PharmaSUG 2017 - Paper BB11

SAS® Programmer's Guide to Life on the SAS Grid

Eric C. Brinsfield, Meridian Analytics

ABSTRACT

With the goal of utilizing computing power and support staff more efficiently, many organizations are moving all or large portions of their SAS® computing to a SAS Grid platform using SAS Grid Manager. This often forces many SAS programmers to move from Windows to Linux, from local computing to server computing, and from personal resources to shared resources. This shift can be accomplished if you make slight changes in behavior, practice and expectation. This presentation offers many suggestions for not only adapting to the SAS Grid but taking advantage of the parallel processing for longer running programs.

Sample programs are provided to demonstrate best practices for developing programs on the SAS Grid. Program optimization and program performance analysis techniques are discussed in detail.

INTRODUCTION

As SAS programmers, you may be switching to SAS Grid platform because you have a specific need for faster throughput or because your organization makes a strategic decision to centralize all SAS processing. You may be moving from running SAS on one operating system to the SAS Grid that is installed on a different operating system. For the purposes of this presentation, I will focus on one very common scenario, but I limit the discussion to differences due to the SAS Grid issues rather than issues related to operating system changes.

USE CASE: SAS GRID INSTALLED ON LINUX; ALL OTHER SAS REMOVED

Many large organizations move to a centralized SAS Grid processing model such that SAS software is only available on the SAS Grid. In this case study, SAS Grid is installed on Linux forcing many Windows users to work in a Linux environment. Additionally, in order to ease the installation and maintenance of SAS clients, most users are encouraged to access SAS Grid via Citrix or some other remote desktop application. The implication is that SAS clients do not have SAS installed in the client environment. All SAS processing is handled remotely on the SAS Grid.

Although not discussed in this presentation, many companies use other configurations including those where users still retain their own copy of SAS software and the SAS Grid provides an option for more intensive processing and analysis. In addition, the user's version of SAS could be running on a different operating system than the SAS Grid. All of these work fine but are different use cases.

JUSTIFICATION

When you analyze the number of dispersed and disparate SAS licenses across a large organization, you find that your total licensing fees are high, many licenses sit idle for 80% of a day or more, and users do not have access to many of the SAS products that they want to use. In addition, individuals or departments are responsible for installation and maintenance, which distracts programmers and statisticians from their purpose.

With centralization on a SAS Grid platform,

- Total licensing costs go down and SAS product inventory control goes up
- Utilization of available SAS computing power goes up
- Support can be centralized or centrally managed with trained specialists
- Users have access to more SAS products and capabilities
- Where needed, users can reduce overall run-times for demanding long-running programs

Of course, downsides do exist, such as:

- Downtimes for maintenance and installations have a more global impact. (In other words, every change to the system becomes a big deal)
- Technical support could be slower depending on the complexity of the support process
- Network speeds could impose a slow response time to the users located far from the SAS Grid
- Users are sharing the system with others, which can impact performance at peak hours and require users to share disk space.

All of these negatives, however, can be handled.

HIGH LEVEL IMPLICATIONS

If you perform an Internet search for "SAS Grid", you see many explanations of why you need SAS Grid Manager, how it is architected, and how it deals with big data and parallel processing. For the average SAS programmer who is being "encouraged" to work only on the SAS Grid, most of this is irrelevant. What you want to know is: how your work will change?

Fortunately, unless you need to run blocks of code in parallel, your work process will change very little. You don't need to know all the SAS Grid functions and options or about the multiple options for submitting programs to the grid. You may need to learn a new SAS client, which I discuss later, but the key is that your SAS client session connects to the SAS Grid and all of your programs execute on the SAS Grid automatically. In other words, programming in SAS changes very little when running on the SAS Grid. The rest of this discussion focuses on the key issues to consider when using a remote SAS client that executes SAS on the SAS Grid.

Next and at a high-level, I address some key concepts used in this discussion and also briefly point out some Windows-to-Linux considerations for SAS programmers.

AVAILABLE SAS CLIENTS

In this use case scenario, users do not have Foundation SAS installed on the client's system, so the default SAS windowing environment (Display Manager) cannot run in the remote Windows environment. You can run Display Manager on the Grid's Linux nodes, but Display Manager on Linux is not very friendly.

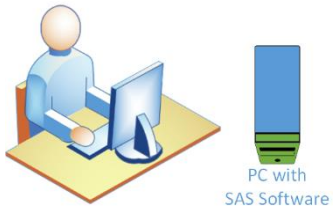
As the closest replacements for SAS Display Manager, SAS Enterprise Guide and SAS Studio work quite well. Other specialized clients such as Enterprise Miner will also work with the SAS Grid. I discuss SAS Enterprise Guide and SAS Studio in more detail below. All of these clients provide a means to set a connection to the SAS Grid servers, so that all programs are automatically sent to the SAS Grid for execution.

SERVER-BASED COMPUTING

Many organizations have been using Citrix or other remote desktop servers for years to provide centralized SAS processing. In most cases, all SAS software is installed on the remote desktop server. When users log into the remote desktop server, they are also on the SAS server.

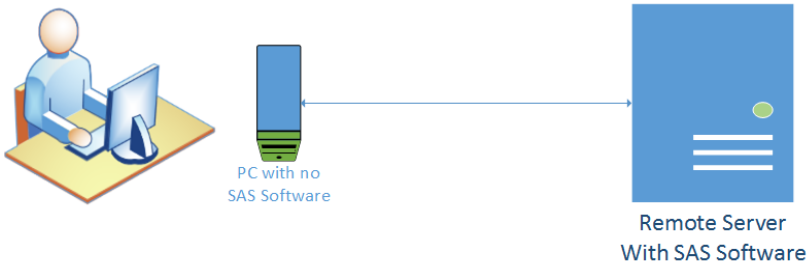
In my scenario, SAS software is not available anywhere on the remote desktop server. You must operate in client-server mode, such that your SAS client is executing on your PC or on the remote desktop, while SAS processing occurs on the SAS Grid Linux servers.

Figure 1 SAS software running on a stand-alone PC or laptop



In this diagram, SAS programs execute on the PC, which means usually that the data, the programs, the execution, and the destination for the output are on the same system. The user may be limited by the number of CPU, memory, disk space, or licensed SAS products installed on the system.

