Paper 046-2008

# SAS® Grid 101: How It Can Modernize Your Existing SAS® Environment

Cheryl Doninger, SAS Institute, Cary, NC
Glenn Horton, SAS Institute, Cary, NC

## ABSTRACT

Grid computing promises many benefits, including improved performance of applications, higher resource utilization, lower cost of ownership, and flexibility for your IT infrastructure. Many of you are evaluating grid computing to assess its relevance and value for your organization. This paper helps you by addressing two topics, "What business problems does SAS® Grid Computing solve?" and "Show me the code." We describe many of the business issues that can be addressed by SAS Grid Computing, as well as provide code examples of how to implement SAS applications on the grid. Learn how you can use SAS Grid Computing to modernize your existing SAS® environment and add new value to your existing applications with little or no change.

## INTRODUCTION

SAS customers span multiple geographies, many different industries, and a multitude of different application spaces. In spite of these differences, there are many common pains and business issues that cause customers to evaluate deploying a SAS grid architecture. From an architecture perspective, common themes include IT concerns such as the following:

- Our current server is obsolete and no longer able to meet our data processing needs. We have additional modeling and simulation that is needed, but cannot be done because of a lack of computing resources. This opportunity cost represents millions of dollars annually. Additionally, not only are the maintenance costs of our current SMP server prohibitively expensive, but estimates to replace this server with something adequate that meets our needs is over $2 million, which substantially exceeds our budget.
- We experience frequent server crashes because our users submit too many jobs to the same central server or jobs requiring a lot of resources. We don't have a way to manage these requests and efficiently map them to the available resources.
- We need an environment that provides the flexibility to incrementally grow as our data and processing needs grow in the future.
- We would like to eliminate the single point of failure of our current environment to guarantee continuous availability. We want to allow the flexibility to do regular maintenance and upgrades to the hardware and software in our compute environment without interruption and system down time for our users.

From an application perspective, the business units have multiple types of workload and application needs to be met. For example:

- Many short ad hoc SAS jobs initiated by the user at the command line, or by a shell script at the command line. The user expects immediate execution and display of results.
- Ad hoc batch job submission by users at the command line. These batch jobs can take a couple of minutes to several hours.
- Standard production SAS jobs. This is a single SAS job, called directly or by a shell script, which runs at a scheduled time or immediately once the required input data arrives.
- Complex workflows that require a way to represent job dependencies and to trigger execution based on complex calendar events, file events, or both.
- Parallelized production SAS jobs that incorporate the parallel capabilities of SAS/CONNECT to accelerate application execution.
- Interactive SAS session execution for users developing SAS applications.

All of the above business needs can be addressed by a SAS grid environment, which provides an integrated solution that meets the needs of your organization today, and is easily extended to meet the needs of your organization as you grow in the future. A SAS grid deployment enabled by SAS Grid Manager can modernize your existing SAS environment.

## SAS GRID MANAGER

SAS Grid Manager was introduced in SAS 9.1.3. It builds on the parallel capabilities of SAS/CONNECT and adds many other capabilities required by enterprise grid deployments. SAS Grid Manager provides load balancing, policy

enforcement, efficient resource allocation, and prioritization for SAS products and solutions running in a shared grid environment. In addition, it decouples the SAS applications and the infrastructure used to execute the applications. This allows hardware resources to transparently grow or contract as needed, and provides tolerance of hardware failures within the grid infrastructure. SAS Grid Manager integrates the resource management and scheduling capabilities of the Platform Suite for SAS with the SAS 4GL syntax, and subsequently with several SAS products and solutions. This integration enables you to create a managed, flexible, and shared environment to efficiently process

- multiple users,
- parallel workloads,
- and enterprise scheduling.

**MULTIPLE USERS**

Many customers have ad hoc SAS users that develop models, do queries, and perform other kinds of ad hoc development, discovery, and analysis. SAS Grid Manager manages this ad hoc environment from the perspective of controlling which jobs get priority, deciding which jobs get which share of the computing resources, and preventing a mass submission of work requests that results in frequent server crashes. SAS Grid Manager maps the SAS work requests to the available resources. If necessary, it runs a subset of the work and queues the remaining work for execution as soon as resources become available. High-priority jobs can even preempt low-priority work so that the most critical business processes execute first. SAS Grid Manager provides the structure to create an organized and managed SAS analytic environment. This ensures that the appropriate resources are allocated to the appropriate users or appropriate workload to meet the objectives of the organization.

Users in the SAS community are using many different interfaces for running SAS applications. Some users might be running SAS in interactive DMS mode. In this case, jobs can be interactively submitted to a grid environment. This is achieved by defining a new key sequence to submit the appropriate statements to send the job to the grid, rather than executing it on the local workstation. For users who prefer batch job submissions, a wrapper script file can submit batch jobs to the grid with no change to the application and no change in the way your users interact with SAS. SAS® Enterprise Guide users can submit their SAS programs and SAS Enterprise Guide tasks to a SAS grid for execution. SAS Grid Manager provides the infrastructure to balance all SAS applications and workload across a shared grid infrastructure.

