Paper 4949-2020

The Hack-a-saurus Guide to SAS® Grid: Saving Time with Parallel Database Queries

Ted D. Williams, PharmD, BCPS, Magellan Method

ABSTRACT

Have you ever worked overtime or missed a deadline because it took hours (even days) to extract the data you needed from multiple sources? And heaven help us if there is a flaw in the query, a change in data structures, new customer requirements, or a execution error after 12 hours of runtime. But a new dawn has broken; SAS® Grid supports parallel processing. SAS Grid enables a single program to query multiple data sources in parallel making retrieval and aggregation as simple as F3. Unlike other parallel processing solutions, SAS Grid does not require learning a completely new syntax (I'm looking at you, DS2). Unfortunately, it is not a simple as standing up a SAS Grid server and running Enterprise Guide 'Analyze for Grid'. Do you live in the WORK library? Sorry, no longer shared. Like global variables? Too bad, those break too. Do you use macros? Tough luck, SAS Grid doesn't know about your complied macros. But do not despair! All these issues can be addressed to without massive rewriting. This paper will save you those tears and overtime hours by giving you the solutions to these problems. Reading this paper is like getting into a prestigious university without your parents having to bribe administrators!

INTRODUCTION

Extraction, Transformation, and Loading (ETL) of data can be one of the most time-intensive tasks for a data analysis project. This is particularly true for projects involving multiple data sources that need to be merged or stacked together. Often retrieving each dataset is an independent task. Because SAS can only execute one command at a time, the data extraction must be run sequentially, one after the other. Multiple SAS instances can be executed manually to collect the data, but babysitting these processes is a poor use of human capital and susceptible to mistakes. A fully automated parallel process is possible without completely rewriting your code using SAS Grid. This paper discusses a few tips and tricks for the minor rewriting and troubleshooting required.

SERIAL PROCESSING OF INDEPENDENT TASKS

Traditional ETL with SAS involved sequential steps, such as:

```
*A macro library with DataSource macros;
%INCLUDE '\\MyNetwork\SASGF_2020_LIB.sas';

%MACRO GetSerialData(Results=);
   PROC SQL;
      create table &Results. (Source CHAR(100),Value CHAR(1));
   QUIT;
   %DataSourceOne(Results=&Results.);
   %DataSourceTwo(Results=&Results.);
   %DataSourceThree(Results=&Results.);
%mend;

%GetSerialData(Results=WORK.Serial);
```

The macro *%GetSerialData* creates a dataset then calls a series of macros, stacking the data each data source into a single dataset (WORK.Serial). Each of the data source macros are independent, except for the shared destination. It is the independent nature of these tasks which makes them ideal for parallelization.

GRID BASICS

The mechanism for spawning parallel processes (aka remote submissions) using SAS Grid can be condensed to:

```
%let RC = %SYSFUNC(grdsvc_enable(_all_,Server=SASApp));
OPTIONS AUTOSIGNON;

RSUBMIT MySes WAIT=NO;
   %put NOTE: Server Host: &SysHostName.;
ENDRSUBMIT;

WaitFor _all_ ;
Signoff _all_;
```

The command grdsvc_enable enables the grid service and returns a value indicating the status of the grid.[1] A return code of zero indicates the grid has been enabled and is ready for a new remote submission. Other return codes indicate various errors. OPTIONS AUTOSIGNON allows SAS to automatically sign on to the grid server when a job is submitted.[2] A manual sign on is possible and allows for greater control of the remote submission. The remote submission is bound by the RSUBMIT and ENDRSUBMIT statements. At a minimum, RSUBMIT requires the name of the remote submission (limited to 8 characters). WAIT=NO indicates the parent process can continue while the SAS server execute the submission (i.e., runs in parallel).[3] The WaitFor _all_ statement is where SAS execution stops until all remote sessions has been completed.[2] Finally, Signoff _all_ closes all the connections to the grid server.

