Paper 211-31

SAS® Goes Grid – Managing the Workload across Your Enterprise

Cheryl Doninger, SAS Institute, Cary, NC
Alan Wong, Platform Computing, Markham, Ontario

ABSTRACT

The intensive computing requirements being placed on IT are demanding a dynamic environment to effectively manage the workload across your enterprise. As data volumes continue to grow while the window of processing time continues to shrink, applications analyzing enormous amounts of data can be optimized with shared and dynamically allocated resources. Learn how grid computing and scheduling have been incorporated and automated to deliver value in a highly efficient manner for SAS analytics, data integration (ETL), data mining, and business intelligence. Also learn about advanced configuration options that you can use to fine-tune your SAS grid environment and allow multiple applications to efficiently and dynamically use a virtual IT infrastructure.

SAS GOES GRID

A grid infrastructure provides many benefits to the enterprise including:

- application acceleration – often using existing IT infrastructure
- increased flexibility of enterprise computing infrastructure – ability to incrementally add resources as your business grows and your needs change
- creation of a virtual environment – to allow you to share and provision resources in order to most effectively balance workload and meet service levels across the enterprise

Exploding data volumes, increasing demands on business, and ever-shrinking windows of time are causing "analysis paralysis" for many organizations. This has resulted in computation-intensive and data-intensive application needs that cannot be met with existing IT infrastructure, given the way in which that infrastructure is currently used and managed. Because of the enormous power of SAS analytics, SAS applications are used for data- and computation-intensive analyses by every type of SAS customer in every industry segment. Therefore, many SAS applications have a natural affinity for a grid infrastructure. There are several ways to allow a SAS workflow to run in a grid environment:

1. by distributing SAS job components and/or multiple SAS jobs to the grid
2. by scheduling SAS workflows made up of one or more jobs to the grid
3. by combining 1 and 2

DISTRIBUTING SAS JOBS TO THE GRID

Many SAS applications are made up of replicate runs of the same fundamental task or many independent tasks that can be distributed to a grid environment for parallel execution resulting in acceleration of information and results. As a SAS user, you can choose from automated grid capabilities in an easy-to-use, point-and-click interface or by using the syntax of the powerful 4GL SAS programming language for distribution of SAS applications to a grid. For example, a common workflow in ETL processing is the need to run the same analysis (multiple iterations) over different subsets of data such as each state in the U.S. or each sales territory in your organization. New loop transforms have been added to SAS Data Integration Studio 3.3 to allow multiple iterations to automatically distribute to a grid environment for parallel and therefore faster execution. A common workflow in data mining is the need to run multiple models (independent tasks) over the same input data source. SAS Enterprise Miner 5.2 can now automatically distribute parallel nodes in a SAS Enterprise Miner workflow to a grid environment with similar benefit. Both SAS Data Integration Studio and SAS Enterprise Miner have built-in intelligence to automatically create the necessary syntax in the generated SAS program to run in a grid environment when appropriate. The result is that the grid infrastructure remains transparent to you as a user of these applications allowing you to focus on your ETL and data mining tasks and also benefit from better performance of your applications. The applications generated by SAS Data Integration Studio and SAS Enterprise Miner can also be saved as a SAS stored process and subsequently used by the SAS Business Intelligence (BI) components.

In addition to the automated grid capabilities of the SAS data integration, data mining, and business intelligence components, you also have the flexibility of developing applications using the SAS programming language to run in a grid environment. The same syntax used by the SAS solutions to generate programs behind the scenes can be used by SAS programmers to identify subtasks of a long-running application to distribute across the grid. This syntax can also be used as a wrapper around each user's job as a whole for the purposes of load-balancing SAS jobs from a number of users needing to use a virtualized pool of resources. Refer to the "For More Information" section at the end of this paper for a link to an example of how this would be done.

SAS/CONNECT provides the syntax to enable distribution of SAS job components. The parallel processing capabilities of SAS/CONNECT have been integrated with components from Platform Computing to provide the most efficient workload distribution to the grid resources, efficient management of the grid resources, and run-time monitoring of the SAS grid environment. Refer to the "For More Information" section at the end of this paper for a link to the SAS/CONNECT grid syntax.

SCHEDULING SAS WORKFLOWS TO THE GRID

The SAS scheduling interface can be used to schedule a SAS workflow. It is directly incorporated into a number of SAS products and solutions, including SAS Data Integration Studio, SAS Web Report Studio, SAS Marketing Automation, and SAS Marketing Optimization. In addition, any SAS program created by a SAS solution or SAS programmer can be scheduled using the Schedule Manager plug-in within SAS Management Console. Using the SAS scheduling integration with Platform Computing, you can schedule a flow based on a trigger event. A trigger event can be a specific date/time event or a file event. A trigger can also be a recurring event. A flow is typically made up of multiple jobs where two or more of the jobs can be executed simultaneously. In this case, when the job is triggered to execute, the parallel jobs are distributed to resources in the grid environment. Even for flows that contain only a single job or jobs that must execute in a serial fashion, multiple users scheduling such flows benefit by having these flows execute on the most appropriate resource and the total multi-user workload effectively balanced within the grid.

The SAS grid capabilities are enabled by a new product called SAS Grid Manager. One of the components of SAS Grid Manager is the Platform Suite for SAS which has three components:

- Process Manager – (previously Platform Job Scheduler for SAS) – serves as an interface to the SAS scheduling capability
- Grid Management Services – communicates with the Grid Manager plug-in to provide run-time monitoring/management capability within SAS Management Console
- LSF for SAS – provides mapping and load balancing of SAS processing across grid resources

COMBINING SCHEDULING AND DISTRIBUTION OF SAS JOBS TO THE GRID

While the interfaces for scheduling a SAS workflow to the grid and distributing the subtasks of a SAS job to the grid differ, the two can be used together to achieve maximum flexibility and performance. A flow could contain a job that was either generated or written with the appropriate SAS/CONNECT syntax to identify parallel subtasks for grid execution. When the flow is triggered to run, that particular job would be directed to a specific grid node by LSF for SAS and as that job executes it would create parallel tasks to be executed across the remaining grid resources. The mapping of the parallel subtasks to grid resources would also be handled by LSF for SAS. This complete scenario is depicted in Diagram 1.