Figure 2 SAS software is installed on a server



In Figure 2 SAS is installed on the server but not on the user's PC. The user logs onto the server using a remote desktop package. In this scenario, the user is not really running SAS in server mode. Because the SAS client and the SAS processing engine all live on the same server, the remote desktop configuration simulates a very large PC or some other operating system workstation. The user will be sharing resources with other users, but this architecture does reduce costs of software, maintenance, and support. Scalability can become a problem quickly with the SAS processing or with the remote desktop software.

Figure 3 Generic SAS Grid architecture with SAS processing only on the SAS Grid servers

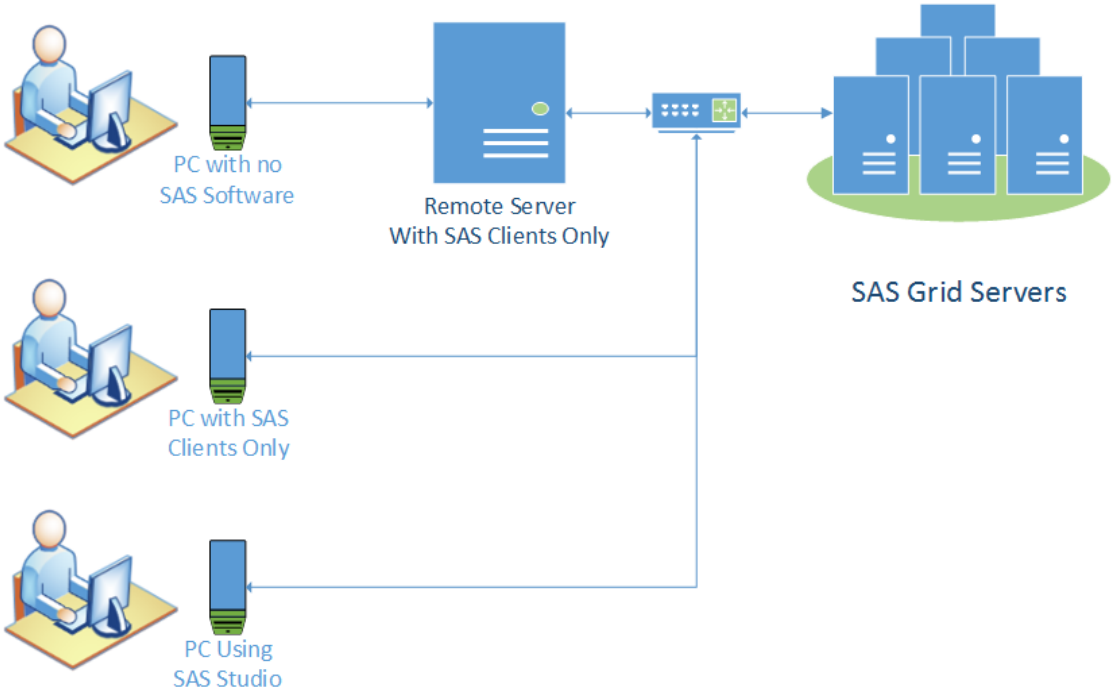


Figure 3 shows a very simplified diagram of a SAS Grid architecture. The point is to clarify what I mean by server-based computing. In this arrangement, the clients live on different computers than the SAS processing engine, the SAS Grid. The clients are on either the remote desktop server, the users' workstation, or on the Web server for those using SAS Studio via a web browser.

As a user who is creating programs, you need to be aware of these architectural factors, which I will discuss in more detail below, that could impact your program response time:

- The physical location of the SAS Grid servers in relation to the location of the remote desktop servers. Usually, the remote desktop servers are located next to the SAS Grid servers.
- Your physical location compared to the SAS Grid or the remote desktop servers. In global organizations, grid clusters may be located in one central location with users distributed all over the world. As a user, you should find out where the grid cluster is located.
- The location of the input data in relation to the SAS Grid servers
- The volume of output or the log file generated
- The destination of the output:
    - Streaming back to the client
    - Routed to another server
- Your network distance and bandwidth

All of these points will have an impact on the perceived performance of your programs in the new environment.

CHANGE IN OPERATING SYSTEM: MOVING FROM WINDOWS OR UNIX TO LINUX

Without going into great detail about the differences between Linux and Windows or Linux SAS and Windows SAS, I do want to point out a few important impacts:

Change in Scripting Language

If you like to run SAS program in batch mode on Windows, you may want to learn some Linux shell scripting so you can run jobs with more flexibility and control. In most cases, your support team will provide some options for you. On the SAS Grid, you will need to use SASGSUB instead of SAS, which I discuss below. Your Windows .bat files will not work on Linux.

Change in SAS Data Set Structure

SAS Version 9 can read data sets automatically that were created on older versions (V7+) of SAS or from other operating systems through Cross-Environment Data Access (CEDA). See *Moving and Accessing SAS® Files, Third Edition.* Depending on the originating operating system, you may experience a performance hit for this automatic conversion. For example, I have relied on CEDA to read SAS data sets created on Windows with very little performance impact, but reading data sets created on AIX have taken up to 10 times longer to read.

Consequently, if you plan to use the data sets regularly, you will want to run PROC MIGRATE to convert the data sets to Linux format. Refer to *Base SAS® 9.4 Procedure Guide*.

Loss of Some Windows Integration

If you have programs that rely on integrations to Windows such as Dynamic Data Exchange (DDE), you will need to design a new approach. DDE will not work in Linux, but you will find many ways to achieve what you need in SAS 9.4.

Change in System Commands

System commands can still be executed using X command or SYSTEM functions, but many of the actual commands may be different. You just need to evaluate your code for those types of changes.

Change in Editing and Managing Files Outside of SAS

If you are moving from Windows and you are not comfortable with Linux editors, you can always use the Windows-based editors that are available on your remote desktop or your PC. You will need your files to be either linked to the Linux storage or use an SFTP-tool to easily upload and download. In most cases, you can open the file in the SFTP tool and edit directly on your PC environment. In other words, do not let Linux scare you.