**PARALLEL WORKLOADS**

A subset of SAS applications consists of sub-tasks that are independent units of work and can be distributed across a grid and executed in parallel. A potential benefit of distributed parallel execution is substantial acceleration of the entire application. A common workflow in applications created by SAS® Data Integration Studio is to repeatedly execute the same analysis against different subsets of data. For this workflow, the Loop and Loop End transformation nodes can be used in SAS Data Integration Studio to automatically generate a SAS application that spawns each iteration of the loop to a SAS grid via SAS Grid Manager. SAS® Risk Dimensions® has a similar iterative workflow of executing the same analysis against different subsets of data. The data is subset based on market states or by instruments. Each iteration of the analysis can be submitted to the grid using SAS Grid Manager to provide load balancing and efficient resource allocation. In contrast, the workflow for SAS® Enterprise Miner™ during the model training phase is executing a series of different models against a common set of data. Because the models are independent of each other, the SAS Enterprise Miner engine that generates the SAS program to execute the user-created workflow automatically inserts the syntax to spawn each model to the grid for parallel and accelerated execution. A programmer might modify an existing application or develop a new application that consists of replicate runs of the same fundamental task or multiple, distinct, independent units of work. The programmer can use the grid syntax in SAS 4GL to create a grid-enabled application. In all of these scenarios, SAS Grid Manager distributes and balances the parallel work requests to a shared pool of SAS grid resources.

**ENTERPRISE SCHEDULING**

Scheduling production jobs is an important and necessary function in every enterprise. SAS provides the Schedule Manager plug-in with SAS® Management Console so you can create SAS workflows and schedule them based on time and file events. SAS Grid Manager distributes multiple workflows or the jobs within a single workflow to a SAS grid environment, while balancing loads and sharing resources. SAS jobs can be created by a variety of SAS applications and solutions or written by SAS programmers. Various levels of scheduling capabilities have been incorporated directly into many SAS products and solutions, including SAS Data Integration Studio, SAS® Marketing Automation, SAS® Marketing Optimization, and SAS® Web Report Studio. The jobs created by SAS products and user-written SAS programs can be used to create simple or complex workflows and Schedule Manager can be used to schedule them to a SAS grid environment.

## MOVING YOUR SAS APPLICATIONS TO THE GRID

A major component of the success of migrating to a grid infrastructure is ensuring that your architecture includes a shared file system that is capable of sustaining the I/O bandwidth required by your applications. SAS grid environments have been tested with a variety of shared file systems showing near-linear scalability. This paper is focused on the steps necessary to run SAS applications on a grid. For complete details on best practices running a shared file system with a distributed grid environment, see "Best Practices for Data Sharing in a Grid and Distributed SAS® Environment" available at http://support.sas.com/rnd/scalability/grid/gridpapers.html.

If you are running SAS solutions or products that have grid capabilities built into their GUI development environments, there is minimal change in the way you use the application to run on the grid. For example, in SAS Enterprise Miner, you need to set just a single option. Select **Options→Preferences**. Under **Run Options**, change the value for **Grid Processing** from **Never use grid processing** to **Use grid processing when available**. You are ready to run SAS Enterprise Miner on the grid. In SAS Data Integration Studio, use the Loop and Loop End transformations as you create your workflow. Select the **Execute iterations in parallel** check box in the Loop Properties window and your jobs will execute on the grid. No programming knowledge is required because the code generation engine within these products generates the syntax to spawn SAS processes on the grid. The grid infrastructure is completely decoupled from the user of the application; therefore, no knowledge of the grid infrastructure is necessary.

If your organization is like most organizations, you have many existing SAS programs that have been developed over the years to help you run your business. SAS has implemented grid capabilities that provide many ways for you to move these SAS programs to the grid with minimal or no change to the SAS program itself. The following sections outline a phased approach to moving your SAS applications to the grid.

- Phase 1 – Just get your jobs running on the grid for multi-user workload balancing
- Phase 2 – Schedule production jobs with enterprise scheduling
- Phase 3 – Examine individual jobs that might be candidates for parallel workload balancing

### PHASE 1 - JUST GET YOUR JOBS RUNNING ON THE GRID

Traditional SAS users run their SAS programs using SAS DMS, which provides a user interface for interactive job submission, or using a command-line interface, which allows for the submission of SAS batch jobs, or using both.

If you use DMS for interactive job submission, it is easy to create a key sequence that submits the contents of the SAS Program Editor window to the SAS grid rather than executing it locally on your workstation. This is done by creating a key definition that inserts the appropriate grid and SAS/CONNECT statements to the beginning and end of your code. Your SAS log and output are returned to your local workstation as if the program had been executed locally. Most important, there is no change to the logic of your SAS program. Two steps detail how this is done:

1. Save the following statements to an external file (for example, c:\gpre.sas).

```
options noconnectpersist;
options noconnectwait;
options metaserver='dnnnn';
options metaport=8561;

%let rc=%sysfunc(grdsvc_enable(grid, resource=SASMain));
signon grid;
```

2. In SAS DMS, open the Keys window and enter the following for any available key (for example, F12):

```
gsubmit "%include 'c:\gpre.sas';"; rsubmit;
```

You can type or include any SAS program in your Program Editor window, and then press F12. The program is submitted to computing resources on your SAS grid instead of executed locally. Note the following:

- Your local machine will be busy until SIGNON completes and RSUBMIT sends the code to the grid. At that point, your local machine is available for further processing.

- Multiple submissions in a row using the same key definition (F12, in this case) will cause multiple jobs to be queued. Each job will be submitted to the grid when the subsequent job has completed. If you need to be able to submit multiple jobs to the grid at the same time from the same SAS session, you need to define multiple keys, each with a unique remote session ID on the SIGNON statement. For example, `signon grid1`, `signon grid2`, and so on.