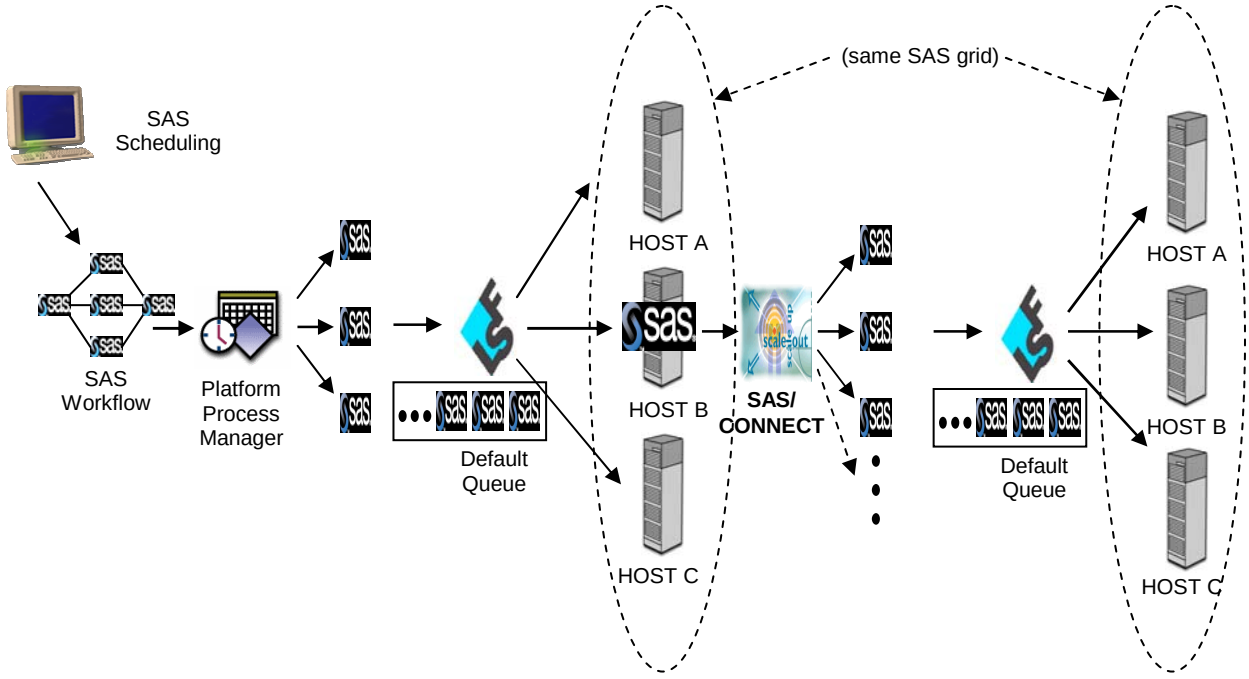

Diagram 1. Scheduling a SAS Workflow with Jobs That Distribute Components to the Grid

Note: While it may appear that there are two grids represented in Diagram 1, there is in fact a single grid represented by the HOST A, HOST B, and HOST C resources. The diagram is drawn in this manner to show the different ways that SAS processing can leverage a grid environment and how LSF for SAS is used to balance the different pieces of SAS work across all of the grid resources.

INTRODUCTION TO ADVANCED SAS GRID CONFIGURATION

Many times, the default installation and configuration of a SAS grid environment will meet the needs of your enterprise. However, if your organization has more complex policy and workload management requirements, it is possible to tune your grid configuration to meet these needs. This portion of the paper details how LSF for SAS works and some of the advanced configuration options available to fine-tune your SAS grid environment.

JOB SUBMISSION

The life cycle of a grid-enabled job starts when the SAS application submits a job to LSF for SAS. When jobs are submitted to LSF for SAS they are placed into a queue before being serviced by a targeted host. Queues contain a series of pending jobs, lined up in a defined order and waiting for their opportunity to use resources. The SAS Grid Manager install creates a default queue called the NORMAL queue. In a standard SAS grid configuration, SAS jobs are automatically submitted to the default or NORMAL queue as illustrated in Diagram 2.

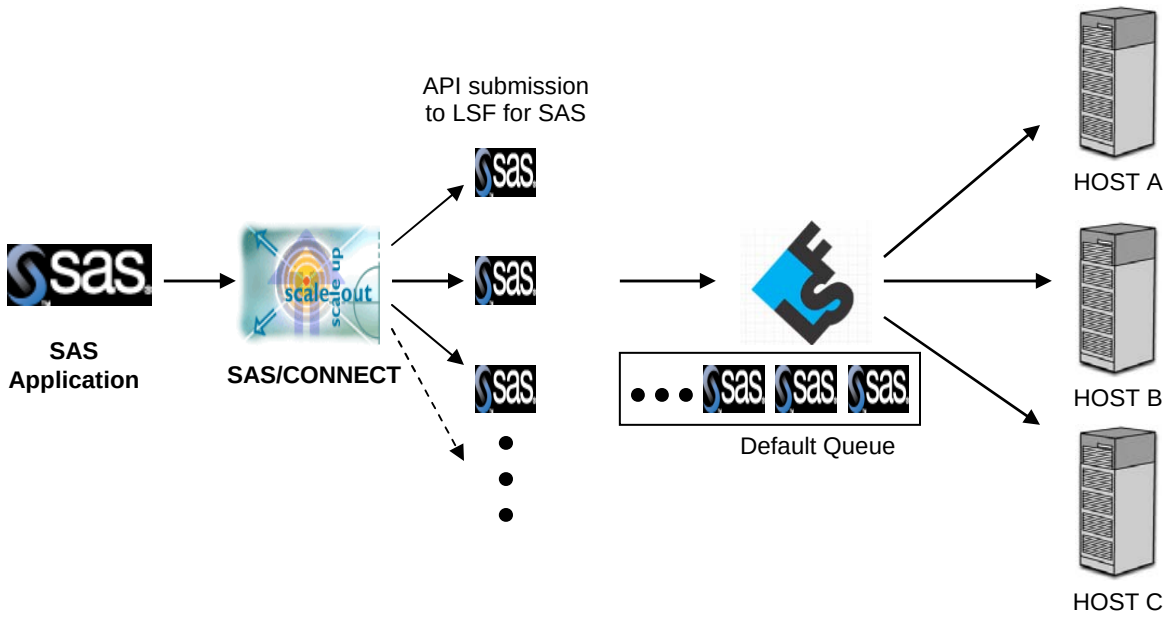

Diagram 2. Job Submission

Queues implement different job scheduling and control policies. All jobs submitted to the same queue share the same scheduling and control policy. Queues do not correspond to individual hosts; each queue can use all server hosts in the grid or a configured subset of the server hosts as illustrated in Diagram 3.

