

Paper 370-2012

RTM and SASGSUB, the Power to Know® ... what your Grid is doing

Erwan Granger, SAS Institute Inc, Cary, NC

ABSTRACT

With the release of SAS® Grid Manager 9.2 M3, two new applications are changing the way that you interact with your SAS Grid. Platform RTM for SAS gives you a GUI to manage your Grid, as well as monitor its activity and performance. SASGSUB brings you the ability to submit a SAS program to the Grid using only the command line.

Individually, each of these tools brings a whole new set of capabilities to the Grid. Together, they allow you to test your Grid configuration and validate its performance, so that you can maximize your investment in SAS Grid Manager technology and proactively manage your Grid environment.

INTRODUCTION

In many ways, SAS Grid Manager is one of the most transformative technologies introduced by SAS in the last few years. Although SAS® Foundation has been running on very powerful machines since its inception, there is now a clear shift toward distributed computing: more and more customers recognize the benefits of having multiple less powerful machines rather than a single more powerful one. This can have positive impacts on performance. It also allows for horizontal scaling and translates into a more highly available environment in case of hardware failure.

Behind the obvious benefits of increased availability, parallelized processing, and the ability to support a larger number of users within a single centralized environment, there is also the concept of built-in governance, which might bring the most benefit of all. Some SAS users might not see this as a great improvement, as they are used to a great deal of independence on their SAS environment, unfettered by the bonds of accountability. That is, until they are found to be the one responsible for consuming all the resources and crashing the server. The governance features will definitely please the IT department, reassuring them that their servers are being used to their full potential, but not more.

However, one must remember that, as Uncle Ben told Peter Parker, "With great power comes great responsibility". Indeed, that is good advice, and not just if you are Spiderman: Some SAS Administrators have now mutated to become SAS Grid Administrators (no official costume yet) and therefore have increased control over their users. They can for example, force them to use certain Grid Nodes, give them higher or lower priorities (based on bribes received), pause and resume their jobs, or altogether lock them out of the Grid during certain periods of time.

While SAS Grid Administrators typically use their "powers" for good, these "powers" can be intimidating at first, and even overwhelming. They can also, in certain cases backfire, and work in unexpected ways; that is where this paper comes in.

First, it explores Platform RTM for SAS® through some of its features and functionalities. Following that, different aspects of SASGSUB are described together with sample code to illustrate its many uses. Finally, this paper dives further into how their combined use can help SAS Grid Administrators in configuring, testing, validating, tuning, and monitoring their SAS Grid in a practical and structured fashion.

PLATFORM RTM FOR SAS

Platform RTM for SAS is a Web interface to administer and monitor SAS Grid Manager Environments. It is a Graphical User Interface (GUI) that interacts with Platform LSF (Load Sharing Facility) to simplify its administration. It also includes features that are not part of the standard LSF toolkit. Platform RTM for SAS is a customized version of Platform RTM created by Platform Computing specifically for SAS Grid Manager Environments. From here onwards, any reference to RTM should be taken as meaning "Platform RTM for SAS".

RTM is available at no extra cost to all customers who have licensed SAS Grid Manager. It is not included in the Software Depot, but is found instead on the SAS Download Site. (See the Further Reading section for link). Because this paper does not cover architecture and installation topics, only a short list of recommended practices regarding RTM are given here:

- Be sure to download, read, and understand the README file before downloading the RTM installer.
- For full feature set:
 - If you have a Windows Grid, install RTM on Windows

RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

- If you have a UNIX or Linux Grid, install RTM on Linux
- Use the latest version (RTM 2.0.7 at time of writing this paper)

WHAT RTM CAN DO

The different features of RTM can be classified in three distinct categories: Monitoring, Management, and Configuration. But before diving further into those, the screenshots in Figure 1 capture the essence of how RTM can change the way the SAS Grid Administrators work:

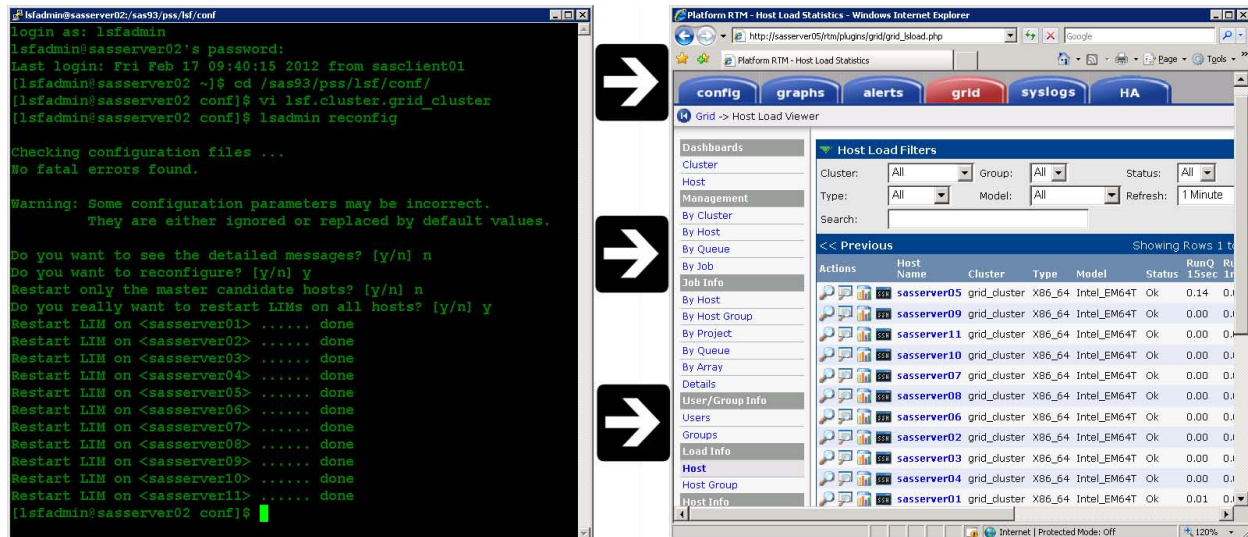


Figure 1: Evolution of SAS Grid Administration

Monitoring Features

In a SAS Grid Environment, the backbone that holds the machines together is Platform LSF. Therefore, each machine participating in the SAS Grid (or the LSF Cluster) has a number of LSF processes running. These processes communicate with each other constantly. One of these processes is called Load Information Manager (LIM) and its purpose, as the name suggests, is to collect load information from the machines (CPU, memory, and so on...), as well as LSF-centric statistics (number of running and pending jobs, statuses of hosts, queues, and so on...).

LSF collects this information so that it can make informed decisions on where to run jobs or what action (kill/pause) to take, if any is needed. There is no reason for LSF to keep this information for long, and so it is quickly discarded.

This is exactly where RTM comes in. It polls LSF regularly to collect all of that information and stores it for later use. This data is then processed by RTM to build graphs, tables, dashboards, and interfaces. Because of this, its strength does not lie in displaying the live activity of the Grid, but more in allowing a historical view of what went on inside your SAS Grid. For example, it makes it easy to see what percentage of the capacity is being used. The following screen captures illustrate some of these metrics.

RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

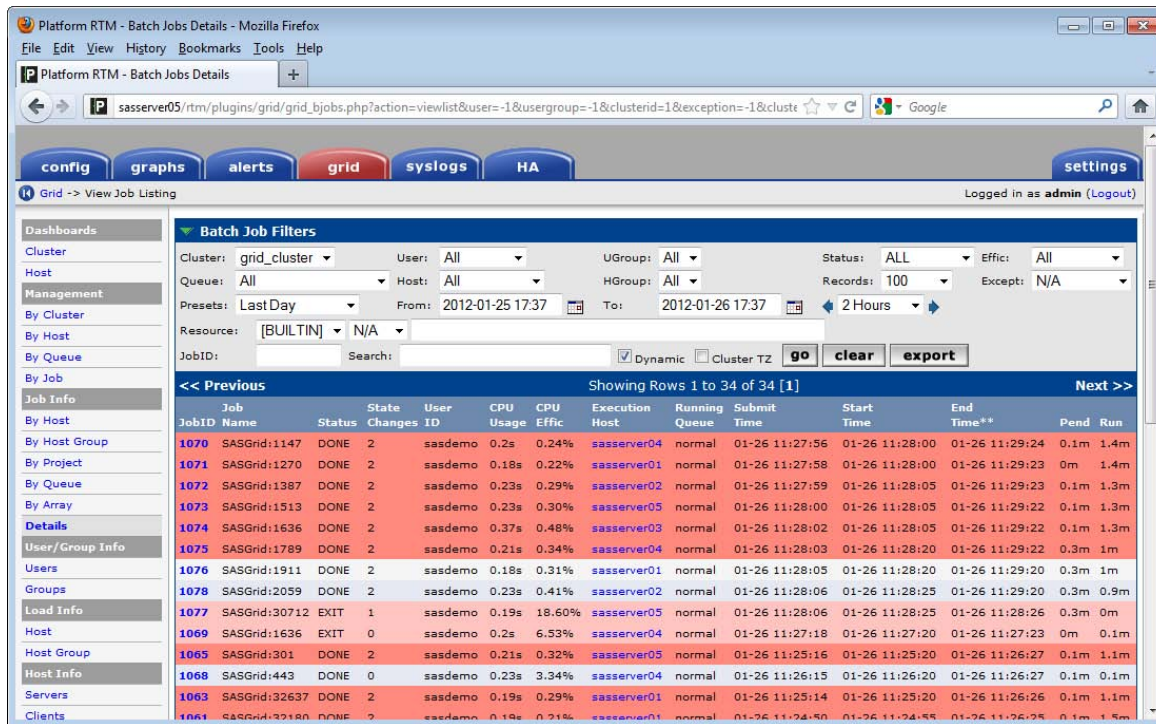


Figure 2: Viewing Grid Jobs and Their Statuses in RTM

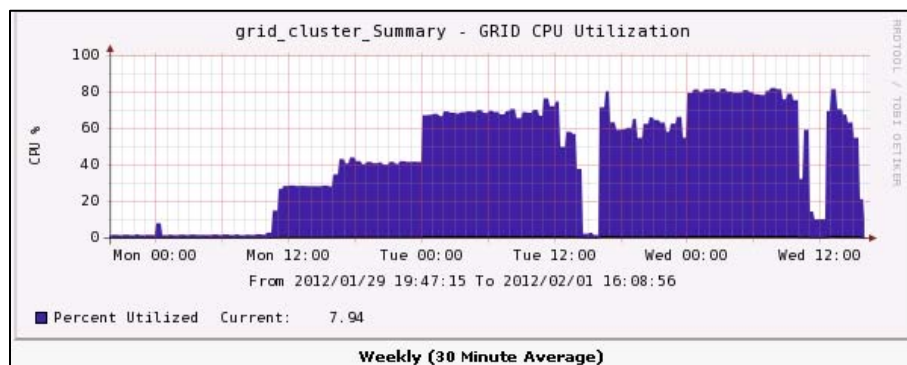


Figure 3: Overall Grid CPU Utilization in RTM over a Three Day Period

Management Features

RTM's monitoring capabilities allow SAS Grid Administrators to know what is going on inside the SAS Grid, but what good is that if they cannot do anything about it?

Therefore, RTM provides them with the ability to interact with the Grid, so they can take the required actions. These actions are referred to here as the Grid management actions.

You can, for example, kill someone's job (because it has been running for much longer than it was supposed to), close a host (because you have to apply an OS patch), or inactivate a queue (because you need to delay the start of the ETL processing tonight). These types of actions affect the environment and the way it functions at different time points, but they do not modify its underlying configuration.

The following screenshots illustrate some of these Grid management actions:

RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

Grid -> Job Control

Logged in as admin (Logout)

Batch Job Filters

Cluster: grid_cluster User: sasdemo UGroup: All Status: RUNNING Eff: All

Queue: All Host: All HGroup: All Records: 100 Except: N/A

Resource: [BUILTIN] N/A

JobID: Search: ☐ Dynamic ☐ Cluster TZ

<< Previous Showing Rows 1 to 5 of 5 [1] Next >>

Job ID	Name	Status	State Changes	Nice Level	User ID	Mem Usage	VM Size	CPU Usage	CPU Eff	Execution Host	Running Queue	Submit Time	Start Time	End Time**	Pend Run	
8192	833:sasdemo:Partial	RUNNING	2	20	sasdemo	23.91M	351.14M	12.83h	99.47%	sgcwin02018.race.sas.com	normal	02-02 23:55:32	02-02 23:55:37	-	0.1m 12.9h	<input checked="" type="checkbox"/>
8193	832:sasdemo:Partial	RUNNING	512	20	sasdemo	23.87M	411.13M	4.35h	97.26%	sgcwin02018.race.sas.com	normal	02-02 23:55:36	02-02 23:55:42	-	0.1m 4.5h	<input type="checkbox"/>
8188	834:sasdemo:Partial	RUNNING	2	20	sasdemo	23.88M	349.14M	12.84h	99.53%	sgcwin02015.race.sas.com	normal	02-02 23:55:12	02-02 23:55:17	-	0.1m 12.9h	<input type="checkbox"/>
8189	834:sasdemo:Partial	RUNNING	2	20	sasdemo	23.86M	349.14M	12.86h	99.69%	sgcwin02014.race.sas.com	normal	02-02 23:55:12	02-02 23:55:17	-	0.1m 12.9h	<input type="checkbox"/>
8191	833:sasdemo:Partial	RUNNING	504	20	sasdemo	23.84M	347.13M	4.37h	99.93%	sgcwin02015.race.sas.com	normal	02-02 23:55:20	02-02 23:55:22	-	0m 4.4h	<input type="checkbox"/>

<< Previous Showing Rows 1 to 5 of 5 [1] Next >>

Warning Efficiency Alarm Efficiency Flapping Dependencies Invalid Dependencies Exited Exclusive Interactive

Last Refresh : 12:49:35 pm

Choose an action:

Set First in Queue
Set Last in Queue
Switch Queue
Run Now
Suspend Job
Resume Job
Kill Job
Kill Job

Figure 4: Terminating a Job

Application Dashboard

Logged in as admin (Logout)

Application Filters

Cluster: All Running Host: All Status: All Refresh: 1 Minute

Search:

Application List

<< Previous Showing Rows 1 to 4 of 4 [1] Next >>

Process ID	Application Name**	Application Version	Cluster	Status	Primary Host	Failover Host	Running Host	Start Time	
3065	FrameWorkServerLev1		grid_cluster	ACTIVE	sasserver02	sasserver02	sasserver02	Thu Jan 26 10:10:30 2012	<input type="checkbox"/>
30569	MetaLev1		grid_cluster	ACTIVE	sasserver01	sasserver01	sasserver01	Thu Jan 26 10:09:34 2012	<input type="checkbox"/>
3064	OlapLev1		grid_cluster	ACTIVE	sasserver02	sasserver02	sasserver02	Thu Jan 26 10:10:30 2012	<input type="checkbox"/>
6972	RemServLev1		grid_cluster	ACTIVE	sasserver05	sasserver05	sasserver05	Thu Jan 26 10:10:30 2012	<input checked="" type="checkbox"/>

<< Previous Showing Rows 1 to 4 of 4 [1] Next >>

Choose an action:

Restart
Restart Migrate
Stop
Kill

Figure 5: Stopping a Highly Available SAS Service

Configuration Features

The configuration features of RTM encompass any change that permanently modifies the configuration and behavior of the SAS Grid, be it through host parameters, by creating a new queue, by changing host group membership, by modifying the MBD_SLEEP_TIME or any other changes that require a reconfiguration of the LSF cluster.