If this code was all that it took to run SAS in parallel in a production environment there would be no need for this paper. There are a number of challenges that must be addressed in order to transform %GetSerialData into %GetParallelData:

1. SAS grid is not always available
2. SAS datasets have no mechanism for managing parallel dataset access
3. The WORK library is not shared with spawned sessions
4. Complied macros are not available to spawned sessions
5. Macro variables are not shared with spawned sessions

OVERCOMING SAS® GRID CHALLENGES

MAKING SURE THE GRID IS AVAILABLE

In the real world, Grid servers are not always available for parallel processing. But that does not mean your program is doomed to fail. If the Grid is not available, simply use the traditional serial approach:

```
%MACRO GetParallelData(Results=);

   PROC SQL;
      create table &Results. (Source CHAR(100),Value CHAR(1));
   QUIT;

   %IF %SYSFUNC(grdsvc_enable(_all_,Server=SASApp)) NE 0 %THEN
   %DO;
      %PUT WARNING: no grid, no worries, run macros in series;
      %DataSourceOne(Results=&Results.);
      %DataSourceTwo(Results=&Results.);
      %DataSourceThree(Results=&Results.);
   %END;
   %ELSE
   %DO;
   /*More to come...*/
   %END;
%MEND;
%GetParallelData(Results=WORK.Parallel);
```

MANAGING PARALLEL SESSIONS

SAS does not like multiple processes working on a single dataset. "Does not like" means SAS will generate an error and fail. In this example, all the ETL macros update a single dataset. To prevent concurrency errors, a dedicated temporary results dataset must be created for each remote submission. After all remote submissions have executed the temporary datasets can be stacked together without conflicts.

But where will the temporary datasets reside? Unfortunately, the usual destination, the WORK library, is NOT shared among remote submissions. Simply using a local drive (e.g. C:\ ) will not work since most GRID environments will have multiple physical servers. A shared network location which is visible to all grid servers can be used. If the same ETL macro is being run simultaneously by multiple clients, the issue of simultaneous dataset edits can also occur on a shared network location. Addressing these concurrency issues requires creating a unique, accessible location for each remote submission and mapping it to a LibName. This functionality can easily be wrapped into a macro:

```
%SafeSpace(NewLib=, ParentLib= );
```

The parameter `ParentLib` should contain a existing LibName mapped to a shared network location, such as:

```
LIBNAME Sandbox '\\MyNetwork\';
```

The parameter `NewLib` will be used to create a new, unique network location and SAS library. The details of the *SafeSpace* macro declaration are included in the appendix. With a dedicated library location, a temporary dataset can be created for each remote submission. These individual temporary datasets will be merged after all parallel processing is completed.

Using this concurrency management strategy, the `GetParallelData` macro definition becomes:

```
%MACRO GetParallelData(WorkingLib=, Results= );

   PROC SQL;
      create table &Results. (Source CHAR(100),Value CHAR(1));
   QUIT;

   %IF %SYSFUNC(grdsvc_enable(_all_,Server=SASApp)) NE 0 %THEN
   %DO;
      %PUT WARNING: no grid, no worries, run macros in series;
      %DataSourceOne(Results=&Results.);
      %DataSourceTwo(Results=&Results.);
      %DataSourceThree(Results=&Results.);
   %END;
   %ELSE
   %DO;
      %PUT WARNING: Grid Available, run macros in parallel;
      *Make sure the WORK-ing directory us usable by grid;
      %IF %UPCASE(&WorkingLib.) = WORK %THEN
      %DO;
         %PUT ERROR: WORK is not shared between RSUBMITs.;
         %GOTO Finish;
      %END;
      *Create a temp library for each remote submission;
      %SafeSpace(NewLib=Temp1,ParentLib=&WorkingLib.);
      %SafeSpace(NewLib=Temp2,ParentLib=&WorkingLib.);
      %SafeSpace(NewLib=Temp3,ParentLib=&WorkingLib.);

      *Make blank copies Results in each temp library;
      PROC SQL;
         CREATE TABLE Temp1.Results AS SELECT * FROM &Results.;
         CREATE TABLE Temp2.Results AS SELECT * FROM &Results.;
         CREATE TABLE Temp3.Results AS SELECT * FROM &Results.;
      QUIT;
      /*More to come...*/
   %END;
%Finish:
%MEND;

LIBNAME Sandbox '\\MyNetwork\';

%GetParallelData(WorkingLib=SANDBOX, Results=WORK.Parallel);
```