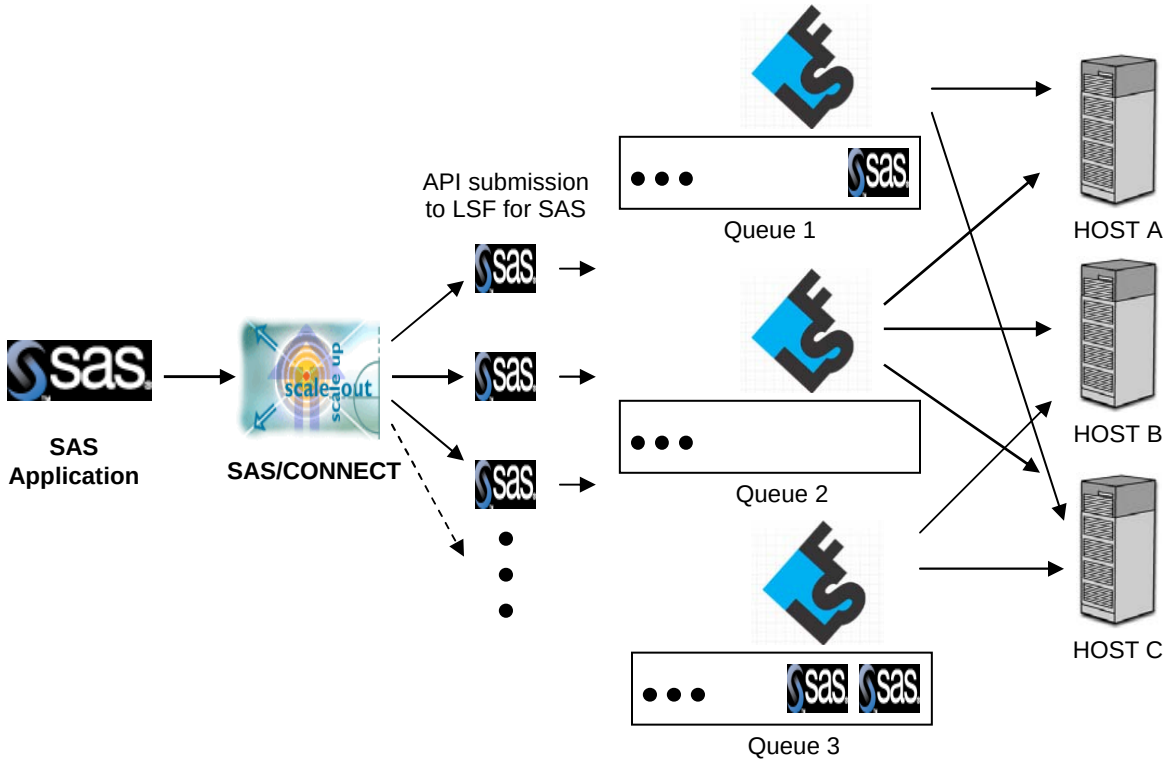

Diagram 3. Sample SAS Grid Configuration with Multiple Queues

Queues are highly configurable to allow you to customize your grid environment. Grid administrators can create multiple, customized queues to provide complete administrative control over the workload being submitted to the SAS grid. For example, queues can be assigned a priority to determine the order in which they are serviced. This allows an administrator to manage higher priority work by automatically assigning work to the appropriate priority queue based on the originating user or application. Queues can have a number of attributes, including the following:

- priority, where a larger integer indicates a higher priority
- name, which uniquely identifies the queue
- queue limits, which restrict hosts, number of jobs, users, groups, processors, and so on
- load-sharing threshold conditions, which apply load sharing to the queue
- job slots, which specify the number of job slots that a queue can concurrently use

The LSB.QUEUES file (as defined in $LSB_CONFDIR/<grid_name>/configdir) contains the queue definitions. For example:

```
Begin Queue
QUEUE_NAME = normal
PRIORITY = 30
STACKLIMIT= 2048
DESCRIPTION = For normal low priority jobs, running only if
hosts are lightly loaded.
QJOB_LIMIT = 60 # job limit of the queue
PJOB_LIMIT = 2 # job limit per processor
ut = 0.2
io = 50/240
USERS = all
HOSTS = all
End Queue
```

The queue priority defines the order in which LSF for SAS searches the queues to determine which job will be executed next. Queues are assigned a priority by the grid administrator; a higher number indicates a higher priority. Queues are serviced by LSF for SAS in order of priority from highest to lowest. If multiple queues have the same priority, LSF for SAS schedules all the jobs from these queues in first-come, first-served order. It is worth noting that in a SAS grid environment, it is not the jobs but rather the queues that have a priority associated with them. In order for more urgent work requests to receive higher priority, they need to be placed into a higher priority queue where the jobs will be serviced ahead of lower priority queues.

Submitted jobs sit in queues until they are scheduled and dispatched to a host for execution. When a job is submitted to LSF for SAS, the following factors control when and where the job starts to run:

- active time window of the queue or hosts
- resource requirements of the job
- availability of eligible hosts
- job dependency conditions
- load conditions

JOB SLOTS

A job slot is a bucket into which LSF for SAS assigns a single unit of work. Hosts are configured to have a number of job slots available and queues dispatch jobs to fill job slots. Job slots provide the flexibility to limit the number of jobs a host will execute concurrently. A job slot is a parameter that is associated to both queues and hosts.

When the job slots parameter is applied to hosts, the number of job slots assigned to a particular host is defined in the lsb.hosts file and defines the number of jobs a host will execute concurrently. Note that job slots do not control the number of CPUs that are used; that is generally determined by the host's operating system. By default, all hosts are configured with the number of job slots equal to the number of physical CPUs on the host. When the job slots parameter is applied to queues, it specifies the maximum number of job slots that a particular queue can

concurrently consume across all hosts. Used together on hosts and queues, the job slots parameter allows an administrator to have granular control over the amount of resources that a specific queue can consume across the entire grid.

ESUB FACILITY

An administrative facility called esub allows for the programmatic modification of jobs before they are submitted to LSF for SAS. It can validate, modify, or reject jobs. When SAS/CONNECT submits a job for execution to LSF for SAS, SAS/CONNECT sends environment information, such as a user ID, and also appends a number of job parameters to the submission such as the number of processors required for the job and the job name. It is sometimes beneficial to modify or add additional parameter to a SAS job submission. Submission to LSF for SAS is done through a job submission API, so these parameters are not seen by the user. However use of the esub facility makes it possible to override the default parameters as illustrated in Diagram 4.