DMS Commands Not Available in Recommended Clients

As suggested above, you will likely move away from SAS Display Manager and over to SAS Enterprise Guide or SAS Studio, neither of which support DMS commands or statements. If you have code that relies on those integrations to SAS Display Manager, the statement will be ignored and may cause subsequent statements to fail or behave unexpectedly.

SAS CLIENT CONSIDERATIONS

GENERAL CONSIDERATIONS

SAS Enterprise Guide and SAS Studio connected to the SAS Grid exhibit some common behaviors that you should consider especially when running in a true server mode.

System options

Both SAS Enterprise Guide and SAS Studio will reset certain system options when they start. These changes override options set in the SAS configuration files as well as the autoexec files, so it is worth it to run PROC OPTIONS after you start so you know what you are dealing with. For example, both clients reset the value for VALIDVARNAME to ANY, which is a nice new feature, but also causes some surprises when importing data from Excel or other external sources. If you import Excel spreadsheets and your subsequent code was designed to expect specific variable names coming out of the import, you may want to reset VALIDVARNAME=V7. Refer to the *SAS® 9.4 System Options: Reference* for more information.

Minimize streaming output volume

When operating in server mode, as shown in Figure 3, your PC could be separated from the remote desktop or the SAS Grid by thousands of miles. Be cautious when you produce a large volume of output. If your program seems to run slower on the SAS Grid than when on PC SAS, you may want to look at how much output you are trying to return to your display over the network. On investigation, you may find that the program runs in seconds, but response on the SAS clients takes many minutes or even hours.

For example, if you run an analysis using a SAS statistical procedure, you may produce thousands of pages of output. By default, the SAS clients will attempt to return that output to your SAS client and display that output for you to see. Even with buffering, SAS and the remote desktop software may be moving significant volumes of data (output) behind the scenes with so many layers of servers and clients.

To circumvent these issues, I discuss options for each SAS client in the next section and additional programming steps in the section entitled Suggested Code Adjustments for SAS Grid Clients. In the performance discussion, I cover techniques for identifying performance problems.

Monitor SAS log volume

Similar to the problems with voluminous output, you may have turned on debugging options, such as MLOGIC or SYMBOLGEN that tend to write thousands of lines to the SAS log. If you experience poor responsiveness running a program, you may want to consider routing all or portions of the log to an external file. Again, in the server-based computing, you need to be aware of the amount of output (in this case the SAS log) is streaming back to your display.

Large data sets

Like SAS output and SAS logs, the SAS client may attempt to open data sets that you created with your program. Be aware.

Shift long-running jobs to batch or RSUBMIT

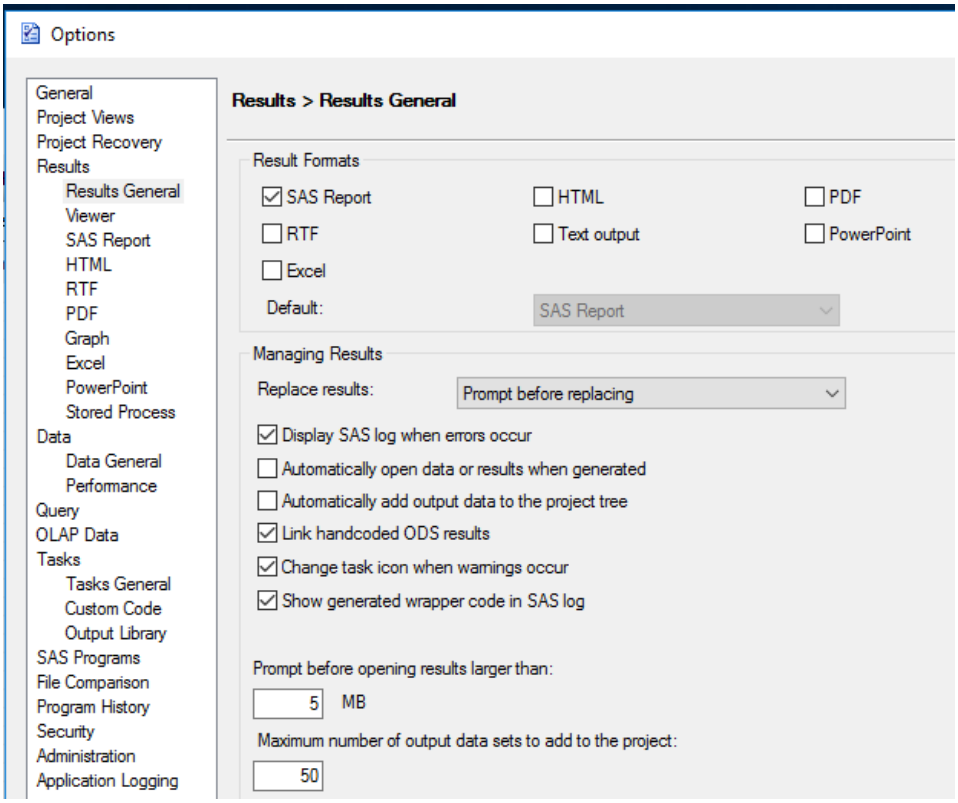
I suspect some of you have started a long-running program on your PC and headed home for the evening, hoping that the program will complete execution by the time you return in the morning. You can still do this, but most sites will discourage that technique in a shared environment. In most cases, your remote desktop session would be disconnected, but the SAS session will continue running. You can reconnect but it does create frustration for support personnel when evaluating who is on the system. Most sites would encourage you to convert these long-running program into batch jobs that can be scheduled and recognized.

If you have a program that just runs a moderately long time, say for one hour, and you do not want to bother creating a batch job, you have another option. You can run the program in the background using RSUBMIT to the SAS Grid, which can be used in both SAS Enterprise Guide and SAS Studio, thereby allowing you to keep on working while the program runs. I show examples in the section below entitled Background and Parallel Processing.

SAS ENTERPRISE GUIDE

To reduce chances of your programs performing slowly due to large resulting data or output, you should examine your SAS Enterprise Guide options. Under Options → Results → Results General, select the least number of Result Formats that you need. I also turn off "Automatically open data or results when generated". Depending on the type of programs you run, you may want to play around with these and other settings on this page. Remember, you do not want to sit there waiting for the output to load when the actual program finished an hour ago.