SHARING OF LIBRARIES, MACROS, AND MACRO VARIABLES ACROSS SESSIONS

No variables, libraries, or macro definitions are automatically passed to remote submissions. Think of it as a completely new instance of SAS Enterprise Guide. Fortunately, there are mechanisms to resolve these issues.

%SYSLPUT can be used to pass variables to the remote session.[4] The basic syntax is:

```
%SYSLPUT <variables to pass> / REMOTE = <Submission Name>;
```

The first parameter <variables to pass> supports several options including _USER_, _GLOBAL_ and _LOCAL_. _USER_ can produce some odd results when using nested macro.

LOCAL transfers only locally defined macro variables and makes debugging much easier. REMOTE =requires the name of the remote submission, as listed in the RSUBMIT statement.

SAS does not provide any mechanism for transferring complied macros to a remote submissions. The %INCLUDE statement can be used to execute a SAS program in the remote submission.[6] Therefore, any macros the remote submission needs to run must be contained within an included file. For this example, the macro definitions for %*DataSourceOne*, %*DataSourceOne*, %*DataSourceOne* are all contained in the \\MyNetwork\SASGF_2020_LIB.sas SAS source file. After executing the %INCLUDE statement within the RSUBMIT block, the macros are available in the remote session.

The INHERITLIB =(<library name>) clause of the RSUBMIT statement transfers a LIBNAME definition to the remote session.[5] Beware that using INHERITLIB with a LIBNAME mapped to an OLEDB connection will clear the *parent* LIBNAME. This can break a lot of downstream code. LIBNAMEs mapped to file system locations do not have this issue. This problem can be corrected by the SAS system administrators add the NOCONNECTEVENTS option to the workspace server's usermod config file. To avoid these issues, a LIBNAME statement can be used within the RSUBMIT block, similar to the %INCLUDE discussed above.

Putting all of these techniques together:

```
OPTIONS AUTOSIGNON;

%SYSLPUT _LOCAL_ / REMOTE = Ses1;
RSUBMIT Ses1 WAIT=NO INHERITLIB=(Temp1);
   %INCLUDE '\\MyNetwork\SASGF_2020_LIB.sas';
   %DataSourceOne(Results=Temp1.Results);
ENDRSUBMIT;

%SYSLPUT _LOCAL_ / REMOTE = Ses2;
RSUBMIT Ses2 WAIT=NO INHERITLIB=(Temp2)     ;
   %INCLUDE '\\MyNetwork\SASGF_2020_LIB.sas';
   %DataSourceTwo(Results=Temp2.Results);
ENDRSUBMIT;

%SYSLPUT _LOCAL_ / REMOTE = Ses3;
RSUBMIT Ses3 WAIT=NO INHERITLIB=(Temp3)     ;
   %INCLUDE '\\MyNetwork\SASGF_2020_LIB.sas';
   %DataSourceThree(Results=Temp3.Results);
ENDRSUBMIT;
```

MERGING PARALLEL RESULTS

The final steps are to wait for all parallel threads to finish, closing all severs sessions, and merging the temporary files:

```
WaitFor _all_ ;
Signoff _all_;

DATA &Results.;
   SET Temp1.Results
       Temp2.Results
       Temp3.Results
;
RUN;
```

CONCLUSION

Converting existing code to parallel processing using SAS Grid is not free, but can dramatically reduce processing time and be a worthwhile investment. Enabling parallel processing on the Grid (grdsvc_enable) and spawning remote sessions (RSUBMIT) are straightforward. Passing variables and libraries to remote sessions are more complex but are possible using intrinsic SAS features (%SYSLPUT, INHERITLIB). Executing macros and merging datasets without causing errors are possible but requires minor rewriting. Because of these complexities, using SAS grid should be consider in cases when the parallel processes are independent and can be easily synchronized.