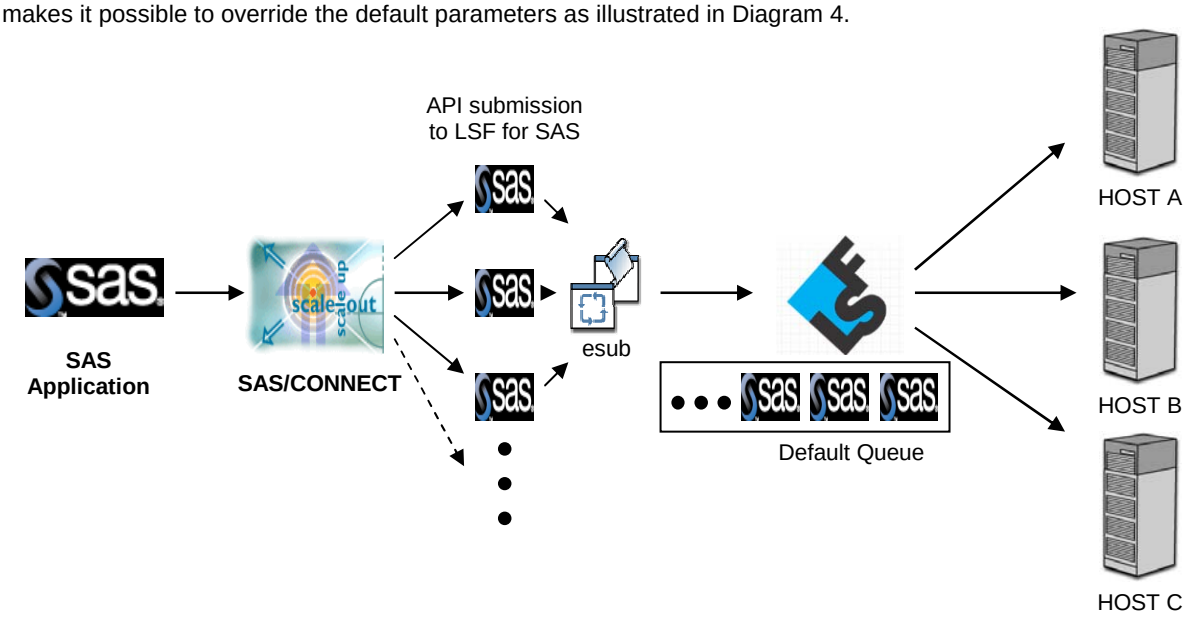

Diagram 4. Sample SAS Grid Configuration Using the esub Command

The esub command is put into the **LSF_SERVERDIR**[1] directory, where LSF for SAS checks for its existence when a job is submitted, restarted, or modified. If LSF for SAS finds an esub command, it executes that command. Whether the job is submitted, modified, or rejected depends on the logic that is built into the esub command.

It is important to differentiate between job management derived from the esub command and job management derived from queue configuration. The esub commands are executed prior to job submission to a queue and are used to manage and modify job submission parameters. Queue configuration is used to manage how jobs are allocated to hosts. The two are not mutually exclusive, and though in some cases it is possible to use the functionality of one to replace the other, the two are typically used in tandem depending on the desired outcome.

UNDERSTANDING ENVIRONMENT VARIABLES TO BRIDGE ESUB AND LSF FOR SAS

There are four environment variables in the esub execution environment:

1. LSB_SUB_PARM_FILE
   points to a temporary file containing the job parameters that the esub command reads when the job is submitted. The submission parameters are name and value pairs on separate lines, specified in the form *option_name=value.*

[1] **LSF_SERVERDIR** is the machine-dependent directory that contains all the server daemon binaries, scripts, and other utilities that are shared by all hosts of the same type. Typically, this directory is located in ***<LSF_TOP>/<version>/<arch>/etc***, where ***<LSF_TOP>*** is the location of your Platform Suite for SAS installation, ***<version>*** is the Platform LSF version number, and ***<arch>*** is the host's architecture. You will have an **LSF_SERVERDIR** for every different machine architecture that you install, although some machine architectures are similar enough to share the same binaries and, therefore, the same **LSF_SERVERDIR** directory.

The default SAS grid submission parameters result in an LSB_SUB_PARM_FILE with the following contents:

```
LSB_SUB_JOB_NAME=<SAS>*
LSB_SUB_NUM_PROCESSORS=1
LSB_SUB_MAX_NUM_PROCESSORS=1
LSB_SUB_COMMAND_LINE=<SAS>*
```

* denotes SAS application supplied value.

The list of all possible job parameters that are written to the LSB_SUB_PARM_FILE can be found in the SAS white paper "Implementing Site Policies for SAS Scheduling with Platform JobScheduler". Refer to the "For More Information" section at the end of this paper for a link to this white paper.

2. LSB_SUB_ABORT_VALUE
   specifies the exit value that esub should return if LSF for SAS is to reject the job submission.

3. LSB_SUB_MODIFY_ENVFILE
   specifies the file to which the esub should write any job environment variable changes. The variables modified should be written to this file in the same format that is used in LSB_SUB_PARM_FILE. The order of the variables does not matter. After esub runs, LSF for SAS checks LSB_SUB_MODIFY_ENVFILE for changes. If changes are found, LSF for SAS applies them to the job's environment variables.

4. LSB_SUB_MODIFY_FILE
   specifies the file to which the esub should write any submission parameter changes. The job options modified should be written to this file in the same format that is used in LSB_SUB_PARM_FILE. The order of the options does not matter. After esub runs, LSF for SAS checks LSB_SUB_MODIFY_FILE for changes. If changes are found, LSF for SAS applies them to the job.

After an esub is issued, LSF for SAS checks to see if the script exited with LSB_SUB_ABORT_VALUE. If it did not, LSF for SAS applies the changes found in the LSB_SUB_MODIFY_ENVFILE and LSB_SUB_MODIFY_FILE files, if those files exist. Depending on your site policies, you can reject a job. To reject a job, your esub command should return with LSB_SUB_ABORT_VALUE. If a job is rejected, the esub command should not write to either LSB_SUB_MODIFY_FILE or LSB_SUB_MODIFY_ENVFILE. Your esub command can make use of the LSB_SUB_ABORT_VALUE to validate a job submission by defining conditions to reject jobs.

The decision tree associated with the esub command is shown in Diagram 5.