If you use the command-line interface for the submission of SAS batch jobs, it is seamless to submit the same SAS batch jobs to a shared grid environment. SAS has script files, one for Windows operating systems and one for UNIX operating systems, that wrap an existing SAS program with the appropriate grid and SAS/CONNECT statements to submit the program to SAS Grid Manager. These script files can be downloaded under "SAS Grid Computing Downloads" from the Web site http://support.sas.com/rnd/scalability/grid/download.html. The syntax for invoking the UNIX script file is:

```
./runsas.sh <sas file name> <extra SAS options>
```

The runsas.sh script does the following:

1. Creates a temporary file.
2. Writes the grid and SAS/CONNECT statements to the temporary file for grid submission.
3. Copies the contents of *<sas file name>* to the temporary file.
4. Submits the contents of the temporary file with *<extra SAS options>* to the grid for execution.
5. Performs error checking and messaging in the event of errors.

If your organization uses SAS Enterprise Guide, there are a couple of ways to easily submit the generated programs to a grid and balance the load of jobs submitted by multiple users. This can be automatic by following a few simple steps documented in http://support.sas.com/rnd/scalability/grid/EG_on_grid.pdf. The steps consist of inserting grid and SAS/CONNECT statements into the following locations:

- For SAS Enterprise Guide programs, select **Tools→Options→SAS Programs** and select **Insert custom SAS code before submitted code**.
- For SAS Enterprise Guide Tasks, select **Tools→Options→Tasks** and select **Insert custom SAS code before submitted code**.

With SAS 9.2, the SAS Workspace Server is integrated with SAS Grid Manager to provide load balancing capabilities for multiple workspace servers. Because SAS Enterprise Guide submits the generated SAS programs to the SAS Workspace Server for execution, you could create a cluster of workspace servers and enable load balancing using the **grid** algorithm. This enables SAS Grid Manager to balance the load of multiple SAS Enterprise Guide jobs across all workspace servers in the cluster. For more information, see "Balancing the Load - SAS® Server Technologies for Scalability" also being presented at SAS Global Forum 2008.

#### PHASE 2 - SCHEDULE PRODUCTION JOBS

The scheduling framework in SAS Grid Manager is a robust capability that enables you to create SAS workflows and schedule them based on time and file events. The Schedule Manager in SAS Management Console provides a point-and-click environment that enables you to schedule your SAS jobs. Three main steps are required to schedule a SAS job to the grid:

1. Deploy the SAS program or job for scheduling.
2. Create a new workflow containing one or more SAS jobs.
3. Schedule the workflow.

Some SAS solutions, such as SAS Web Report Studio, provide their own templates that allow you to complete all three of the steps from within the solution. Other products, such as SAS Data Integration Studio, enable you to deploy a SAS job for scheduling, and then you use the Schedule Manager plug-in to perform steps 2 and 3. If you are scheduling user-written SAS programs, then you perform all three steps using the Schedule Manager plug-in. Figures 1 through 3 are displays of the three main steps.