The configuration changes in LSF are traditionally done by editing the text files that make up the Grid configuration and issuing some LSF commands (badmin reconfig and lsadmin reconfig). RTM changes the way this is done by providing a user-friendly interface that makes these tasks easier and more intuitive. To make a comparison that everyone reading this paper should understand, RTM is to LSF as SAS® Enterprise Guide® is to Base SAS®: RTM does not add features to LSF, aside from making it more intuitive and seamless. As a GUI, it also does not cover 100% of the things that can be done in the "old school" command-line and configuration-file fashion, but the latest release of RTM (2.0.7) addresses all of the most commonly used settings and is enough for most common scenarios.

RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

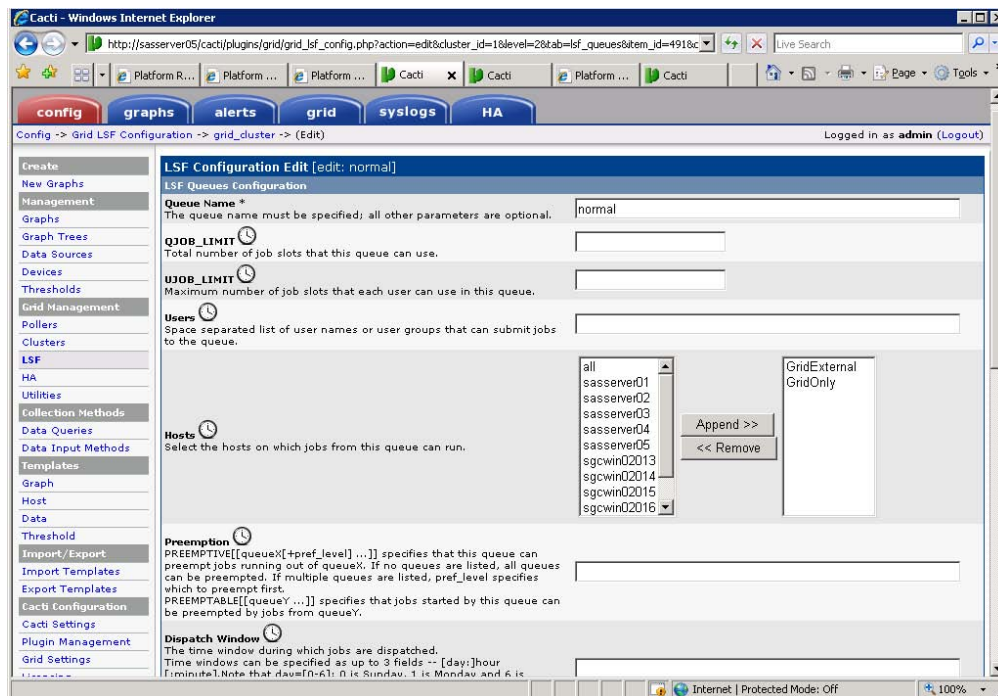


Figure 6: Updating the Definition of the Queue "Normal"

WITHOUT RTM, YOU ARE MISSING OUT

As mentioned previously, a SAS Grid Manager environment can be managed and configured without the help of Platform RTM for SAS. There is no loss of functionality by doing so. RTM should nevertheless make your job as a SAS Grid Administrator easier and more intuitive.

The monitoring features outlined previously are unique to RTM and currently cannot be reproduced without it: The LSF poller constantly queries the LIM, and stores the retrieved information in the MySQL database. Some of this information is also rolled up and stored into RRD files that are then used to build the graphs available in RTM. The host-based load information collected in RTM is sometimes seen as redundant by customers who already have another monitoring software, like Tivoli, running on their servers. While that can be true for the host-level information, only RTM can collect LSF job information out of the box.

BEYOND THE BASICS

Load versus Apply

When using RTM to configure your SAS Grid environment, it is important to understand the process by which it interacts with the rest of the Grid.

There are two key actions to be performed in RTM:

- Load configuration from Cluster (depicted in Figure 7)
- Apply settings to Cluster: <<cluster_name>> (depicted in Figure 8)

RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

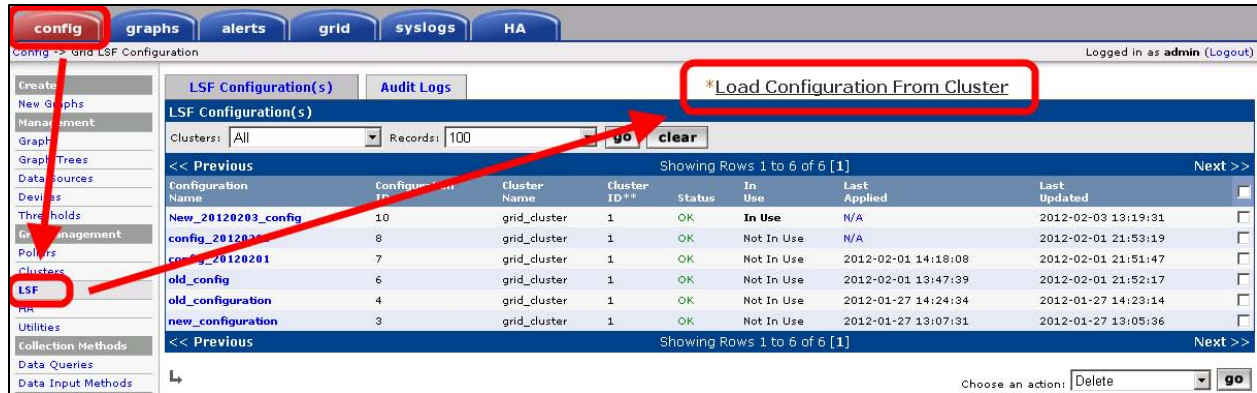


Figure 7: Loading Configuration from Cluster in RTM

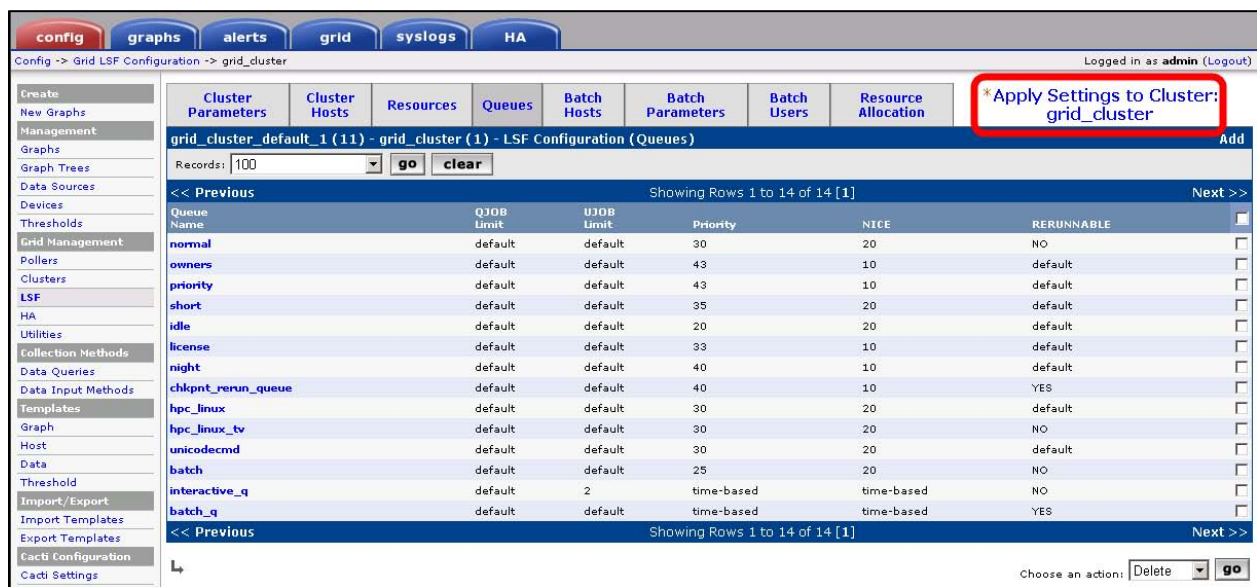


Figure 8: Applying Settings to Cluster in RTM

Once you start using RTM as the tool to configure your Grid, recommended practice is to only use RTM, and to no longer modify the LSF configuration files manually. If you ignore this advice, you might find yourself wondering where your manual changes have gone since RTM can overwrite a manual change in a future update.

But there might also be some settings that cannot be controlled from RTM that have to be modified "manually".

In that case, having a better understanding of how RTM works with LSF to perform these configurations is key to a smooth Grid operation. Figure 9 shows a simplified representation of that process.

RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

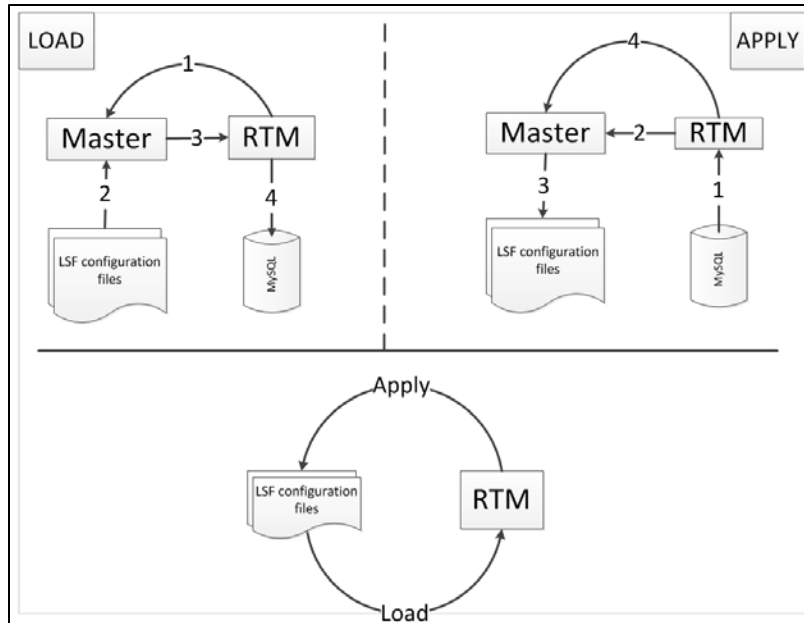


Figure 9: RTM Synchronization Process (High Level)

Loading process:

1. RTM contacts the Master of the cluster
2. The LSF configuration files are read
3. Their information is passed back to RTM
4. RTM stores this information in its MySQL database

Applying process:

1. RTM gathers the data from the MySQL database
2. RTM passes this information to the Master of the Cluster
3. The LSF configuration files are updated
4. RTM instructs the cluster to reconfigure itself (re-read configuration files)

As depicted in the bottom section of Figure 9, the Load/Apply actions effectively synchronize configuration between LSF and RTM. It is up to the SAS Grid Administrator to always remember to synchronize their changes in the right direction.

Managing Multiple Configurations

One very powerful feature of RTM is the fact that it stores the cluster information inside a database. Because of this, it is trivial to keep multiple LSF configurations inside the database simultaneously. Therefore, you are able to switch from one configuration to the other very quickly. The same result can be achieved by taking named copies of all the LSF configurations files, modifying them and moving them back and forth, in and out of the right folders. However, this approach is time consuming, error prone, and inelegant.

By loading the configuration from cluster into RTM multiple times, you can create multiple lines in RTM representing equivalent but differently named configurations. It is then easy to rename them to make them more meaningful and to switch back and forth between configurations. In the example below (Figure 10), the cluster is currently using the configuration called config_20120202. If there is a need to revert to the way the cluster was configured the previous day that can be done by checking the box next to config_20120201 and selecting "Apply Configuration".

RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

Config -> Grid LSF Configuration

Logged in as admin (Logout)

LSF Configuration(s) Audit Logs *Load Configuration From Cluster

Clusters: All Records: 100 go clear

<< Previous Showing Rows 1 to 5 of 5 [1] Next >>

Configuration Name	Configuration ID	Cluster Name	Cluster ID**	Status	In Use	Last Applied	Last Updated	
config_20120202	8	grid_cluster	1	OK	In Use	N/A	2012-02-01 21:53:19	<input checked="" type="checkbox"/>
config_20120201	7	grid_cluster	1	OK	Not In Use	2012-02-01 14:18:08	2012-02-01 21:51:47	<input type="checkbox"/>
old_config	6	grid_cluster	1	OK	Not In Use	2012-02-01 13:47:39	2012-02-01 21:52:17	<input type="checkbox"/>
old_configuration	4	grid_cluster	1	OK	Not In Use	2012-01-27 14:24:34	2012-01-27 14:23:14	<input type="checkbox"/>
new_configuration	3	grid_cluster	1	OK	Not In Use	2012-01-27 13:07:31	2012-01-27 13:05:36	<input type="checkbox"/>

<< Previous Showing Rows 1 to 5 of 5 [1] Next >>

Choose an action: Delete go

Delete
Rename
View Configuration
Apply Configuration

Figure 10: Multiple Configurations Seen in RTM

Managing Multiple Clusters

A single RTM installation can also be used to manage multiple clusters. For example, if a customer has three different instances of LSF (Dev, Test, Prod), you can either use one RTM to manage, monitor, and configure all three or, if you want full separation, deploy three different RTM instances on different machines.

However, in the situation where one RTM is connected to multiple grids, the following are required:

1. The RTM machine be defined as a client in all clusters.
2. The RTM machine not be defined as a Server in any of the clusters.

Care has then to be taken as to which Cluster is selected when you are performing actions in RTM.