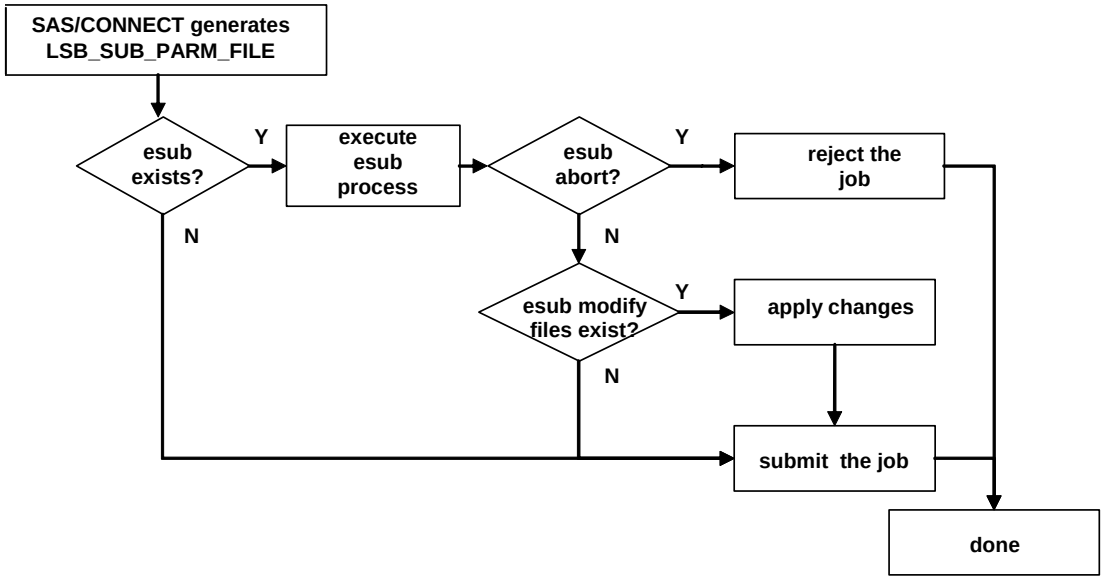

Diagram 5. Decision Tree Associated with the esub Command

MANAGING THE WORKLOAD ACROSS THE ENTERPRISE

The following sections describe a real world business scenario and the associated requirements for resource usage and policy enforcement. The configuration options presented in the previous sections of this paper will be used to alleviate the business pains and meet the specified requirements.

THE BUSINESS PAIN

John works for a financial institution that provides home equity loans. He currently gets a hard-copy report every week that graphically tells him which applicants from the multiple branches will most likely default on their loan. John has determined that he could substantially improve the level of service to his customers if he could run this report himself, on demand, and obtain the information in near-real time. In addition, John frequently uses Microsoft Excel and would prefer to view the application results in Excel.

There are several obstacles standing in John's way. A substantial amount of data preparation, data mining and integration of results must take place in order to generate this report, and these steps are very data- and computation-intensive. In addition, the IT resources that John has available to execute this application are also used by others in the organization doing data integration and data mining work. Also, this application has always been run on a weekly basis to produce a hard-copy report and John is asking to change "the way it has always been done". To effect this type of change, John will have to figure out how the IT infrastructure can be better used to accelerate his application and also to optimally balance the SAS workload across multiple groups sharing the same computing infrastructure.

The workflow of the home equity loan application is made up of many components:

- ETL jobs must be created to load multiple data feeds of loan applications from multiple sources into a data warehouse
- an ETL flow is created to subset a portion of data from the data mart for data mining purposes
- the data miners run different models over the subset of data
- the statisticians determine the best model to predict bad loans
- the selected model is incorporated into an ETL flow that can be run to loop over the loan application data from multiple branches and generate a report to indicate bad loans.

After several conversations with the IT group, it became clear that a grid environment would meet all of the goals and requirements across the organization. Migrating to a grid environment would provide John with the acceleration of his home equity loan application to enable him to respond more quickly to his customers. It would also allow multiple workloads to efficiently share the resources in order to provide acceleration to multiple applications as well as better use of the hardware resources in the grid. The IT department decided to implement the new environment in three steps, each building on the infrastructure of the previous one:

1. Establish a grid to improve performance with a single application.
2. Create policies for sharing resource capacity with multiple applications.
3. Provision additional capacity dynamically to be shared with multiple applications.

ESTABLISH A GRID TO IMPROVE PERFORMANCE WITH A SINGLE APPLICATION

The initial step is to configure a grid environment that consists of a dedicated set of machines to provide the computing power to enable acceleration of a single application. This is easily accomplished from the application perspective by using either SAS Data Integration Studio or SAS Enterprise Miner. Both solutions offer an easy-to-use point-and-click development environment that can be used to create SAS workflows. The engines generating the actual program logic behind the scene have the intelligence to automatically generate the appropriate syntax to distribute the application across the grid environment and to automatically aggregate the results back to the client environment.

There are many choices for the hardware to make up the grid environment. You could use existing hardware that is underused or perhaps unused because it is thought to be obsolete. Alternatively, you could migrate from a large symmetric multi-processor (SMP) environment to lower-cost commodity hardware such as a collection of workstations or a rack of blades. You might also choose a grid environment made up of a combination of SMP servers and commodity hardware.

An example grid environment being used to accelerate a single application is represented in Diagram 6.

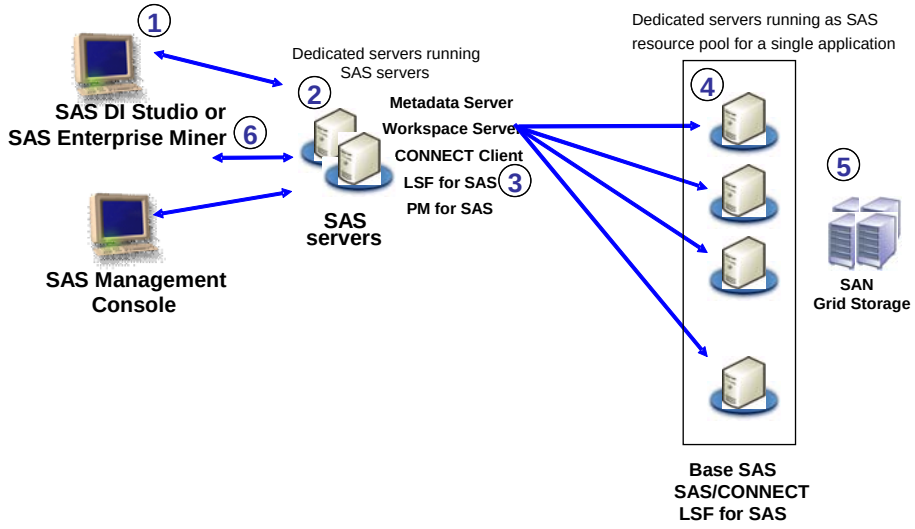

Diagram 6. Improve Performance with a Single Application

1. A SAS Data Integration Studio or SAS Enterprise Miner client requests a workload to run.
2. The request is routed through the SAS application servers.
3. The workload is dispatched to the appropriate grid nodes.
4. The ETL or data mining workload is executed on dedicated servers managed by Platform LSF for SAS.
5. The transformed data is stored in a SAS datastore.
6. The results are returned to the client via SAS servers.