Figure 4 Screenshot of the options panel in SAS Enterprise Guide



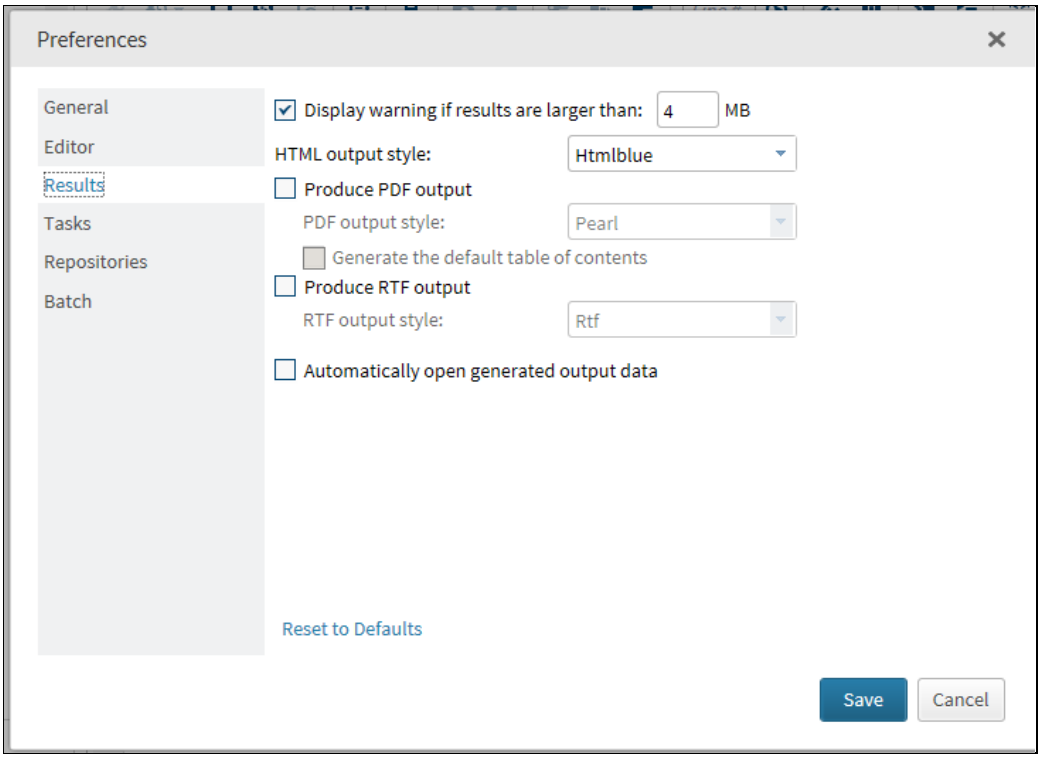
In the screenshot in Figure 4, I also show the "Show generated wrapper code in SAS log" check box. I suggest turning that on temporarily so you can see what options SAS Enterprise Guide is setting in front of your code.

I have provided a few references to other papers and presentations that make tuning recommendations for SAS Enterprise Guide. These are worth reviewing if you are new to SAS Enterprise Guide.

SAS STUDIO

SAS Studio contains a similar options window that is accessed by clicking on the "Preferences" button at the top right of the main page. If you open the Results page, you will see the options shown in Figure 5 below. I make sure that I do not have multiple forms of output being sent back to my session. I also turn off the automatic opening of generated output data, but each of these really depend on what you are generating. For small programs, it is nice to see data and results automatically.

Figure 5 Screenshot of Preferences window in SAS Studio



SUGGESTED CODE ADJUSTMENTS FOR SAS GRID CLIENTS

Minimizing Streaming Output

Expanding on the suggestion to be careful of streaming large volumes output to the SAS clients, I also take an extra step in my code if I know that output could be large. Specifically, I use PROC PRINTTO or ODS to route large output to external files stored on drives that are local to the SAS Grid servers. The SAS client will receive a response almost as soon as the processing completes and you can view the files directly on Linux. You do not have to route all of your output to a file. You can control which go where.

Using ODS

If you are using a SAS procedure or generating a report that creates a large output file, you might want to consider routing the output to an external file using ODS. Refer to *SAS® Output Delivery System: User's Guide*. For example, in either SAS Enterprise Guide or SAS Studio:

```
ODS _ALL_ CLOSE;
ODS PDF FILE="&projroot/output/myanalysis.pdf";
/*--- all of the code to generate the report */
ODS PDF CLOSE;
ODS HTML;
```

For the ODS format, you can use RTF, LISTING, or whichever format is appropriate. Refer to the They key point is that you are routing the output directly to a file system that is close to the servers rather than sending this output across the network and making the SAS client load it into your session. You can still open the file from within SAS or use another tool outside of the SAS environment.

Doing this is often inconvenient, because you want to see the output. The key is to position the ODS statement exactly where you generate the large output rather than at the top of the program, so you can still see interim results in the client interface.

Using PRINTTO

You can also use PROC PRINTTO to route your output to an external file, but I prefer using ODS because it gives you more control. The SAS log, however, is another story. I often use CALL EXECUTE that might be generating code from a large data set or from a loop, which can sometimes generate a very large SAS log if you are still debugging. The same is true for some statistical PROCEDURES or analyses. For example, I have helped some statisticians with Monte Carlo simulations that generate over 900,000 lines in the log. I am sure those are all interesting notes, but there is no value in streaming that all back to the SAS clients.

Immediately prior to the offending code, route the log to an external file with:

```
PROC PRINTTO LOG="&projroot/saslogs/mc_sim_001.log"  NEW;  RUN;
/* Insert code here that creates a large SAS log file */
PROC PRINTTO LOG=LOG; RUN;
```

The NEW option instructs the procedure to replace the LOG if it already exists rather than appending. If you need to analyze the logs afterward or you just want it all together in one place, you can always place the PRINTTO at the top of the program. This, however, takes the convenience out of using SAS Enterprise Guide or SAS Studio, so I prefer to target specific sections.

Of course you also have the ability to turn off much of the SAS log with OPTIONS NONOTES, etc., but you often want to preserve the log.