A Few Tips

The default display in RTM does not always contain the best set of information. Here are a few very useful nonstandard settings:

RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

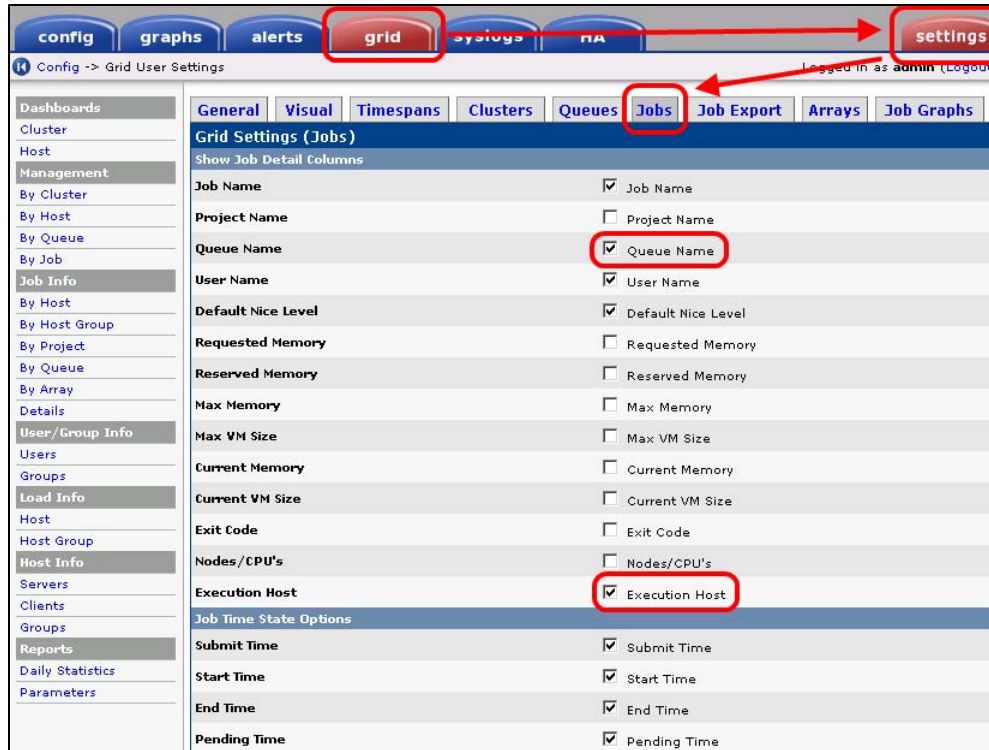


Figure 11: Adding Queue Name and Execution Host to the Job Listings Displays

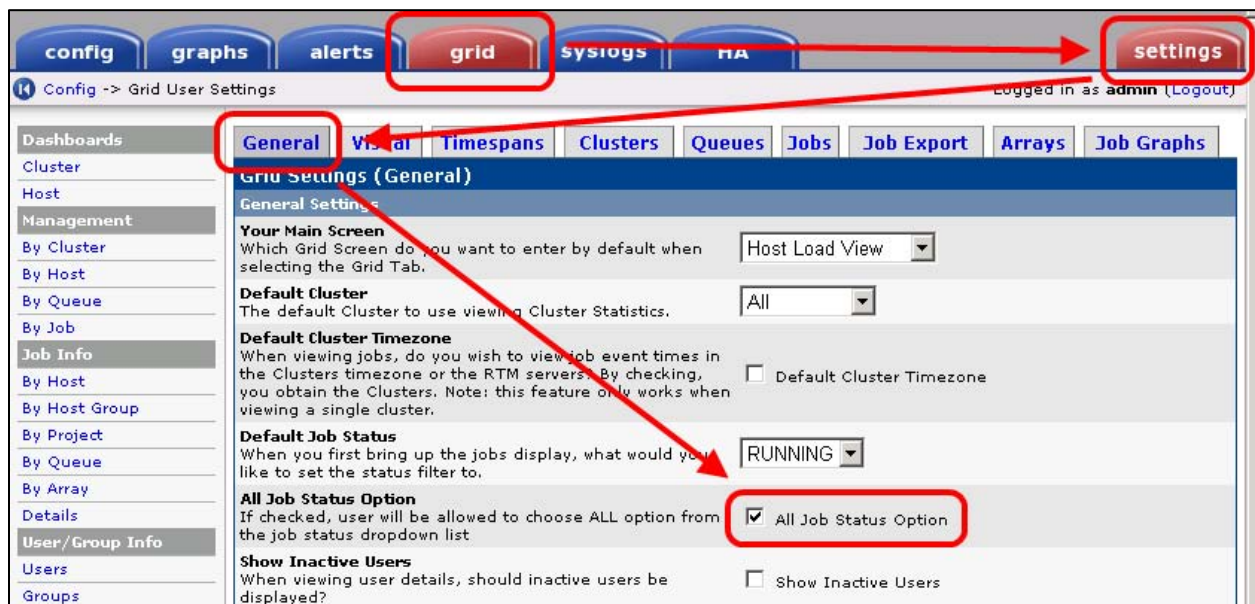


Figure 12: Adding an "All Status" Option to the Job Filters

If you look back at the screenshot in Figure 2, you will notice that the queue and execution host are visible, and that jobs with different statuses are displayed. That is due to the changes highlighted here.

RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

SASGSUB

This paper goes the extra mile to provide entertainment for audiences at both ends of the spectrum. So, if you cringed at the richness of the RTM interface and generally long for the "good old days" when you did not even need a mouse to use a computer, then you will certainly like SASGSUB.

AN END-USER TOOL FOR BATCH SUBMISSIONS

SASGSUB (also called the SAS Grid Manager Client Utility) is a Command-Line based application that allows you to submit jobs to the Grid. With SASGSUB, SAS jobs run in a complete stand-alone fashion. SASGSUB is not meant to replace other standard batch submission methods that are typically performed with the help of Process Manager and the SAS DATA Step Batch server. Instead, its role is to complement it by giving advanced users and administrators the ability to fire and forget SAS programs onto the Grid, in an ad hoc way.

By its nature (thin-client), SASGSUB might not be considered user friendly by people who are used to the rich Graphical User Interfaces that SAS offers (such as SAS Enterprise Guide or SAS® Enterprise Miner™). You should therefore carefully explore the features of SASGSUB in order to make the best decision on how to leverage it within your own Grid environment.

For SASGSUB to be able to communicate with the LSF cluster, the machine on which it is installed has to be defined as part of that cluster, either as a Server or a Client. This can have an impact on the software deployment strategy. However, it is important to note that SAS Foundation does not have to be installed on the machine for SASGSUB to work.

Despite being a very simple client for SAS Grid Manager, SASGSUB can be quite powerful, as described in the rest of this paper.

ALTERNATIVES TO SASGSUB

In order to perform batch submissions to the Grid, there are some alternatives that can be considered.

bsub

bsub is the traditional LSF client used to submit jobs to the Grid. Anyone who has set up an LSF cluster in the past has probably used the command "bsub sleep 30" to see if simple jobs could be started on the Grid, outside of SAS.

While easy to use, bsub has two major shortcomings:

1. The OEM LSF license that comes with SAS Grid Manager does not allow you to use bsub beyond simple testing. In practice, that is implemented by the fact that any job submitted to the SAS Grid through bsub is automatically killed after 30 seconds.
2. Unlike SASGSUB, bsub is not integrated with the SAS Metadata. This removes an important layer of security and can allow users not defined in Metadata to use the SAS Grid.