CREATE POLICIES FOR SHARING RESOURCE CAPACITY WITH MULTIPLE APPLICATIONS

Recognizing that no organization runs only a single application, the second step is to create an environment that can be most effectively shared by multiple applications. It is critical that these applications be able to efficiently share the grid environment in a load-balanced and policy-based manner in order to meet the service level agreements (SLAs) of the multiple business units. Referring back to the original workflow, there are several business groups that can benefit from the acceleration that a grid environment promises. The ETL developers who are loading data from multiple input sources can do much of this loading in parallel. Much of the analysis being performed by the data miners to do model training over multiple data sources can also be executed in parallel. It is critical that John's report, which he wants to run from Microsoft Excel to launch a SAS stored process, be able to run in the grid environment in order to provide the near-real-time response that John expects.

The environment will need to be able to balance the workload across the multiple groups of users based on their computing needs. After further discussion it is determined that the different users need to be able to define and enforce policies based on four main factors:

- user priority
- time of day
- processor speed
- temporary storage space

Enforcing the policies based on these factors means that the infrastructure must be able to lend and borrow resources among the different workloads.

This environment is represented in Diagram 7.

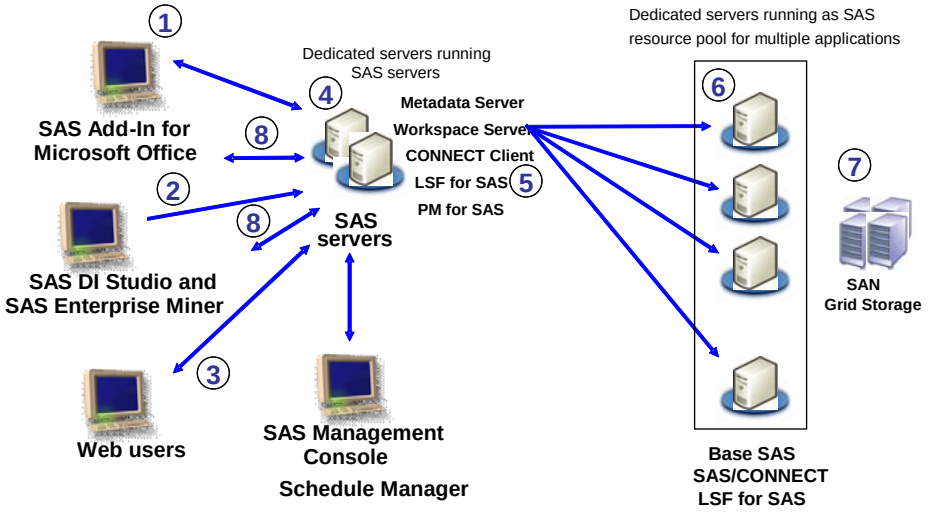

Diagram 7. Create Policies for Sharing Resource Capacity with Multiple Applications

1. A Microsoft Excel user launches a SAS stored process.
2. A SAS Data Integration Studio or SAS Enterprise Miner client requests a workload to run.
3. A BI user invokes a SAS stored process through a Web application.
4. The requests are routed through the SAS application servers.
5. The workload is dispatched to the appropriate grid nodes.
6. The entire SAS workload is balanced across the dedicated servers managed by Platform LSF for SAS.
7. The transformed data is stored in SAS datastore.
8. The results are returned to the clients via SAS servers.

PROVISION ADDITIONAL CAPACITY DYNAMICALLY TO BE SHARED WITH MULTIPLE APPLICATIONS

With multiple applications running in a shared computing infrastructure, it is easy to imagine that system load could peak at various times and resource limits could be reached trying to meet the SLAs of the entire organization. The final step is to add the ability to dynamically provision additional computing resources into the grid environment based on reaching a pre-defined load on the system. The goal is to be able to bring additional hardware into the grid at peak application load times, dynamically detect the presence of the new resources and have the applications that are running in the grid automatically begin using these additional resources. You can accomplish this by bringing additional physical boxes into the grid. However, this assumes that you have additional hardware available and ready to be used. Another option would be to take a larger SMP machine and provision multiple virtual machines (VMs) using, for example, VMware or Xen (a Linux VM product) to be available for dynamic addition to the grid. Virtual machine technology enables hardware to run more than one concurrent operating system instance. Each instance of the operating system runs its own applications as if it were the only OS on the computer. Each operating system instance is referred to as a *virtual machine.* You would also need a virtual machine manager to optimize the use of the virtual machines by dynamically balancing and controlling the allocation of virtual server resources according to service level policies. Virtual Machine Orchestrator™ (VMO) from Platform Computing is an example of a virtual machine manager. The virtual environment has the advantage of creating a virtual solution to provide additional computing resources rather than requiring you to purchase additional physical boxes. The VMs also allow you to provision clean, captured, and cloned environments easily and on demand.

This environment is represented in Diagram 8.

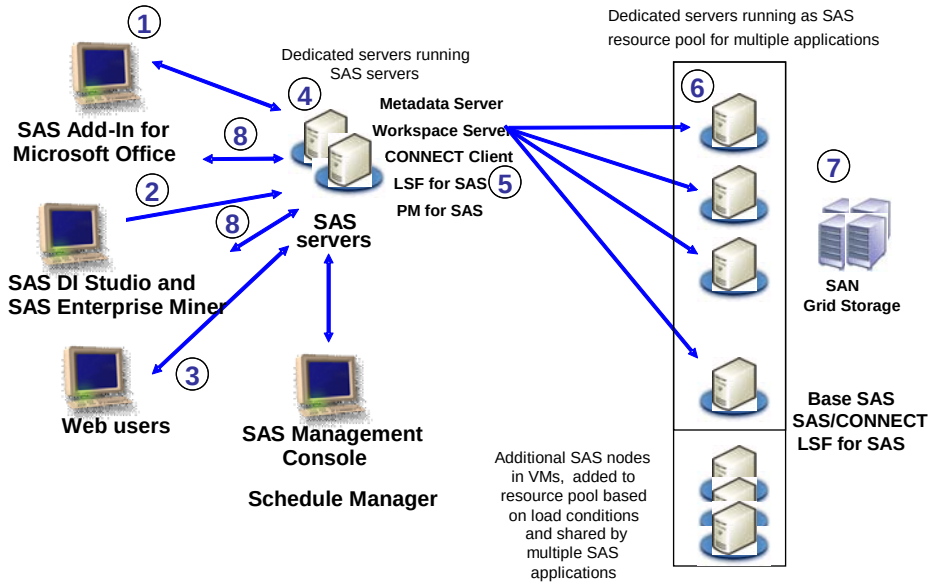

Diagram 8. Provision Additional Capacity Dynamically to Be Shared With Multiple Applications

1. A Microsoft Excel user launches a SAS stored process.
2. A SAS Data Integration Studio or SAS Enterprise Miner client requests a workload to run.
3. A BI use invokes a SAS stored process through a Web application.
4. The requests are routed through the SAS application servers.
5. The workload is dispatched to the appropriate grid nodes.
6. The entire SAS workload is balanced across the dedicated servers managed by Platform LSF for SAS.
7. The transformed data is stored in a SAS datastore.
8. The results are returned to the clients via SAS servers.
9. Additional SAS node capacity is added in VMs based on load conditions and shared by multiple applications.