APPENDIX

SASGF_2020_LIB.SAS

```
%Macro IsNumeric(VarToCheck);
   %local Result;
   %let Result = %eval(not %sysfunc(verify(&VarToCheck.,-0123456789)));
&Result
%mend;

%Macro LibraryExist(lb);
   %local Result;
   %Let Result=-1;

   %IF %LENGTH(&lb)=0 %Then
      %Let Result=0;
   %else
   %do;
      %let Result = %sysfunc(LibRef(&lb));
      %if &Result = 0 %then
         %let Result = 1;
      %Else
         %let Result = 0;
   %end;
&Result %mend;

%macro SafeSpace(ParentLib=, NewLib=);

   %local ParentPath;
   %Local CreatedPath;
   %local UniqueName;
   %local Leader;
   %let Leader=;
   %let UniqueName =;
   %let ParentPath=;
   %LET CreatedPath=;

   /*Default to work if no library was provided*/
   %IF %LENGTH(&ParentLib.)=0 %THEN
      %LET ParentLib = WORK;

   %if %SUBSTR(%IsNumeric(&NewLib.),1,1) = 1 %then
   %do;
      %PUT ERROR: NewLib cannot start with a numeric character *&LibName.*;
      %goto Finish;
   %end;
```

```
   %IF %length(&NewLib.)>8 %THEN
   %do;
      %PUT ERROR: NewLib must be <=8 characters &NewLib.*=%length(&NewLib.);
      %goto Finish;
   %end;

   *Get the network path for the Existing Parent Library;
   proc sql noprint;
      SELECT DISTINCT path
      INTO :ParentPath
      FROM sashelp.vLibNam
      WHERE UPCASE(libname)=UPCASE("&ParentLib.")
      ;
   quit;

   %If %Length(&ParentPath.)=0 %THEN
   %do;
      %PUT ERROR: Cannot find path for *&ParentLib.* in sashelp.vLibNam;
      %Goto Finish;
   %end;

   *Create both a new file sub directory and a new library in sas
    with a unique locations (as defined by macro %GetUniqueName();

   *Get a unique name based on the server, calling macro and a datestamp;
   %let UniqueName =UPCASE(&SYSHOSTNAME.)_%SUBSTR(%Sysfunc(datetime()),1,10);
   *Limit length to 32;
   %let Result=%Right(&UniqueName.,32);

   *If name starts with a number, replace with a letter;
   %let Leader = %substr(&UniqueName.,1,1);
   %if %IsNumeric(&Leader.) = 1 %THEN
   %do;
      %let UniqueName = %SUBSTR(&UniqueName.,2);
      %LET UniqueName=A&UniqueName.;
   %end;

   %let CreatedPath = %SYSFUNC(TRIM(&ParentPath.))\&UniqueName.&NewLib.;
   options dlcreatedir;
   libname &NewLib. "&CreatedPath.";
   options NODLCREATEDIR;

   *Just in case this file system folder was used before,
    destroy anything in that folder.;
   PROC DATASETS LIB=&NewLib. NOLIST NOWARN KILL;
   RUN;

%Finish:
%mend;

%PUT Data Access Simulation Macros;
%MACRO DataSourceOne(Results=);
   *Some dummy data, replace with database queries;
   proc sql;
      Insert into &Results. VALUES("&SysMacroName.",'A');
      Insert into &Results. VALUES("&SysMacroName.",'B');
      Insert into &Results. VALUES("&SysMacroName.",'C');
```

```
      Insert into &Results. VALUES("&SysMacroName.",'D');
      Insert into &Results. VALUES("&SysMacroName.",'E');
   quit;

%MEND;
%MACRO DataSourceTwo(Results=);
   proc sql;
      Insert into &Results. VALUES("&SysMacroName.",'A');
      Insert into &Results. VALUES("&SysMacroName.",'B');
      Insert into &Results. VALUES("&SysMacroName.",'C');
      Insert into &Results. VALUES("&SysMacroName.",'D');
      Insert into &Results. VALUES("&SysMacroName.",'E');
   quit;

%MEND;
%MACRO DataSourceThree(Results=);
   proc sql;
      Insert into &Results. VALUES("&SysMacroName.",'A');
      Insert into &Results. VALUES("&SysMacroName.",'B');
      Insert into &Results. VALUES("&SysMacroName.",'C');
      Insert into &Results. VALUES("&SysMacroName.",'D');
      Insert into &Results. VALUES("&SysMacroName.",'E');
   quit;

%MEND;
%put NOTE: Done loading macro library;
```

