPROC is to get you most of the way there in terms grid-enabling legacy programs. The best approach is to use PROC SCAPROC and then look through the results for potential improvements you can make yourself. Here are a few things to keep in mind when using PROC SCAPROC and evaluating the results.

PROC SCAPROC will not "reorganize" code logically in order to put related code into the same remote submit block. For example, what if our program above had a second DATA step that creates data set b and one or more of the PROC SUMMARY steps executed against DATA step b instead of DATA step a? For example:

```
options user='c:\temp';

proc scaproc;
 record 'out2.txt' grid 'out2.grid';

run;

data a;
  attrib x label="xnum" format = 1.;
  input x y z ;
  cards;
1 2 3
3 4 3
1 2 2
4 5 6
run;

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

data b;
  attrib n label="nop" format = 1.;
  input n o p ;
  cards;
1 2 3
3 4 3
```

```
1 2 2
4 5 6
run;

proc summary data=b;
  var o;
  output out=new3 mean=omean;
run;

proc summary data=a;
  var y;
  output out=new2 mean=meany;
run;

proc scaproc; write; run;
```

The results file generated by this program will properly indicate that there are five tasks and all five can be remote submitted. PROC SCAPROC will have detected that Task 2 is dependent on Task 1 as before. It will also determine that Task 4 (PROC SUMMARY using data set b) is dependent on Task 3 (the DATA step that created data set b) and that the last PROC SUMMARY in Task 5 depends on Task 1. So the output is correct and the program will run on a grid, but if you were re-writing this program yourself, you might re-order the tasks. For example, you could put the two DATA steps first. All the other tasks are dependent on one of them and have to wait for one of them to run, so the sooner they run, the sooner the others can start executing. With PROC SCAPROC, all the tasks are submitted sequentially. Since Task 2 has to wait for Task 1, nothing after Task 2 will be submitted until after Task 2 is submitted, even though Task 3 for example (the second DATA step) is independent of Tasks 1 and 2 and could be running while Task 2 waits. SAS is doing research on the possibility of enhancing PROC SCAPROC in a future release to do some re-ordering of tasks based on dependencies, but as of SAS 9.2 this is something a programmer will want to do manually, if appropriate.

Another thing to be aware of in this initial release of PROC SCAPROC is that it does not fully expand any macros that might be invoked in your existing program. This is something that will change in a future release with the addition of an “expand macros” option, but in SAS 9.2 code encapsulated in macros is treated as a black box and the code generated by the macro is not split into separate tasks even if the generated code consists of multiple DATA and PROC steps.

PROC SCAPROC gathers its information at run time, so the information gathered is a snapshot of the conditions at the point in time when the program was run. If your program is data driven, that is, if data or macro variable values at run time determine what code is executed, then only the code actually generated and executed within the PROC SCAPROC process will be analyzed. If you are using macro programming or any other approaches to generate data driven programs, then you might need to do some manual work in order to “generalize” the results of PROC SCAPROC.

CONCLUSION

The SAS Code Analyzer is an easy to use tool that helps automate the process of analyzing program flows and generating grid enabled programs. When used with the SAS Data Integration Studio or SAS Enterprise Guide, it provides an easy way to move existing programs into a managed environment. PROC SCAPROC creates results files of executable SAS code and automatically inserts the necessary syntax for grid enablement, which helps limit potential programming errors due to manual modification and accelerates the learning curve for creating parallel programs. PROC SCAPROC can get you started quickly on the road to modernizing your SAS code and moving to a grid environment.

REFERENCES

Thies, Eric, and Rick Langston. 2008. "Introducing the SAS® Code Analyzer." *Proceedings of the SAS Global Forum 2008 Conference*. Cary, NC: SAS Institute Inc. Available at http://www2.sas.com/proceedings/forum2008/006-2008.pdf.

Doninger, Cheryl, and Nancy Rausch. 2009. "Data Integration in a Grid-Enabled Environment." *Proceedings of the SAS Global Forum 2009 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings09/098-2009.pdf.

Stander, Jeff. 2009. “For Base SAS® Users: Welcome to SAS® Data Integration!”*Proceedings of the SAS Global Forum 2009 Conference*. Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings09/092-2009.pdf.

SAS Institute Inc. 2009. *Base SAS® 9.2 Procedures Guide.* Cary, NC: SAS Institute Inc. Available at http://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/a003199742.htm.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author:

Merry Rabb
SAS Campus Drive
SAS Institute Inc.
Work Phone: 919-531-7042
Fax: 919-677-4444
E-mail: Merry.Rabb@sas.com