RUNNING SAS BATCH JOBS

SAS COMMAND → SASGSUB

To execute SAS on the SAS Grid, you replace the SAS command with the SASGSUB command.

```
sasgsub -gridsubmitpgm programname.sas
```

Note that on Linux, commands are case-sensitive and usually implemented in lower case. SASGSUB ties into the SAS Grid Manager that uses IBM's LSF scheduling so that your job is routed to the "most-available" SAS Grid node. For documentation refer to *Grid Computing in SAS® 9.4*. As you will see in the documentation, the SASGSUB command can be used for many other functions, including the retrieval of results. In addition, you can add SAS options to the SASGSUB command using the GRIDSASOPTS option, such that the log and output are routed where you want them. For example:

```
sasgsub -gridsubmitpgm programname.sas
-gridsasopts "'-altlog /GRIDFOLDER/yourproject/programname.log -print
/GRIDFOLDER/yourproject/programname.lst'"
```

Note that the spacing and quoting within the -gridsasopts phrase are very touchy. You should check the documentation to study all of the options that are available for SASGSUB.

SASGSUB offers several other documented options, which you may want to examine in detail. SASGSUB options that I like to use with my batch jobs are GRIDJOBNAME and GRIDWAIT, GRIDWAITNORESULTS, or GRIDWAITRESULTS. Of course, when using GRIDSASOPTS, you can also pass in the AUTOEXEC option or any other SAS option. As you start adding options, you will find that it is worthwhile to start using a custom Linux script or something developed by your support staff. In the next section, I provide sample Linux script segments.

Batch SAS with Linux Scripts

The following script called exec_one_batchjob.sh, takes one parameter: the name of the SAS program without the extension. The script assumes that your programs, logs, and output will all live under the same high level-folder called *projroot*/*yourstudyname*. Obviously, you could pass these as parameters, but you have to decide how much typing you want to do with each execution. This example calls SASGSUB and routes the log and output to the desired folders:

```
#!/bin/sh
#*** Check for correct number of parameters passed to the script ***
if [ $# -lt 1 ] ; then
echo "Please specify the SAS program name. Do not include .sas in the name."
exit 1
fi
pgmname="$1"
projroot=/GRIDFOLDER/yourproject
stdyname=yourstudyname
pgmdir=${projroot}/${stdyname}/programs
logdir=${projroot}/${stdyname}/programs
sasgsub -gridsubmitpgm ${pgmdir}/${pgmname}.sas -gridsasopts "'-altlog
${logdir}/${pgmname}.log -print ${logdir}/${pgmname}.lst'"
```

Details:

- The parameter that is passed by the user during the call is assigned to pgmname.
- projroot, studyname, pgmdir, logdir are all script parameters that are referenced above with ${*parmname*}
- -gridsubmitpgm is the SASGSUB option that identifies the program
- -gridsasopts is the SASGSUB option that signals a list of SAS options

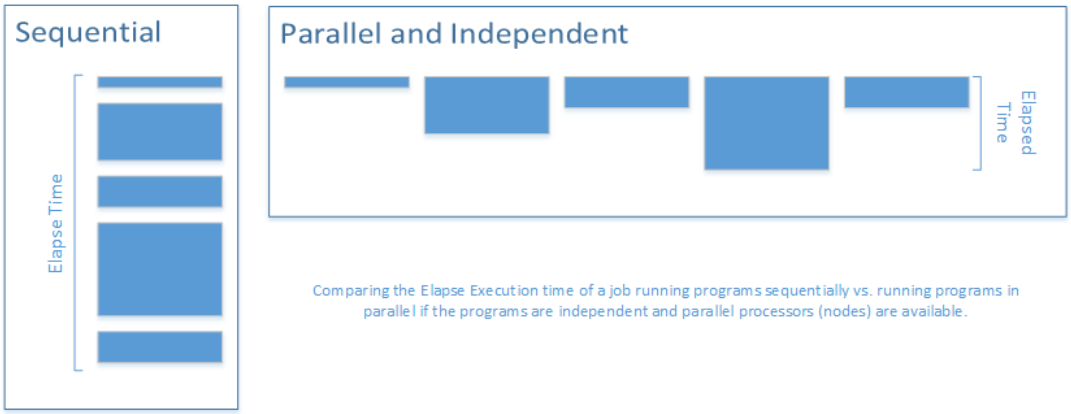
Submitting in parallel

Running in parallel means that multiple steps or programs are running at the same time, so that the overall run time to complete the set of programs is faster, as depicted in Figure 6 below. Each program will likely run in the same amount of time, but the elapsed time for the group is faster. The speed of parallel processing will depend system capabilities such as:

- Number of CPU per server
- Number of servers in a cluster
- Available memory to process multiple processes at once on the same server
- Write and read speed and pathways to the disks

Of course, you would not want to run in parallel if one program needed output from another.

Figure 6 Depiction of sequential vs. parallel program execution



If you modify the script from above by removing the parameter prompt and adding a second call to SASGSUB, you can run more than one program with the script. In this case, SASGSUB executes each call in sequence but all jobs start immediately, thereby running in parallel. The second SASGSUB does not wait for the first to finish.

```
#**** First program name
pgmname="pgm01"
sasgsub -gridsubmitpgm ${pgmdir}/${pgmname}.sas

#****  Program name
pgmname="pgm02"
sasgsub -gridsubmitpgm ${pgmdir}/${pgmname}.sas
```

Submitting in sequence

If you want the second program to wait for the completion of the first program, you should add the GRIDWAIT options as shown below for pgm01:

```
#**** First program name
pgmname="pgm01"
sasgsub -gridsubmitpgm ${pgmdir}/${pgmname}.sas -gridwait

#****  Program name
pgmname="pgm02"
sasgsub -gridsubmitpgm ${pgmdir}/${pgmname}.sas
```