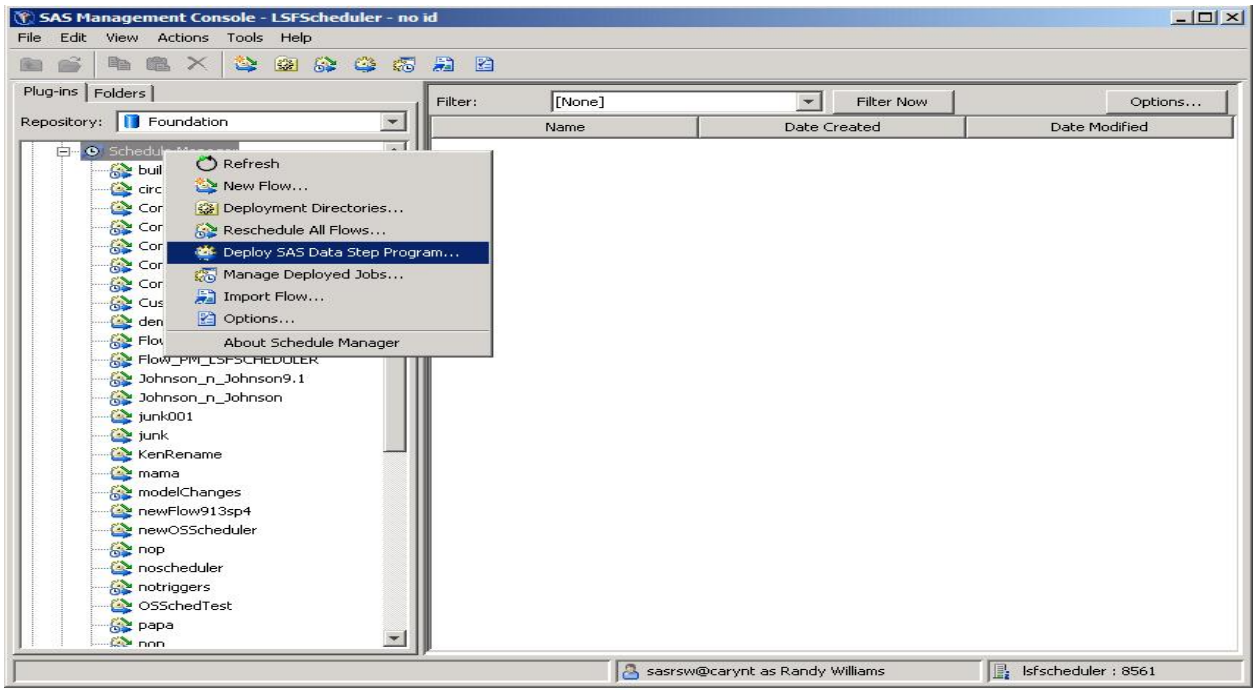


**Figure 1: Deploy a SAS Program for Scheduling**

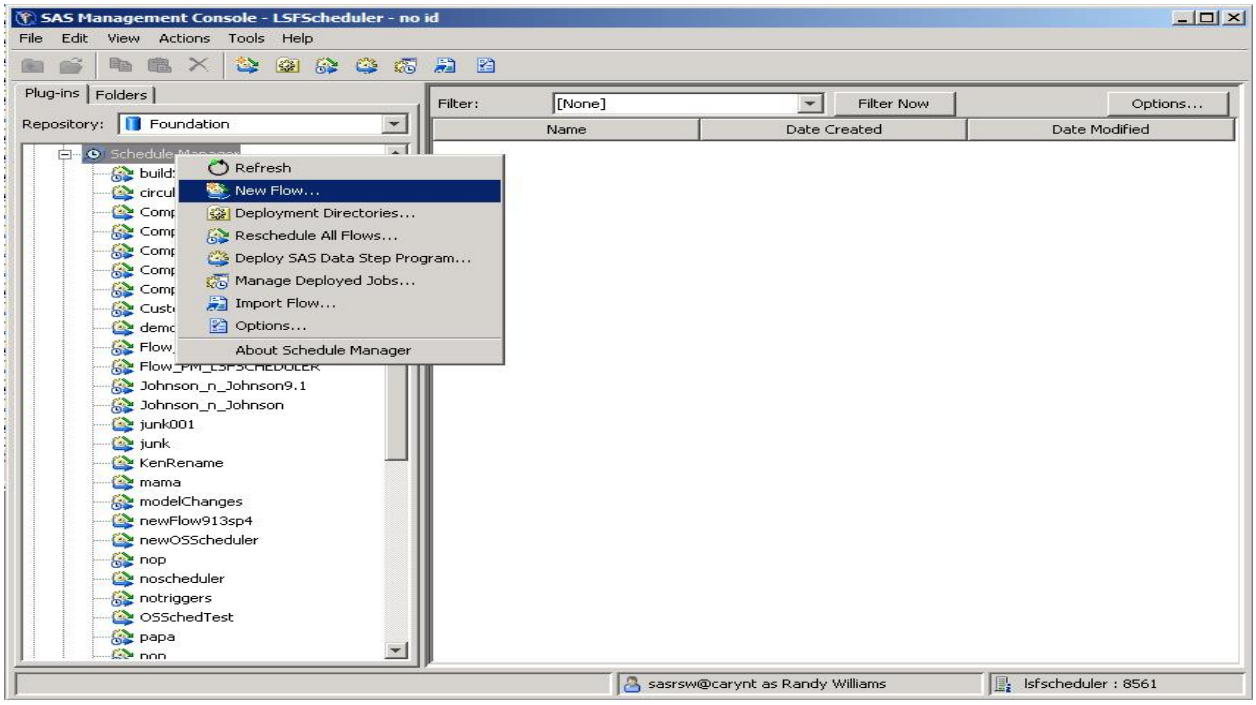


**Figure 2: Create a New Workflow**

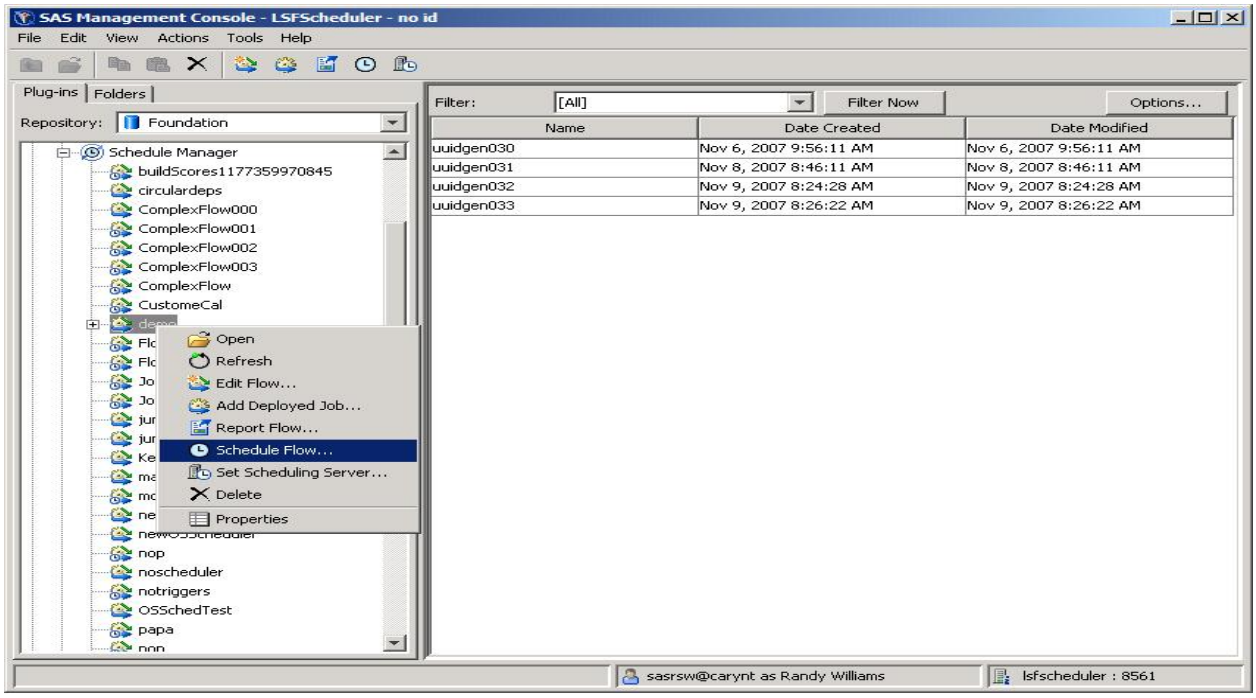


**Figure 3: Schedule the Workflow**

**PHASE 3 - EXAMINE INDIVIDUAL JOBS FOR PARALLEL EXECUTION**

A subset of SAS applications consists of sub-tasks that are independent units of work and can be distributed across a grid and executed in parallel. A potential benefit of distributed parallel execution is substantial acceleration of the entire application. A programmer might modify an existing application or develop a new application that consists of replicate runs of the same fundamental task or multiple, distinct, independent units of work. The following code shows the SAS/CONNECT syntax that identifies the parallel sub-tasks of an application.

```
options autosignon;
rsubmit task1 wait=no;
   /* code for parallel task #1 */
endrsubmit;
rsubmit task2 wait=no;
   /* code for parallel task #2 */
endrsubmit;
   .
   .
   .
rsubmit taskn wait=no;
   /* code for parallel task #n */
endrsubmit;
waitfor _all_ task1 task2 ... taskn;
signoff _all_;
```

The parallel capabilities in this code have been available with SAS/CONNECT since Version 8 of SAS. To execute the program using just SAS/CONNECT, you would have to assign specific machine names or IP addresses to the `task1` through `taskn` variables. Because SAS Grid Manager decouples the infrastructure from the application, the programmer does not have to assign processing to specific machines. This becomes part of the functionality provided by SAS Grid Manager. The only code that needs to be added is the GRDSVC_ENABLE function call. Complete syntax for all of the grid function calls is available under “Using SAS on a Grid” from the Web site http://support.sas.com/rnd/scalability/grid/gridfunc.html. The GRDSVC_ENABLE function provides the information for SAS to locate the grid logical server definition in SAS metadata and to read the parameters to start a SAS process on

the grid. The following GRDSVC_ENABLE function call is the only statement that needs to be added to the beginning of the previous SAS program for it to spawn multiple SAS sessions to the grid to execute `task1` through `task`*n*.

```
%let rc=%sysfunc(grdsvc_enable(_all_, resource=SASMain));
```

If your organization already has applications that use the parallel capabilities of SAS/CONNECT, you simply add this single line of code to the beginning of each application to begin running it with SAS Grid Manager. The GRDSVC_ENABLE function call requires a connection to the SAS® Metadata Server. Therefore, the metadata connection options of METAXXXX must have been specified or you are prompted to enter this information when the GRDSVC_ENABLE function executes. A best practice is to specify these options in the autoexec.sas file, rather than inserting them into multiple programs.