So, unless a full LSF license is obtained, bsub is not a good alternative to SASGSUB. Also, while the OEM LSF license grants unlimited installations of the LSF client pieces, the full LSF license might not.

Platform Process Manager

Platform Process Manager is the scheduler that works in conjunction with LSF. It is the piece of software that knows that a given flow or a given job has to run at a certain time or if a certain event happens. It is one of the most powerful ways of using the Grid because it is so tightly integrated with SAS, the SAS® Management Console, and LSF.

However, Process Manager is not traditionally a tool that all users are allowed to use. It is usually reserved to the few people who are in charge of aggregating all different batch jobs and scheduling them according to priorities and dependencies. In other words, Process Manager is used to run recurring programs, not ad hoc or on-demand ones.

Because it takes a few steps to perform the required actions (write program, deploy for scheduling, create new flow, insert job into flow, schedule, or run flow), Process Manager is not a very good alternative to SASGSUB.

RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

UNDER THE HOOD

SASGSUB is a very small (<500 KB) utility that allows users to submit programs or commands to the SAS Grid without the need to have Foundation SAS or SAS/CONNECT® installed. It does require that the machine on which it is installed be part of the LSF cluster and that the LSF client binaries are installed on it. This can be challenging when trying to install SASGSUB on hundreds of desktops. The use of response files, both for the SASGSUB installation and the LSF installation can streamline that installation process. Also, having to define all client machines in the LSF cluster file can be impractical. The use of the FLOAT_CLIENTS and FLOAT_CLIENT_ADDR_RANGE parameters in LSF also comes in handy when the number of client machines becomes large.

SASGSUB consists of some binaries (<sas_home>\SASGridManagerClientUtility\9.3\), a configuration file (<config_dir>\Lev1\Applications\SASGridManagerClientUtility\9.3\sasgsub.cfg), and a script (<config_dir>\Lev1\Applications\SASGridManagerClientUtility\9.3\sasgsub.cmd or sasgsub). The .cfg file, created by the SAS Deployment Wizard, contains the parameters that SASGSUB needs. Figure 13 shows some of the most common and important parameters, typically found in the .cfg file.

```
-METASERVER      sasserver01
-METAPORT        8561
-METAREPOSITORY  Foundation
-METAUSER        sasdemo
-METAPASS        _PROMPT_

-GRIDWORK /sas93/sharedwork/gridshared

-GRIDAPPSERVER  SASApp

-GRIDLICENSEFILE '/sas93/sharedwork/gridshared/SAS93_ORDER_SITENUM.txt'
```

Figure 13: Selected SASGSUB Parameters

These parameters can all be overridden at submission time if necessary.

Submitting a Job with SASGSUB

In order to illustrate how SASGSUB works, it is best to look at standard SASGSUB submission, from beginning to end:

```
[sasdemo@sasserver03 ~]$
/sas93/config/compute/Lev1/Applications/SASGridManagerClientUtility/9.3/sasgsub -
gridsubmitpgm /sas93/resources/paper/sas_programs/sleep5.sas -GRIDJOBNAME Sleep5 -
GRIDJOBPTS 'queue=normal'

SAS Grid Submit Utility Version 9.30 (build date: May 11 2011)
Copyright (C) 2009-2011, SAS Institute Inc., Cary, NC, USA. All Rights Reserved
Please enter the metadata password:
Job <2860> is submitted to queue <normal>.
Job ID:          2860
Job directory:   "/sas93/sharedwork/gridshared/sasdemo/SASGSUB-2012-01-
30_11.50.11.619_Sleep5"
Job log file:    "/sas93/sharedwork/gridshared/sasdemo/SASGSUB-2012-01-
30_11.50.11.619_Sleep5/sleep5.log"
[sasdemo@sasserver03 ~]$
```

Figure 14: Initial Submission of SAS Program with SASGSUB

In Figure 14, the program sleep5.sas is submitted to the Grid using SASGSUB. The name of the job is Sleep5, it is sent to the queue called "normal". Because the METAPASS parameter value is _PROMPT_ (see Figure 13) SASGSUB prompts the user for a password. The job is assigned the ID 2860 by LSF, and a job directory is created.

Querying Job Status

```
[sasdemo@sasserver03 ~]$
/sas93/config/compute/Lev1/Applications/SASGridManagerClientUtility/9.3/sasgsub -
GRIDGETSTATUS 2860

SAS Grid Submit Utility Version 9.30 (build date: May 11 2011)
Copyright (C) 2009-2011, SAS Institute Inc., Cary, NC, USA. All Rights Reserved
```

RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

```
Please enter the metadata password:
 2860 (Sleep5) is Running: Submitted: 30Jan2012:11:49:51, Started:
30Jan2012:11:49:56 on Host sasserver04
[sasdemo@sasserver03 ~]$
```

Figure 15: Querying the Job Status with SASGSUB

In Figure 15, the parameter GRIDGETSTATUS used in conjunction with the job ID informs the user that the job is currently running on host sasserver04.

```
[sasdemo@sasserver03 ~]$
/sas93/config/compute/Levl/Applications/SASGridManagerClientUtility/9.3/sasgsub -
GRIDGETSTATUS 2860

SAS Grid Submit Utility Version 9.30 (build date: May 11 2011)
Copyright (C) 2009-2011, SAS Institute Inc., Cary, NC, USA. All Rights Reserved
Please enter the metadata password:
 2860 (Sleep5) is Finished: Submitted: 30Jan2012:11:49:51, Started:
30Jan2012:11:49:56 on Host sasserver04, Ended: 30Jan2012:11:54:57, RC:0
[sasdemo@sasserver03 ~]$
```

Figure 16: Job Has Completed

Eventually, the output changes to inform the user that the job has completed successfully: (Figure 16).

```
[sasdemo@sasserver03 ~]$ ls -al /sas93/sharedwork/gridshared/sasdemo/SASGSUB-2012-
01-30_11.50.11.619_Sleep5
total 128
drwxr-xr-x 2 sasdemo sas 4096 Jan 30 11:54 .
drwxr-xr-x 84 sasdemo sas 102400 Jan 30 11:49 ..
-rw-r--r-- 1 sasdemo sas 17 Jan 30 11:49 host.sasserver04
-rwxr-xr-x 1 sasdemo sas 2 Jan 30 11:49 job.id.2860
-rw-r--r-- 1 sasdemo sas 2227 Jan 30 11:54 sleep5.log
-rwxr-xr-x 1 sasdemo sas 36 Jan 30 11:49 sleep5.sas
-rw-r--r-- 1 sasdemo sas 8 Jan 30 11:54 time.end.0
[sasdemo@sasserver03 ~]$
```