Checking return codes

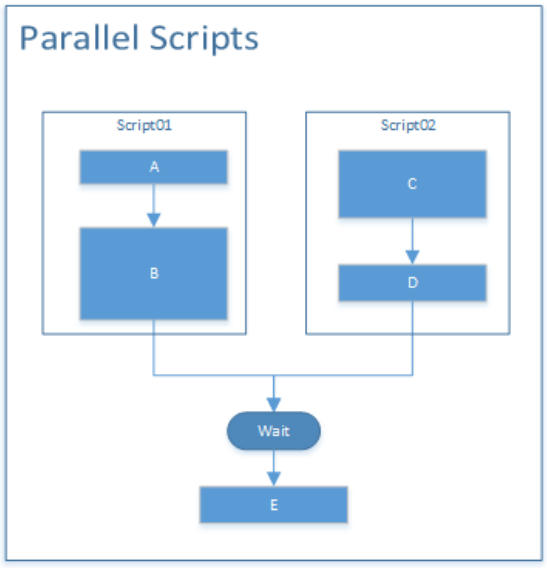
If you have a series of programs to submit, you may want to check the return code of the SAS program before you start the next job. In order to force SASGSUB to pass the return code to your script, you need to use either GRIDWAITRESULTS or GRIDWAITNORESULTS. In the example below, I use GRIDWAITNORESULTS so that SASGSUB passes a return code but does not copy the SAS log, output and SAS program back into a subdirectory within the program folder:

```
pgmname="pgm01"

sasgsub -gridsubmitpgm ${pgmdir}/${pgmname}.sas -gridwaitnoresults -
gridsasopts "'-altlog ${logdir}/${pgmname}.log -print
${logdir}/${pgmname}.lst'"

RC=$?

if [[ $RC -ge 2 ]] ; then

echo "ERROR IN JOB ${pgm_name}.sas failed"

echo "Return Code: ${RC}"

else

#****  Program name

pgmname="pgm02"

sasgsub -gridsubmitpgm ${pgmdir}/${pgmname}.sas -gridsasopts "'-altlog
${logdir}/${pgmname}.log -print ${logdir}/${pgmname}.lst'"

fi
```

Note that in the example above, the return code is retrieved with the RC=$? statement. RC can then be tested for conditional action.

Submitting parallel with dependencies

Sometimes you have collections of programs or steps that can be run in parallel but within the group, programs must run in a sequence. You may also have a final program that depends on both groups.



Each blue block represents a program. The height shows the relative execution time.

- Programs within a script must run sequentially
- Program E depends on completion of both scripts

If you want to run collections of programs in parallel but run a final program when all collections have completed successfully, you can execute separate scripts that use GRIDWAITNORESULTS within them.

When each finishes, the master script tests the return codes from each child script before executing the final program:

```
batdir=${rootdir}/batchscripts              <<< location of batchscripts
nohup ${batdir}/exec_script01.sh &
pidone=$!
nohup ${batdir}/exec_script02.sh &
pidtwo=$!
wait $pidone
rcone=$?
wait $pidtwo
rctwo=$?

if [[ $rcone -lt 2 ]]  &&  [[ $rctwo -lt 2 ]] ; then
#****  Program name
pgmname="pgm03"
sasgsub -gridsubmitpgm ${pgmdir}/${pgmname}.sas -gridsasopts "'-altlog
${logdir}/${pgmname}.log -print ${logdir}/${pgmname}.lst'"
fi
```

The NOHUP statement executes the script and disconnects until it returns. The WAIT statement causes the script to hold further processing until the values of PIDONE and PIDTWO are populated.

Using this type of script, you can optimize (minimize) the run times of a large number of independent and dependent programs. You can achieve the same results with the SAS RSUBMIT statement, but depending on your needs, you may want to run in batch. For a discussion of optimizing parallel processing of SAS programs on the SAS Grid see the reference *Brinsfield, Eric. 2016*. The paper focuses on using RSUBMIT, but could also be applied to Linux scripting.

Valuable LSF commands

SAS GRID Manager uses IBM's LSF to manage and schedule SAS processes on the SAS Grid. Consequently, you can use LSF commands to query the status of jobs. Refer to the *IBM Platform LSF Command Reference*.

bjobs

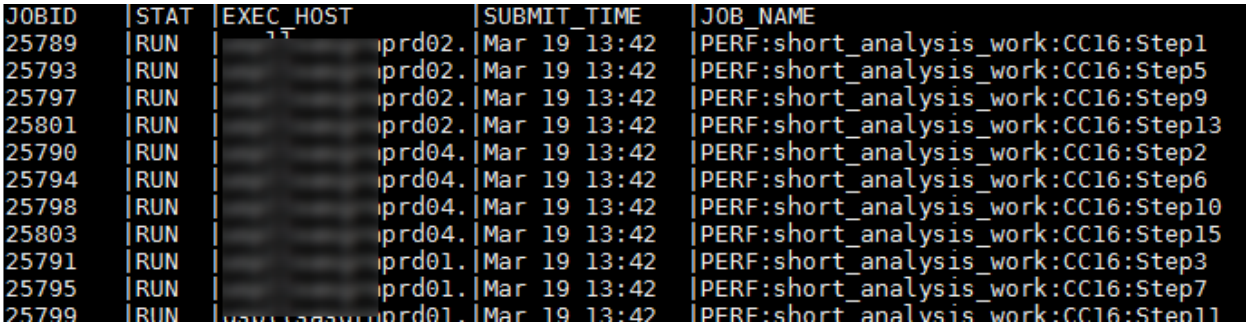
The bjobs command lists jobs that are running on the SAS Grid. Without any parameters, the command lists your jobs including any interactive sessions. If you have the appropriate permissions, you can list all jobs with:

```
bjobs -u all
```

I like to reformat the default output from the bjobs command, so I get back the information that I want. For example, I created a Linux script called myjobs that issues the following bjobs command:

```
bjobs -o "jobid:8 stat: exec_host:17 submit_time:14 job_name delimiter='|'"
```