IMPLEMENTATION DETAILS

This section presents the contents of some of the files and many of the details needed to configure an environment to enforce policies to share resource capacity and provision additional capacity in order to balance the workload of multiple applications.

The first step in balancing the workload among the ETL, data mining and BI users is to create the following queues:

- a queue for the ETL jobs doing the initial loading
- a queue for the data mining jobs doing model comparisons
- a queue for the execution of SAS stored processes when launched from Microsoft Excel or other BI components

The LSB.QUEUES file (as defined in $LSB_CONFDIR/<grid_name>/configdir) contains the queue definitions. The following LSB.QUEUES file defines the three queues described above:

```
Begin Queue
QUEUE_NAME   = ETL_queue
DESCRIPTION  = for SAS Data Integration Studio users
End Queue
```

```
Begin Queue
QUEUE_NAME   = BI_queue
DESCRIPTION  = for Business Intelligence users
End Queue

Begin Queue
QUEUE_NAME   = EM_queue
DESCRIPTION  = for Enterprise Miner users
End Queue
```

In addition, a best practice of selecting standard user IDs specific to each group of users is adopted. Policies can then be enforced by creating an esub to direct the jobs to the appropriate queues based on the user ID used to submit the job. The following three user IDs are used:

- sasetl – for the ETL load jobs
- sasem – for the data mining jobs
- sassrv – for the stored process launched from Microsoft Excel

The function of the esub is to examine the user ID parameters of the submitted job, route the job to the appropriate queue and add the appropriate resource as an additional job submission parameter. This causes LSF for SAS to search for a machine that has both SASApp and the appropriate additional resource defined when determining which node is the most appropriate node to execute the job. This is illustrated in Diagram 9.

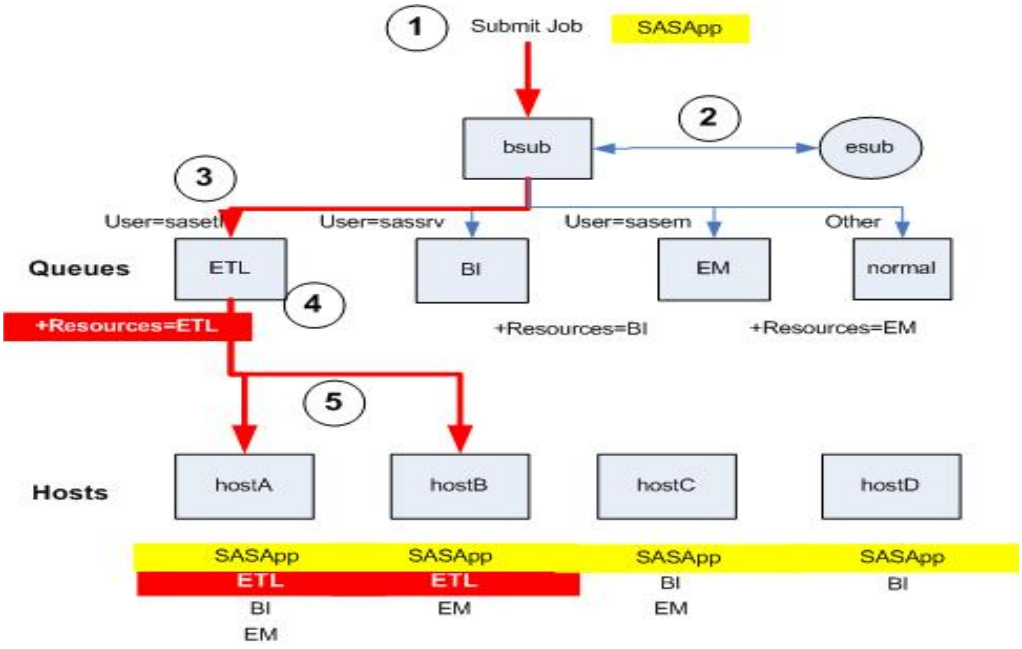

Diagram 9. Esub Queue Allocation

1. An ETL user submits a job to be executed.
2. The esub executes to determine where to route the job.
3. The job is routed to the ETL queue by the esub because the job is running under the user ID SASETL.
4. The ETL queue adds the additional resource requirement of ETL to the job.
5. A host is chosen to run the job based on having a resource of ETL and the current load.

Note: This same functionality can be accomplished from within the SAS environment (without defining an esub or queues) by specifying a workload of ETL in the SAS Data Integration Studio development environment, defining a workload of ETL in the SAS metadata and defining the ETL resource for the appropriate grid nodes. However, this is an intermediate step in a more complex policy management solution that requires the use of an esub and queues.

The esub used to direct the jobs to the appropriate queues follows:

```
#!/bin/sh
# source in the parameter file so that they can be treated as environment
variables. $LSB_SUB_PARM_FILE

# redirect stdout to stderr so echo can be used for error messages

exec 1>&2
if [ “$USER” = "sasem" ];
then
   # "ENTERPRISE_MINER" jobs run on the EM_queue
   if [ "$LSB_SUB_QUEUE" != "EM_queue" ];
   then
      # write queue change to EM_queue to the LSB_SUB_MODIFY_FILE

       echo 'LSB_SUB_QUEUE = "EM_queue"' > $LSB_SUB_MODIFY_FILE
    fi
elif [ “$USER” = "sasetl" ];
then
   # "ETL" jobs run on the ETL_queue
   if [ "$LSB_SUB_QUEUE" != "ETL_queue" ];
   then
      # write queue change to ETL_queue to the LSB_SUB_MODIFY_FILE

        echo 'LSB_SUB_QUEUE = "ETL_queue"' > $LSB_SUB_MODIFY_FILE
   fi
elif [ “$USER” = "sassrv" ];
then
   # "BI" jobs run on the BI_queue
   if [ "$LSB_SUB_QUEUE" != "BI_queue" ];
   then
      # write queue change to BI_queue to the LSB_SUB_MODIFY_FILE

        echo 'LSB_SUB_QUEUE = "BI_queue"' > $LSB_SUB_MODIFY_FILE
   fi
fi
```

As an extension to the queue allocation scenario described above, it is possible to configure the queues so that they are automatically redirected to the desired hosts using a date- or time-based policy. For example, it may be desirable to allocate more hosts to BI users during normal business hours and then shift these resources to data mining users during off peak hours. In addition, if some of the same hosts are being shared by the BI and data mining workloads during the same time period, it is possible to further manage the grid resources by associating a maximum number of job slots to each queue to ensure that one type of workload does not consume all available job slots. These concepts are illustrated in Diagram 10 and detailed in the LSB.QUEUES file that follows the diagram.