Figure 17: Content of the Directory Created by the SASGSUB Submission

Getting the Results

The directory that was created by SASGSUB for this submission contains not only the actual program that was run, but also the log file resulting from it. It also contains other files indicating the host on which it was executed (sasserver04), the ID that was given to the job by LSF (2860), and the time the program finished running (timestamp of the time.end.0 file: 11:54 am). These files are used by SASGSUB for its internal tracking and are subject to changes in the future.

```
[sasdemo@sasserver03 ~]$
/sas93/config/compute/Levl/Applications/SASGridManagerClientUtility/9.3/sasgsub -
GRIDGETRESULTS 2860

SAS Grid Submit Utility Version 9.30 (build date: May 11 2011)
Copyright (C) 2009-2011, SAS Institute Inc., Cary, NC, USA. All Rights Reserved
Please enter the metadata password:
 2860 (Sleep5) is Finished: Submitted: 30Jan2012:11:49:51, Started:
30Jan2012:11:49:56 on Host sasserver04, Ended: 30Jan2012:11:54:57, RC:0
Moved job information to "/.SASGSUB-2012-01-30_11.50.11.619_Sleep5"
[sasdemo@sasserver03 ~]$ cd SASGSUB-2012-01-30_11.50.11.619_Sleep5
[sasdemo@sasserver03 SASGSUB-2012-01-30_11.50.11.619_Sleep5]$ ls -al
total 20
drwxr-xr-x 2 sasdemo sas 4096 Jan 30 11:58 .
drwx----- 7 sasdemo sasdemo 4096 Jan 30 11:58 ..
```


RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

```
-rwxr-xr-x 1 sasdemo sas      220 Jan 30 11:58 job.info
-rwxr-xr-x 1 sasdemo sas     2227 Jan 30 11:58 sleep5.log
-rwxr-xr-x 1 sasdemo sas       36 Jan 30 11:58 sleep5.sas
[sasdemo@sasserver03 SASGSUB-2012-01-30_11.50.11.619_Sleep5]$
```

Figure 18: Getting the Results Back After Job Has Run

As shown in Figure 18, the GRIDGETRESULT parameter is used to get the output and log of the job that was run on the Grid by copying the files back from the <<GRIDWORK>> location and onto the user's desktop.

BEYOND THE BASICS

SASGSUB and Credentials

SASGSUB always needs to authenticate users to the SAS® Metadata Server in order to determine whether they are allowed to use the SAS Grid. If the user does not have the ReadMetadata permissions on the Logical Grid Server defined in Metadata, that user cannot use the Grid from SASGSUB or from any other SAS client, for that matter.

However, the authentication process has nothing to do with the actual user identity that is running the SAS program: even though the -METAUSER parameter drives the metadata authentication, the job started by LSF on the Grid runs under the credentials of the user that is currently logged in. To illustrate this, one could very well be logged in to a desktop as the user GRIDUSER01, and use -METAUSER sasadm@saspw when submitting using SASGSUB. For the authentication to metadata the password for sasadm@saspw would have to be supplied. When looking at the running job, the user ID GRIDUSER01 would be the one appearing next to the job. This is an exaggerated case, for illustration purposes only, and is obviously not recommended at all, unless you consider troubleshooting to be an Olympic discipline, and want to prepare for London 2012.

In effect, if SASGSUB was deployed on many different desktops, it is likely that either the METAUSER parameter would have a different value on each desktop, or that the SAS Grid Administrator created a shared internal account (for example gsub_user@saspw) and made everyone use those credentials.

That being said, even the most old-fashioned among us will quickly become tired of constantly having to enter their password every time a job is submitted. This can quickly become frustrating, even more so when you later realize that there are alternatives:

1. **Passing the password as a parameter**
Just like the METAUSER parameter, you can specify a value for the METAPASS parameter. That can be done at invocation time, or changed in the sasgsub.cfg file, as long as the file is specific to you and protected by some OS-level security. Whether in the configuration file or at invocation time, it is always better to provide the password in an encrypted format. Your improved submission command could therefore look like the one depicted in Figure 19.
2. **Pointing to an xml profile file**
Another way of providing password (and general metadata connection information) is to provide SASGSUB with the location of a .xml file describing the connection parameters to the metadata. An example of such a file can be found in the Object Spawner folder (<<sas_config>>/Lev1/ObjectSpawner2/metadataConfig.xml). You can reference that .xml file by using the METAPROFILE parameter, as seen in Figure 20

```
/sas93/config/compute/Lev1/Applications/SASGridManagerClientUtility/9.3/sasgsub -
gridsubmitpgm /sas93/resources/paper/sas_programs/sleep5.sas -GRIDJOBNAME Sleep5 -
GRIDJOBPTS 'queue=normal' -METAPASS {SAS002}DA9A0A5C20629B7F34D2C88A165E5530
```

Figure 19: Passing the Encoded Password as a Parameter

```
[sasdemo@sasserver03 sasinst]$
/sas93/config/compute/Lev1/Applications/SASGridManagerClientUtility/9.3/sasgsub -
gridsubmitpgm /sas93/resources/paper/sas_programs/sleep5.sas -GRIDJOBNAME Sleep5 -
GRIDJOBPTS 'queue=normal' -METAPROFILE ~/conf.xml
```

RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

```
SAS Grid Submit Utility Version 9.30 (build date: May 11 2011)
Copyright (C) 2009-2011, SAS Institute Inc., Cary, NC, USA. All Rights Reserved
Job <3931> is submitted to queue <normal>.
Job ID:          3931
Job directory:   "/sas93/sharedwork/gridshared/sasdemo/SASGSUB-2012-01-
31_10.55.50.950_sleep5"
Job log file:    "/sas93/sharedwork/gridshared/sasdemo/SASGSUB-2012-01-
31_10.55.50.950_sleep5/sleep5.log"
[sasdemo@sasserver03 sasinst]$
```

Figure 20: Using a Metadata Profile .xml File

Executing Commands

Starting with SAS® 9.3, SASGSUB gives the users the ability to do more than executing just SAS programs on the Grid: It allows them to execute scripts or commands as well. However, the SAS Grid Manager license requires that these scripts and commands be in support of the SAS environment and do not invoke other products or applications.

This can be useful for users who want to perform some SAS related activities on the server(s) for which a SAS program is not the best fit. For example, it can be used to delete old SASGSUB folders (Figure 21), to compress a file resulting from a SAS program (Figure 22), or even to run the cleanwork utility (Figure 23). Note that the OS account used to execute the cleanwork utility is not sasdemo but sasinst in the example. This illustrates the fact that scripts or programs can be run instead of commands, as long as they are available on the Grid Nodes.

```
[sasdemo@sasserver03
/sas93/config/compute/Levl/Applications/SASGridManagerClientUtility/9.3/sasgsub -
metapass {SASENC}46EE8513422F11D859B25CC6 -GRIDRUNCMD "rm -rf
/sas93/sharedwork/gridshared/sasdemo/SASGSUB-2012-01-27*"
```

Figure 21: GridWork Cleanup

```
[sasdemo@sasserver03
/sas93/config/compute/Levl/Applications/SASGridManagerClientUtility/9.3/sasgsub -
metapass {SASENC}46EE8513422F11D859B25CC6 -GRIDRUNCMD "zip
/sas93/sharedwork/gridshared/sasdemo/output.txt"
```

Figure 22: Compressing an Output File

```
[sasinst@sasserver03 ~]$
/sas93/config/compute/Levl/Applications/SASGridManagerClientUtility/9.3/sasgsub -
metapass {SASENC}46EE8513422F11D859B25CC6 -GRIDRUNCMD
"/sas93/software/compute/SASFoundation/9.3/utilities/bin/cleanwork /tmp"
```

Figure 23: Executing the cleanwork utility

File Staging

File staging is beyond the scope of this paper, but it is important for people to be aware of its existence.

Part of the actions of a default SASGSUB submission includes copying the local script or SAS program into the <<GRIDWORK>> location. That location has to be available on all the Grid nodes, so that regardless of where the job starts, it can access the required files. By default, to perform this copy, SASGSUB executes an OS-level copy. To put it another way, it would be as if SASGSUB would issue a command like "copy C:\myprog.sas \\file_share\GRIDWORK\sasdemo\SASGSUB-2012-01-30_11.50.11.619_MyProg\".