Figure 7 Output from a customized bjobs command called myjobs



JOBID	STAT	EXEC_HOST	SUBMIT_TIME	JOB_NAME
25789	RUN	[redacted]prd02.	Mar 19 13:42	PERF:short_analysis_work:CC16:Step1
25793	RUN	[redacted]prd02.	Mar 19 13:42	PERF:short_analysis_work:CC16:Step5
25797	RUN	[redacted]prd02.	Mar 19 13:42	PERF:short_analysis_work:CC16:Step9
25801	RUN	[redacted]prd02.	Mar 19 13:42	PERF:short_analysis_work:CC16:Step13
25790	RUN	[redacted]prd04.	Mar 19 13:42	PERF:short_analysis_work:CC16:Step2
25794	RUN	[redacted]prd04.	Mar 19 13:42	PERF:short_analysis_work:CC16:Step6
25798	RUN	[redacted]prd04.	Mar 19 13:42	PERF:short_analysis_work:CC16:Step10
25803	RUN	[redacted]prd04.	Mar 19 13:42	PERF:short_analysis_work:CC16:Step15
25791	RUN	[redacted]prd01.	Mar 19 13:42	PERF:short_analysis_work:CC16:Step3
25795	RUN	[redacted]prd01.	Mar 19 13:42	PERF:short_analysis_work:CC16:Step7
25799	RUN	[redacted]prd01.	Mar 19 13:42	PERF:short_analysis_work:CC16:Step11

bkill

If you need to cancel a job on the SAS Grid, you can use the BKILL command. For example, if I wanted to cancel the job Step9 above, I would look up the JOBID and enter:

```
bkill 25797
```

PERFORMANCE CONSIDERATIONS FOR PROGRAMMERS

Based on the information I have discussed, it should be clear that performance (response time and run time) should be something you think about as you move to the SAS Grid. Of course, our expectation from the SAS Grid includes faster performance, which is true if the system is sized properly and you apply the considerations above and learn to take advantage of the SAS Grid parallel processing.

EVALUATING PERFORMANCE OF YOUR PROGRAM AND THE SAS GRID

Apparent SAS Performance

One of the most important performance measures for a system is user perception. To steal from sailing terminology, where you have "true wind speed" and "apparent wind speed", you also have "true SAS performance" and "apparent SAS performance". True SAS performance includes the elapsed time, CPU time, and memory usage on the SAS server, while the apparent SAS performance includes the true SAS performance plus the time it takes for you to see the results. Just like in sailing where apparent wind speed affects your boat speed, apparent SAS performance impacts your program response time or perceived completion time.

The reason that I spent so much time earlier discussing the routing of output and logs when running in SAS Enterprise Guide or SAS Studio is because the long transmission times for output and SAS logs create "slow apparent SAS performance".

Detecting an Apparent SAS Performance Problem

If you perceive a performance issue with your SAS programs when in SAS Studio or SAS Enterprise Guide, your first response should be look at the SAS log before you call corporate SAS technical support. After each DATA or PROC step you should see some notes such as:

```
NOTE: DATA statement used (Total process time):
      real time           0.05 seconds
      cpu time            0.03 seconds
```

If you do not see notes like this, turn them on with:

```
OPTIONS NOTES STIMER;
```

Or for more detailed information, enter:

```
OPTIONS NOTES FULLSTIMER;
```

```
NOTE: DATA statement used (Total process time):
      real time           0.02 seconds
      user cpu time       0.00 seconds
      system cpu time     0.01 seconds
      memory              344.65k
      OS Memory           8160.00k
      Timestamp           03/23/2017 05:27:35 PM
      Step Count          2  Switch Count  0
```

The actual statistics you see will depend on your operating system, but the information that you need is real time, which is the elapsed time for the program to execute on the SAS server, and CPU time, which tracks the time the program was using CPU.

If your "real time" is very low (fast) as shown above, but the responsiveness in the SAS client is very slow, say 1 hour, you have an "apparent" SAS performance problem. The SAS log is saying that the SAS processor only had the program for .02 seconds, so what's your problem? That means that somewhere between completing the processing and displaying the results to you, there is a bottleneck. Referring back to Figure 3, you might have a problem between the servers and the remote desktop or the remote desktop and your PC. It could be the volume you are sending.

With a big discrepancy between real time and response time, your first step should be to route your output directly to a file as described above. If that solves the problem, you have your answer. In most cases, you will not actually read the entire output results of 2000+ pages anyway.

If the problem is not solved by routing your output, the system may have a problem. In that case, you can start with the FULLSTIMER output and send that to your local SAS support person. The problem could be caused by many other external problems that can only be determined by your system support personnel.

Of course, if the true SAS performance is poor, you need to look at:

- The amount of work the program is completing
- The design of your program
- The load on your system.

Diagnosing other performance issues are beyond the scope of this discussion, but SAS Institute offers several documents to help with this analysis. See the Focus Area for Scalability and Performance on the SAS Web site at http://support.sas.com/rnd/scalability/index.html.

Variations over time

On the SAS Grid, you are now sharing the system with many other users, so it is possible that you are working during peak load period. If this seems to be impacting your work, you may want to talk with the system support team to see if additional resources can be added to the SAS Grid or if you can work during an off-peak time.

You can schedule larger long-running programs as batch jobs so they run when the system is not so loaded. In general, I like to convert long-running programs to batch jobs so I am not wasting my time waiting for the program to complete. "Long-running" is a question of how soon you need the results, how frequently you will run the program, and how comfortable or fast you are at creating and running batch jobs in Linux.

BACKGROUND AND PARALLEL PROCESSING

As an alternative to creating a batch job using SASGSUB, you can also use RSUBMIT to send a program to execute in a separate session; thereby freeing you up to continue working in SAS Enterprise Guide or SAS Studio. In fact, when people think about SAS Grid, remote submit comes to mind first for parallel processing. I provided examples above for parallel processing in a batch job, but you can do the same within a SAS session.

RSUBMIT for background processing

With the following code snippet, your program that includes all the necessary statements, such as LIBNAME and OPTIONS, is submitted to the SAS Grid in the background. The SAS Grid Manager will determine what server node runs the program and return the results to your session when it finishes. While the program executes, you are free to continue running other code in your current SAS Enterprise Guide or SAS Studio session.

```
%let grid_rc=%sysfunc(grdsvc_enable(_all_,resource=SASApp));

signon batch cmacvar=signonstatus;

rsubmit batch wait=no;

      <<< Enter your full program here >>>

endrsubmit;
```

Explanation:

- The GRDSVC_ENABLE statement using _ALL_ instructs the RSUBMIT facility to send all RSUBMIT requests to the SAS Grid application server called SASApp.
- The SIGNON statement creates a session called BATCH.
- The RSUBMIT sends any code after it to the SAS Grid session called BATCH.
- The WAIT=NO instructs RSUBMIT to return control back to your current session as soon as the RSUBMIT completes and not to wait for the program to complete.
- Note that my snippet does not include a SIGNOFF. In this case, I am assuming that you might want to resubmit the code again or submit new code using the same session.