SASGF_2020.SAS

```
*A macro library;
%INCLUDE '\\MyNetwork\SASGF_2020_LIB.sas';
options mprint;

%MACRO GetSerialData(Results=);

   PROC SQL;
      create table &Results. (Source CHAR(100),Value CHAR(1))   ;
   QUIT;

   %DataSourceOne(Results=&Results.);
   %DataSourceTwo(Results=&Results.);
   %DataSourceThree(Results=&Results.);

%mend;

%MACRO GetParallelData(WorkingLib=,Results=);

   *CalledBy used to create a unique temporary WORK-ing library;
   %LOCAL CalledBy;
   %LET CalledBy=&SysMacroName.;

   *Make sure the WORK-ing directory provided;
   %IF %LibraryExist(&WorkingLib.)=0 %THEN
   %DO;
      %PUT ERROR: WorkingLib *&WorkingLib.* does not exist.;
```

```
   %GOTO Finish;
%END;

*Create the final Results dataset;
PROC SQL;
   create table &Results. (Source CHAR(100),Value CHAR(1))   ;
QUIT;

*Check Grid Status (zero is operational );
%IF %SYSFUNC(grdsvc_enable(_all_,Server=SASApp)) NE 0 %THEN
%DO;
   %PUT WARNING: no grid, no worries, run macros in serial;
   %DataSourceOne(Results=&Results.);
   %DataSourceTwo(Results=&Results.);
   %DataSourceThree(Results=&Results.);
%END;
%ELSE
%DO;
   %PUT WARNING: Grid Available, run macros in parallel;

   *Make sure the WORK-ing directory us usable by grid   ;
   %IF %UPCASE(&WorkingLib.) = WORK %THEN
   %DO;
      %PUT ERROR:  Grid servers cannot use WORK.;
      %goto Finish;
   %END;

   %SafeSpace(NewLib=Temp1,ParentLib=&WorkingLib.);
   %SafeSpace(NewLib=Temp2,ParentLib=&WorkingLib.);
   %SafeSpace(NewLib=Temp3,ParentLib=&WorkingLib.);

   *Make blank copies Results in each temp library;
   PROC SQL;
      CREATE TABLE Temp1.Results AS SELECT * FROM &Results. ;
      CREATE TABLE Temp2.Results AS SELECT * FROM &Results. ;
      CREATE TABLE Temp3.Results AS SELECT * FROM &Results. ;
   QUIT;

   *Use automatic sign on for simplicity;
   OPTIONS AUTOSIGNON;
   ************************
   Create First Grid Session: Ses1
   ************************;

   *Use SYSLPUT Inherit variables into Ses1;
   %SYSLPUT _LOCAL_ / REMOTE = Ses1;

   RSUBMIT Ses1
      WAIT=NO           /*enables parallel processing*/
      INHERITLIB=(Temp1)/*specifies SAS libraries to inherit*/
   ;
      *Between RSUBMIT and ENDSUBMIT do everything you
      would do for a new program.;
```

```
         *include reference to macro library;
         %INCLUDE '\\MyNetwork\SASGF_2020_LIB.sas';
         *Run the macro;
         %DataSourceOne(Results=Temp1.Results);
      ENDRSUBMIT;

      ************
      Repeat for session 2
      *************;
      %SYSLPUT _LOCAL_ / REMOTE = Ses2;
      RSUBMIT Ses2 WAIT=NO INHERITLIB=(Temp2) ;
         %INCLUDE '\\MyNetwork\SASGF_2020_LIB.sas';
         %DataSourceTwo(Results=Temp2.Results);
         %put NOTE: Server Host: &syshostname.;
      ENDRSUBMIT;

      ************
      Repeat for session 3
      *************;
      %SYSLPUT _LOCAL_ / REMOTE = Ses3;
      RSUBMIT Ses3 WAIT=NO INHERITLIB=(Temp3) ;
         %INCLUDE '\\MyNetwork\SASGF_2020_LIB.sas';
         %DataSourceThree(Results=Temp3.Results);
         %put NOTE: Server Host: &syshostname.;
      ENDRSUBMIT;

      *Syncronize parallel processes;
      WaitFor _all_ ;
      *Sign off to free up grid resources;
      Signoff _all_;

      *Now merge parallel results without fear of locking;
      DATA &Results.;
         SET Temp1.Results
             Temp2.Results
             Temp3.Results
         ;
      RUN;

      *A bit of clean up;
      PROC DATASETS LIB=Temp1 NOLIST NOWARN KILL; RUN;
      PROC DATASETS LIB=Temp2 NOLIST NOWARN KILL; RUN;
      PROC DATASETS LIB=Temp3 NOLIST NOWARN KILL; RUN;

   %END;

%Finish:
%MEND;


%PUT;
%GetSerialData(Results=WORK.Serial);
```

```
Title Serial Results;
PROC SQL OUTOBS = 100;
   SELECT * FROM Serial;
QUIT;

LIBNAME Sandbox '\\MyNetwork\';
*
Note that the only difference between the calls is the
definition of a local variable for a new WORK-ing library
;
%GetParallelData(Results=WORK.Parallel,WorkingLib=SANDBOX);

Title Parallel Results;
PROC SQL OUTOBS = 100;
   SELECT * FROM Parallel;
QUIT;
```