The following SAS program is a complete application that uses SAS Grid Manager for execution across a grid. The original application used PROC IML to calculate a statistic called the pairwise test statistic. This statistic is the sum of the number of shared genes between all possible pairs of genomes in two populations. This calculation requires the comparison of two matrices, each of which includes 4600 columns and 40 rows. Each column in one matrix is compared to all other columns in the other matrix. This results in 21,000,000 comparisons and an unacceptably long execution time if run sequentially. Running this application on a grid is the only way to get results in a reasonable and timely fashion.

```
/* Program to calculate pairwise test statistic on a grid */
%macro combine_data_sets;
data sumds;
set %do i = 1 %to &n_nodes;
      mydata.sumds&i
    %end;
;
run;
%mend combine_data_sets;

%macro data_macro;
data %do i = 1 %to %eval(&n_nodes - 1);
        mydata.input&i(keep = x%eval(&columns_per_node * (&i - 1) + 1) - x%eval(&i *
        &columns_per_node))
     %end;
        mydata.input&n_nodes(keep = x%eval(&columns_per_node * (&n_nodes - 1) + 1) -
        x%sysfunc( attrn(&dsid,nvars)))
;
set mydata.p1;
run;
%mend data_macro;

libname mydata "/shared";

/* Generate sample data sets p1 and p2 */
%let ncol = 500;
proc iml;
pop1 = j(40,&ncol);
pop2 = j(40,&ncol);
p = j(1, 64, 1/64);
call randseed(11111111);
call randgen(pop1, 'table', p);
call randgen(pop2, 'table', p);
cname = "x1":"x&ncol";
create mydata.p1 from pop1[colname=cname];
append from pop1;
create mydata.p2 from pop2;
append from pop2;
quit;
```

```
/* tell SAS how to connect to the SAS Metadata Server */
options metaserver=aaa.bbb.ccc.com;
options metaport=8561;

/* Tell SAS/CONNECT to do grid signons */
%let rc=%sysfunc(grdsvc_enable(_all_, "resource=SASMain"));

/* indicate how many nodes should be used */
%let n_nodes = 4;

/* break up data set p1 into separate input data sets, one for each node */
%let dsid=%sysfunc( open(mydata.p1,i));
%let columns_per_node = %eval( %sysfunc( attrn(&dsid,nvars)) / &n_nodes);
%put &columns_per_node;
%data_macro
%let rc = %sysfunc( close(&dsid));

%macro accumulate_loop;
/* loop over the data sets */
%do i = 1 %to &n_nodes;
%put &i;
%let start_column=%eval((&i - 1) * &columns_per_node + 1);
signon t&i;
/* let remote know which data sets to use */
%syslput i=&i;
/* let remote know start column */
%syslput start_column=&start_column;
rsubmit wait=no persist=no;
libname mydata "/shared";
proc iml;
use mydata.input&i;
read all into pop1;
use mydata.p2;
read all into pop2;
sum = j( ncol(pop1), ncol(pop2));
zero_vec = j(1,ncol(pop2),0);
do i=1 to ncol(pop1);
  column_1 = pop1[,i];
  a = zero_vec;
  do k = 1 to nrow(pop1);
  a = a + (column_1[k] = pop2)[+,];
  end;
  sum[i,] = a;
end;

/* output data */
t0 = time();
block = j(ncol(pop2),3);
block[,2] = t(1:ncol(pop2));
create mydata.sumds&i from block;
do i = 1 to ncol(pop1);
  block[,1] = j(ncol(pop2),1,i + &start_column - 1);
  block[,3] = t(sum[i,]);
  append from block;
end;
quit;
endrsubmit;
%end;

waitfor _all_ t1 t2 t3 t4;
%mend accumulate_loop;
```

```
/* run the PROC IML program on each data set */
%accumulate_loop

/* combine all data sets into one final data set */
%combine_data_sets
```