RSUBMIT for parallel processing

In the following program, I use RSUBMIT to run two sections of code in parallel (simultaneously and asynchronously) and a third section of code that waits for both to finish before it combines the results of both. In this way, you can reduce your overall run time rather than waiting for each section to complete sequentially.

First, I define and create a temporary or permanent folder that can be shared between the two parallel sessions. I have macros called MAKENAME and MAKEDIR that I will not discuss in this presentation, but the first creates a unique name that I use as a subfolder and the second creates that subfolder under a folder called /SASDATA/shared/workshr. Once created, I issue a LIBNAME statement in the current session.

```
%let workroot=/SASDATA/shared/workshr;

%makename(outname=workdir);

%makedir(&workroot,&workdir);

%let datawrk=&workroot/&workdir;

libname tempshr "&datawrk";
```

As before, I am instructing SAS to send all RSUBMITS to the SAS Grid application server called SASAPP. I have removed all of my comments and other code automation for simplicity.

```
%let grid_rc=%sysfunc(grdsvc_enable(_all_,resource=SASApp));
```

Next I sign onto the first SAS Grid session called TSK1 with the SIGNON statement. Note that the INHERITLIB option makes the TEMPSHR libname available in the remote session also. I pass the value of J and WORKDIR to the remote session using %SYSLPUT so the program also has access to those macro variable values.

```
%let j=1;
signon tsk&j cmacvar=signonstatus inheritlib=(tempshr);;
%syslput j=&j;
%syslput workdir=&workdir;
```

Next the program runs the RSUBMIT to send code to the TSK1 session using WAIT=NO, so it runs asynchronously without waiting to complete, and CONNECTPERSIST=NO, so that the session is closed automatically when the RSUBMIT program completes.

```
rsubmit tsk&j wait=no connectpersist=no;
```

> Your program goes here. For this example, assume that it creates a data set called TEMPSHR.TEST1.

```
endrsubmit;
```

Repeat the same steps for the second section of code.

```
%let j=2;
signon tsk&j cmacvar=signonstatus inheritlib=(tempshr);;
%syslput j=&j;
%syslput workdir=&workdir;
rsubmit tsk&j wait=no connectpersist=no;
```

> Your program goes here. For this example, assume that it creates a data set called TEMPSHR.TEST2.

```
endrsubmit;
```

At this point, both sections of code are running in separate sessions on the SAS Grid and both are running at the same time. In this case, however, you want to wait for both to finish before you execute any more of your program. To do that, you issue the WAITFOR statement as shown below. You can list both sessions TSK1 and TSK2 or use _ALL_.

```
waitfor _all_;
```

Once the two parallel sessions have completed and returned to the current parent session, the following code could execute.

```
data combine;
   merge tempshr.test1
         tempshr.test2;
   by ……
run;
```

If I do not need the temporary folder, I will automatically release and delete it, but in many cases it is handy to keep it around so you can review the temporary and final data sets.

```
libname tempshr clear;

%deletedir(&datawrk);
```

At this point, you can review your results and resubmit the code as needed. Each time, the code could potentially run on different SAS Grid nodes depending on where the SAS Grid Manager sends them. The SAS Grid Manager routes your work to the best server node based on several factors including how busy the server is at the time of submission.

CONCLUSION

Hopefully, I have provided you with some useful tips and examples and some assurance that working on the SAS Grid is quite simple and not much different than how you have worked in the past. Although the SAS Grid provides many advanced features and options for parallel processing, you do not have to always use those features. In other words, RSUBMIT to the SAS Grid is an option for background or parallel processing, but it is not the most common or necessary way to work.

Because the servers are usually installed with multiple CPU and lots of memory, a properly sized SAS Grid platform will provide users better performance than experienced with a single-CPU PC. With the proper attention to the underlying architecture and the movement of output and data, you will be able to experience that benefit yourself.

With a little homework, patience, and experimentation, you will find a transition to SAS Grid platform running on Linux to be rather painless and ultimately rewarding.

REFERENCES

Brinsfield, Eric. 2016. "Reducing Build Time of Integrated Clinical Databases with SAS® Grid and SAS/OR®" Proceedings of the PhUSE Annual Conference 2016, Barcelona, Spain. Available at http://www.phusewiki.org/docs/Conference%202016%20CS%20Paper/CS04.pdf

Hall, Angela. 2014. "Tips and Tricks to Using SAS® Enterprise Guide® in a BI World Angela Hall, SAS Institute Inc., Cary, NC. http://support.sas.com/resources/papers/proceedings14/SAS331-2014.pdf

IBM Corporation. "IBM Platform LSF Command Reference." © Copyright IBM Corporation 1994, 2017. Available at https://www.ibm.com/support/knowledgecenter/SSETD4_9.1.3/lsf_kc_cmd_ref.html

Jumper, Susan and Kotian, Harsha. 2014 "SAS Enteprise Guide® Options: Let the Tools Work for You". Available at https://www.sas.com/content/dam/SAS/en_ca/User%20Group%20Presentations/TASS/JumperKotian-EGTools.pdf

ACKNOWLEDGMENTS

I want to thank Joe Olinger of d-Wise for his installation and configuration expertise and support through multiple SAS Grid Manager projects.

RECOMMENDED READING

- *Base SAS® 9.4 Procedure Guide, Seventh Edition*
- *Grid Computing in SAS® 9.4, Fifth Edition*
- *Moving and Accessing SAS® Files, Third Edition.*
- *SAS® 9.4 System Options: Reference, Fifth Edition*

CONTACT INFORMATION

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

Eric Brinsfield
Meridian Analytics
919-302-3747
Eric.Brinsfield@MeridianAnalytics.com
Virginia Beach, VA