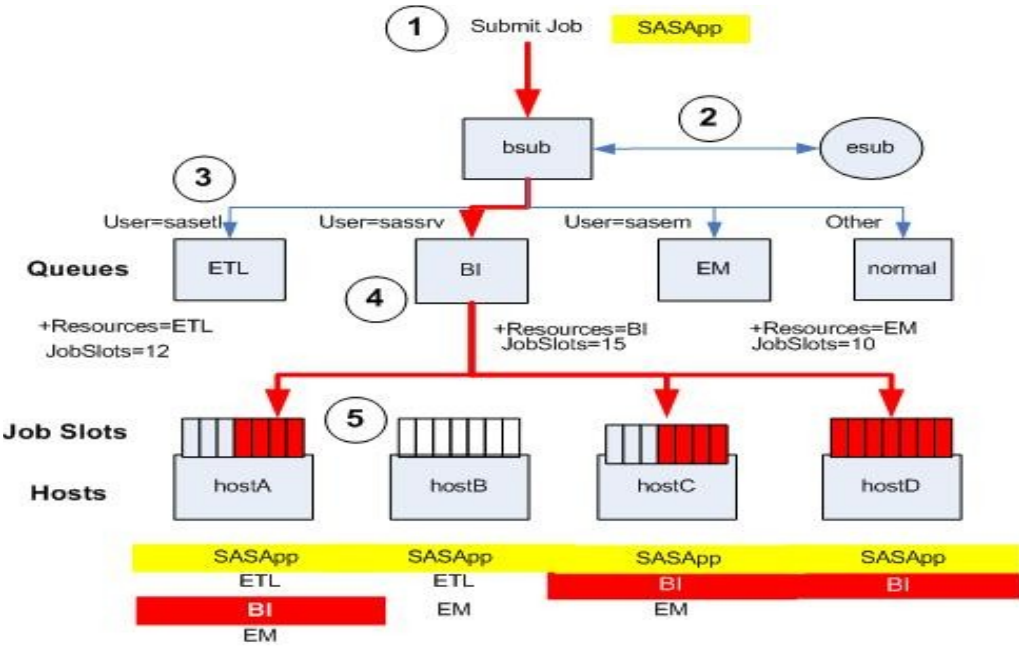

Diagram 10. Esub Queue and Host Allocation

1. A BI user submits a job to be executed.
2. The esub executes to determine where to route the job.
3. The job is routed to the BI queue by the esub because the job is running under the user ID SASSRV.
4. The BI queue adds the additional resource requirement of BI to the job. The BI queue also has a maximum number of job slots specified that the jobs routed to the BI queue can use. The number of job slots could apply to the entire grid, each host in the grid, or even individual processors within a single host – depending on the syntax used in the LSB.QUEUES file.
5. A host is chosen to run the job based on having a resource of BI, the current load and one of the fifteen BI job slots being available for use.

Assume that from 8:30a.m. to 5:00p.m., three hosts are to be made available to BI and two hosts made available to SAS Enterprise Miner. Conversely, during off-peak business hours, from 5:00p.m. to 8:30a.m., BI has two hosts and SAS Enterprise Miner has three hosts. To set up this configuration, you need to add host and time specifications to the queue definitions previously created in your LSB.QUEUES file (as defined in $LSB_CONFDIR/<grid_name>/configdir). However, because the BI and SAS Enterprise Miner workloads have at least one host in common during the specified time periods (hostA from 8:30a.m. to 5:00p.m. and hostC from 5:00p.m. to 8:30a.m.) a maximum job slot allocation per queue is used to ensure that neither workload exhausts all possible job slots. Limiting the maximum job slots available to each queue is useful either if there are common hosts allocated to the same time slots or if the time slots themselves overlap. A portion of the LSB.QUEUES file used to manage the queue configurations and running the jobs follows:

```
Begin Queue
QUEUE_NAME=BI_queue
#if time(8:30-17:00)
HOSTS             = hostA hostC hostD
#else
HOSTS             = hostC hostD
#endif
QJOB_LIMITS = 15
DESCRIPTION = for BI users
End Queue

Begin Queue
QUEUE_NAME=EM_queue
#if time(8:30-17:00)
HOSTS             = hostA hostB
```

```
#else
HOSTS             = hostA hostB hostC
#endif
QJOB_LIMITS = 10
DESCRIPTION = for EM users
End Queue
```

CONCLUSION

Today's enterprises have many different types of applications, business units and users with varying service level needs. Often these needs exceed the capacity of the current IT infrastructure with the current policies and usage patterns. Organizations are moving to virtual environments that can satisfy these needs and they require software solutions that can easily and automatically run in these environments. SAS®9 is designed to facilitate adoption of a virtual IT environment by providing grid automation to those components and solutions where appropriate by incorporating the enterprise-level scheduling and grid distribution and management capabilities provided by Platform Computing and the Platform Suite for SAS. The implementation detailed in this paper shows that SAS is a leader in both the grid and virtual computing spaces.

ACKNOWLEDGMENTS

The authors acknowledge Craig Rubendall, Director of Enterprise Computing Management at SAS Institute Inc. and Glenn Horton, Systems Developer at SAS Institute Inc., for their help in implementing the scenarios and their contributions to this paper.

FOR MORE INFORMATION

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

Introduction to Grid Computing. Scalability and Performance Community. SAS Institute Inc. Available support.sas.com/rnd/scalability/grid.

SAS Grid Computing. Technologies/Enterprise Intelligence Platform. SAS Institute Inc. Available www.sas.com/grid.

Syntax for SAS/CONNECT Grid Functions. Syntax for Grid Enablement. SAS Institute Inc. Available support.sas.com/rnd/scalability/grid/gridfunc.html.

Sample Code for Load Balancing SAS Jobs from Multiple Users. Syntax for Grid Enablement. SAS Institute Inc. Available support.sas.com/rnd/scalability/grid/gridfunc.html.

Tran, A., and R. Williams, 2004. "Implementing Site Policies for SAS Scheduling with Platform JobScheduler." Available support.sas.com/documentation/whitepaper/technical/JobScheduler.pdf.

CONTACT INFORMATION

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

Cheryl Doninger
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Phone: 919-531-7941
Email: cheryl.doninger@sas.com

Alan Wong
Platform Computing Corporation
Markham, Ontario
Phone: 905-948-4308
Email: awong@platform.com