REFERENCES

1. SAS Help Center: GRDSVC_ENABLE Function. https://documentation.sas.com/?docsetId=gridref&docsetTarget=n1x4fhd14au95fn1hhcknbvtpn1b.htm&docsetVersion=9.4&locale=en. Accessed January 29, 2020.

2. Quick-Start to SAS(R) Programming on the Grid. Presented at the: SAS Global Forum 2019; March 25, 2019; Dallas, Tx.

3. Jacobsen G, Lavery R. A Parallel Processing Primer. Presented at the: NorthEast SAS Users Group; 2017.

4. %SYSLPUT Statement - 9.3. http://support.sas.com/documentation/cdl/en/connref/63066/HTML/default/viewer.htm#n1bc4zi0lp02ivn1oxn9gc1bsaha.htm. Accessed January 29, 2020.

5. Syntax for the RSUBMIT Statement and Command: RSUBMIT Statement and Command. http://support.sas.com/documentation/cdl/en/connref/61908/HTML/default/viewer.htm#a002590433.htm. Accessed January 29, 2020.

6. SAS Help Center: %INCLUDE Statement. https://documentation.sas.com/?docsetId=lestmtsref&docsetTarget=p1s3uhhqtscz2sn1otiatbovfn1t.htm&docsetVersion=1.0&locale=en. Accessed January 29, 2020.

ACKNOWLEDGMENTS

I would like to thank all of the Magellan Health SAS users who supported me while I refined these techniques. Special thanks go out to Scott Smith at Magellan Health who answered my incessant questions and without who's help these technique would have been unachievable. Additionally, I would like to thank XXX, XXX, XXX, and XXX for their editorial assistance.

CONTACT INFORMATION

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

Ted D. Williams, PharmD, BCPS
Senior Clinical Program Manager / Health Economics & Outcomes Research (HEOR)
Magellan Health / Magellan Method
tdwilliams1@magellanhealth.com