## MANAGING YOUR SAS APPLICATIONS ON THE GRID

SAS Grid Manager provides an integration between the SAS application environment and grid middleware from Platform Computing, which is known as the Platform Suite for SAS. The Platform Suite for SAS has three components:

- Process Manager (previously Platform Job Scheduler) – provides the interface used by the SAS scheduling capability. This interface controls the submission of scheduled jobs to LSF and manages any dependencies between the jobs.
- Grid Management Services – communicates with the Grid Manager plug-in to provide run-time monitoring and management within SAS Management Console for the jobs, hosts, and queues in the grid environment.
- LSF – dispatches all jobs submitted to it, either by the Process Manager or directly by SAS, and then returns the status of each job. LSF manages any resource requirements and performs load balancing across machines in a grid environment.

As the name implies, the load sharing facility (LSF) is the component responsible for performing the load balancing and resource allocation of all work requests submitted to the grid. LSF does not execute jobs immediately. It puts the jobs in a specified queue or in the **normal** queue if no queue is specified. Multiple queues can be defined with different priorities, resource allocations, time policies, and preemptive capabilities. LSF matches the requirements of the job with the resources of the host as defined by the administrator. It decides when jobs should be picked up from the queue and executed on a host. Several factors affect this decision, such as the load on the host and the requirements of the job.

In SAS 9.2, many enhancements have been made to the GRDSVC_ENABLE function to allow additional job submission capabilities[1]. One enhancement is the ability to specify the name of the job that is run in the grid. The advantage of this feature is that the user can assign a meaningful name to the job so that when the job shows up in the Grid Manager plug-in in SAS Management Console, it is much easier to recognize and monitor. Another enhancement is the ability to direct your job to a specific queue. Your administrator might choose to define multiple queues to meet the needs of different users. The **normal** queue is where all SAS work requests are directed by default. It might be useful to create a high-priority queue for jobs that need to be executed immediately, a short queue for jobs of short duration, and a nighttime queue for batch jobs to be run overnight. The following code shows the syntax to specify a job with the name `FinanceSummary` and to direct that job to the `priority` queue.

```
%let jnvar=FinanceSummary;
%let jovar=queue=priority;

%put %sysfunc(grdsvc_enable(_ALL_, server=SASApp; jobname=jnvar; jobopts=jovar));
signon t;
```

Figure 4 is a display of the Grid Manager plug-in in SAS Management Console. This job view shows the status of each job currently running in the grid. As a result of the previous code, the job name `FinanceSummary` is in the **Job Name** column.

[1] In SAS 9.1.3, the ability to rename jobs and direct job execution to a specific queue is implemented with an ESUB script file that modifies the job submission parameters.

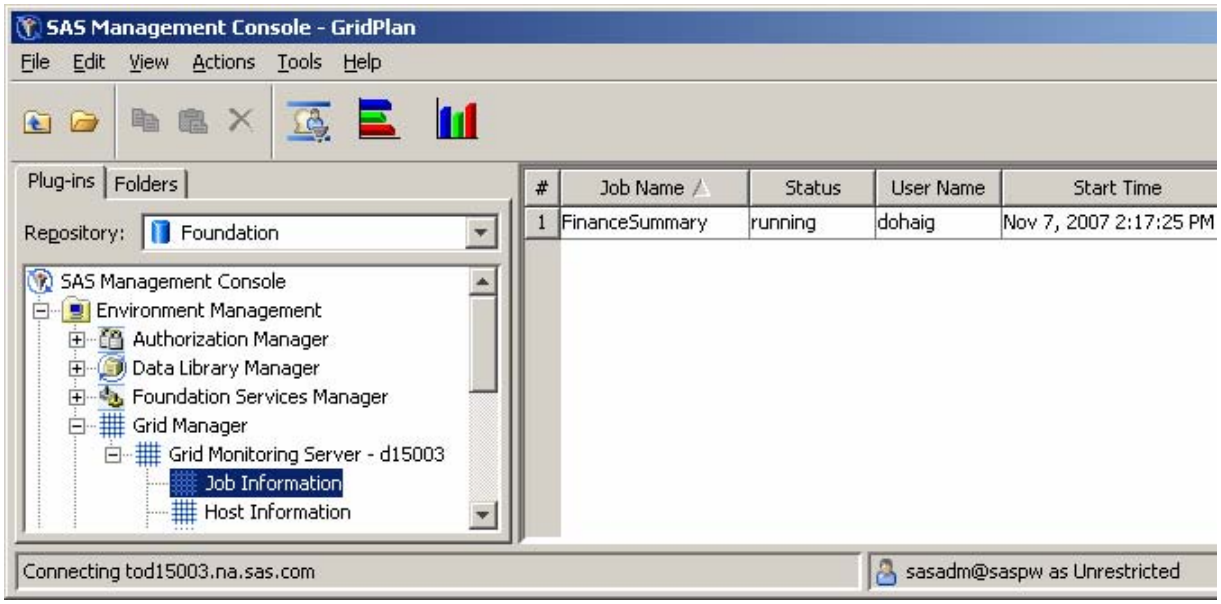


**Figure 4: User-Named Job - FinanceSummary**

Figure 5 displays the queue view of the Grid Manager plug-in in SAS Management Console. This view shows all queues that have been defined by the grid administrator and current activity and status for each queue. As a result of the previous code, the job named `FinanceSummary` was directed to the high-priority queue.

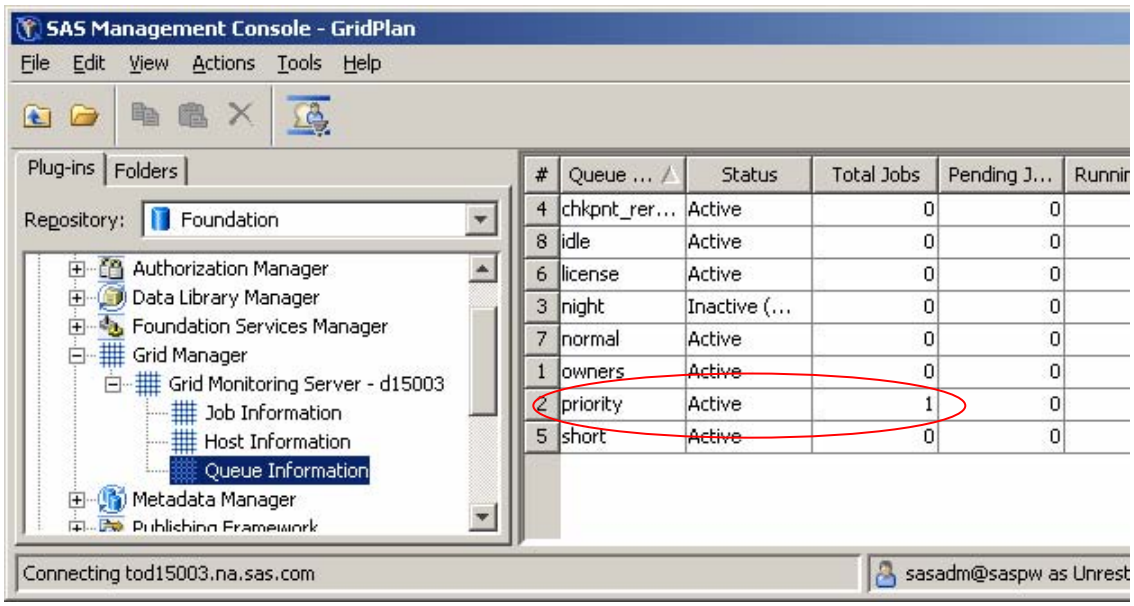


**Figure 5: User-Named Queue - Priority**

All options and capabilities are available for use at the discretion of the programmer or user. The grid administrator can set all of these options and capabilities in the grid logical server definition in SAS metadata. Metadata access permissions can control who has access to modify or read the contents of the definition. Any option that is specified in the SAS metadata overrides any option value that is specified by the programmer in the GRDSVC_ENABLE function call. Figure 6 is a grid logical server definition with the queue specified. As a result, any SAS work requests that access this grid logical server definition are directed to the `priority` queue. Any queue specification on the GRDSVC_ENABLE function call will be over-ridden by this specification in the grid logical server definition.

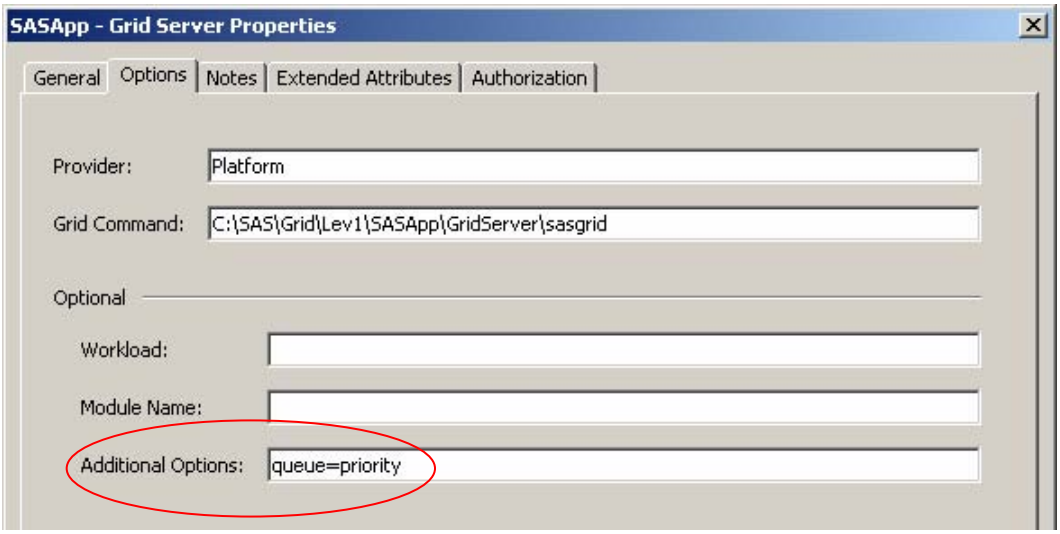


**Figure 6: Queue Name of priority Specified in Grid Logical Server Definition**

The grid or system administrator defines the queues and their associated attributes in the lsb.queues file (as defined in ***$LSB_CONFDIR/<cluster_name>/configdir***). The administrator can completely control which users are able to access which queues. The following example is a portion of an lsb.queues file. It shows the definition of the default **normal** queue and three additional queues. The **priority** queue has the highest priority specified, which results in jobs in this queue being dispatched for execution before any jobs in the remaining queues. The **night** queue has time policies to limit the time that jobs are dispatched from this queue. The time has to be between the hours of 6:00 p.m. and 7:30 a.m. A run-time policy is included to ensure that jobs finish running by 8:00 a.m. each day. By default, any jobs that have not completed by 8:00 a.m. are suspended and resumed the next day at 6:00 p.m. The **short** queue has been defined to process jobs of short duration. The **PREEMPTIVE** attribute on the **short** queue allows jobs in this queue to preempt any currently executing jobs that originated from a queue with the **PREEMPTABLE** attribute, which is the **normal** queue in this case.

```
Begin Queue
QUEUE_NAME = normal
PRIORITY = 30
PREEMPTION = PREEMPTABLE[short]
DESCRIPTION = default queue
End Queue

Begin Queue
QUEUE_NAME = priority
PRIORITY = 40
DESCRIPTION = high priority users
End Queue

Begin Queue
QUEUE_NAME = night
PRIORITY = 30
DISPATCH_WINDOW = (18:00-07:30)
RUN_WINDOW = (18:00-08:00)
HOSTS = all ~host1
DESCRIPTION = night time batch jobs
End Queue

Begin Queue
QUEUE_NAME = short
PRIORITY = 35
PREEMPTION = PREEMPTIVE[normal]
```

```
DESCRIPTION = for jobs of short duration
End Queue
```

## CONCLUSION

There are many benefits to running SAS applications and solutions in a grid environment. These include:

- Creating an environment that can be shared by multiple users and multiple applications
- Allocating resources dynamically to meet peak demands
- Implementing policies and priorities to govern the use of computing resources
- Running larger or more complex analysis
- Easing maintenance of applications by separating the applications from the infrastructure
- Easing maintenance of infrastructure by allowing machines to be taken offline without affecting applications
- Improving price and performance through the use of commodity hardware
- Scaling out cost effectively as data volumes, computing needs, and number of users grow

The SAS grid capabilities were designed at the syntactic level to enable SAS programmers and SAS solutions to leverage the many benefits of a grid infrastructure. The SAS grid capabilities were also designed so that an existing SAS environment can be transitioned from an obsolete silo server environment to a shared and flexible grid infrastructure with minimal risk and minimal change to both the application and the processes by which users interface with SAS.

The business pains outlined in the introduction, such as obsolete hardware, existing systems at capacity, the need to run more complex analysis, frequent server crashes because of overloaded systems, and the need to establish and enforce policies to meet the needs of multiple SAS users can all be addressed by a SAS grid environment. The phased transition to a SAS grid environment helps minimize risk and ensures a successful move to a shared, managed, and more robust computing environment that meets the needs of your organization both today and in the future.

## ACKNOWLEDGMENTS

The authors acknowledge Joe Hutchinson at SAS for his contributions to this paper.

## FOR MORE INFORMATION

For more information about SAS and grid computing, visit the following Web sites:

Introduction to Grid Computing – support.sas.com/rnd/scalability/grid

SAS Grid Computing – www.sas.com/grid

Platform Suite for SAS – http://support.sas.com/rnd/scalability/platform/index.html

Syntax for SAS/CONNECT Grid Functions – support.sas.com/rnd/scalability/grid/gridfunc.html

## REFERENCES

Doninger, C., and A. Wong, 2006. “SAS Goes Grid – Managing the Workload Across Your Enterprise.” Proceedings of the Thirty-first Annual SAS Users Group Internation Conference. Cary, NC: SAS Institute Inc. Available at http://www2.sas.com/proceedings/sugi31/211-31.pdf.

Tran, A., and R. Williams, 2004. “Implementing Site Policies for SAS Scheduling with Platform JobScheduler.” Available at http://support.sas.com/resources/papers/JobScheduler.pdf.

## CONTACT INFORMATION

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

Cheryl Doninger
SAS Institute, Inc.
Phone: 919-531-7941
E-mail: cheryl.doninger@sas.com

Glenn Horton
SAS Institute, Inc.
Phone: 919-531-6640
Email: glenn.horton@sas.com