For this to work, your windows desktop needs to be able to access the GRIDWORK share. In cases where the Grid consists of UNIX or Linux servers, it is possible (even likely) that the Clustered File System of choice is dedicated to those servers and therefore is not readily accessible from the Windows desktops.

RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

In those situations, some customers choose to surface the <<GRIDWORK>> location through a SAMBA share so that the desktops can have access to it.

You can also configure an alternative way of getting those files onto the server: This is referred to as file staging.

The GRIDSTAGECMD and GRIDSTAGEFILEHOST parameters allow you to instruct your Grid on how to obtain the required files when running jobs. These methods include rcp, lsrcp, scp, pscp, or smbclient. (For more information about file staging, consult the Recommended Reading section of this paper).

USING RTM AND SASGSUB TO TUNE YOUR ENVIRONMENT

In the two previous sections, this paper has explored the monitoring, management, and configuration features of RTM, as well as how SASGSUB can be used to submit jobs to the Grid. Although it is obvious that these tools are very different from one another, they do complement each other. Used together with a little scripting knowledge, they can be very powerful instruments in the SAS Grid Administrator's hands.

This section covers techniques used to validate and tune SAS Grid Manager environments. Each Grid is different, and each scenario has to be designed appropriately. These examples are meant to provide you with both inspiration and sample code to start from. This code was written for a very specific environment and scenario. Its modification and use outside can lead to unpredictable results.

In the examples of this section, SASGSUB was not only installed as part of the server installation, but it was installed on the clustered file system that is shared by all machines. Because of this, SASGSUB can be called from any machine in the cluster.

VERIFYING LSF PARAMETERS CHANGES

People not familiar with SAS Grid Manager suppose that jobs are started by LSF as quickly as possible. That is not the case, and in fact, that would not work out very well. Instead, LSF has a set of built-in sleep times to give itself enough time to make good decisions about the job control.

There are a number of parameters in LSF that can be modified to increase or decrease the “rate of fire” of the Grid. It's important to understand them, and SASGSUB can help you in seeing what is happening when you modify those parameters.

```
[sasdemo@sasserver03 sasinst]$ bparams
Default Queues: normal
Job Dispatch Interval: 20 seconds
Job Checking Interval: 15 seconds
Job Accepting Interval: 20 seconds
[sasdemo@sasserver03 sasinst]$ bhosts
```

HOST_NAME	STATUS	JL/U	MAX	NJOBS	RUN	SSUSP	USUSP	RSV
sasserver01	ok	-	5	0	0	0	0	0
sasserver02	ok	-	5	0	0	0	0	0
sasserver03	ok	-	5	0	0	0	0	0
sasserver04	ok	-	5	0	0	0	0	0
sasserver05	ok	-	5	0	0	0	0	0

```
[sasdemo@sasserver03 sasinst]$
```

Figure 24: Checking Parameters

In Figure 24, you can see the dispatch intervals and the status of the batch hosts. As this is a “vanilla” installation the job dispatching is done every 20 seconds. This means that LSF wakes up, performs some actions on the jobs that have accumulated, goes to sleep for 20 seconds, and starts all over again. The hosts have been configured to have 5 job slots each, and currently no job is running.

```
## this shell script is used to quickly submit multiple jobs to the grid
GSUB='/sas93/config/compute/Lev1/Applications/SASGridManagerClientUtility/9.3/sasgs
ub -METAPASS {SASENC}46EE8513422F11D859B25CC6'
```

RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

```

PROG_FOLDER=/sas93/resources/paper/sas_programs/
OPT="-GRIDJOBPTS 'queue=batch'"

for i in `seq 1 40`;
do
    $GSUB -gridsubmitpgm $PROG_FOLDER/sleep5.sas -GRIDJOBNAME Job_$i $OPT &
done

while [[ $c -le 0 ]]; do
    clear ;
    echo We have submitted 10 jobs simultaneously;
    date
    echo ----- bqueues output: -----
    bqueues batch
    echo ----- bhosts output: -----
    bhosts
    echo
    echo Type Ctrl-C to cancel
    sleep 2;
done

```

Figure 25: submitting jobs in rapid succession

The shell script in Figure 25 submits the sleep5.sas program 40 times to the grid. These are sent to a queue called "batch" and the jobs are named Job_1 to Job_40. The second part of the script continuously displays the output of the bqueues and bhosts commands in the prompt window.

Job ID**	Job Name	Status	State Changes	User ID	CPU Usage	CPU Effic	Execution Host	Running Queue	Submit Time	Start Time	End Time	Pend	Run
4325	Sleep5_Job_1	RUNNING	2	sasdemo	-	-	sasserver04	batch	01-31 15:00:57	01-31 15:01:01	-	0.1m	1.7m
4326	Sleep5_Job_2	RUNNING	2	sasdemo	-	-	sasserver03	batch	01-31 15:00:57	01-31 15:01:01	-	0.1m	1.7m
4327	Sleep5_Job_3	RUNNING	2	sasdemo	-	-	sasserver02	batch	01-31 15:00:57	01-31 15:01:01	-	0.1m	1.7m
4328	Sleep5_Job_4	RUNNING	2	sasdemo	-	-	sasserver01	batch	01-31 15:00:57	01-31 15:01:01	-	0.1m	1.7m
4329	Sleep5_Job_5	RUNNING	2	sasdemo	-	-	sasserver05	batch	01-31 15:00:58	01-31 15:01:01	-	0.1m	1.7m
4330	Sleep5_Job_6	RUNNING	2	sasdemo	-	-	sasserver03	batch	01-31 15:00:58	01-31 15:01:21	-	0.4m	1.4m
4331	Sleep5_Job_7	RUNNING	2	sasdemo	-	-	sasserver04	batch	01-31 15:00:58	01-31 15:01:21	-	0.4m	1.4m
4332	Sleep5_Job_8	RUNNING	2	sasdemo	-	-	sasserver02	batch	01-31 15:00:58	01-31 15:01:21	-	0.4m	1.4m
4333	Sleep5_Job_9	RUNNING	2	sasdemo	-	-	sasserver01	batch	01-31 15:00:59	01-31 15:01:21	-	0.4m	1.4m
4334	Sleep5_Job_10	RUNNING	2	sasdemo	-	-	sasserver05	batch	01-31 15:00:59	01-31 15:01:21	-	0.4m	1.4m
4335	Sleep5_Job_11	RUNNING	2	sasdemo	-	-	sasserver03	batch	01-31 15:00:59	01-31 15:01:42	-	0.7m	1.1m
4336	Sleep5_Job_12	RUNNING	2	sasdemo	-	-	sasserver04	batch	01-31 15:00:59	01-31 15:01:42	-	0.7m	1.1m
4337	Sleep5_Job_13	RUNNING	2	sasdemo	-	-	sasserver02	batch	01-31 15:00:59	01-31 15:01:42	-	0.7m	1.1m
4338	Sleep5_Job_14	RUNNING	2	sasdemo	-	-	sasserver05	batch	01-31 15:00:59	01-31 15:01:42	-	0.7m	1.1m
4339	Sleep5_Job_15	RUNNING	2	sasdemo	-	-	sasserver01	batch	01-31 15:00:59	01-31 15:01:42	-	0.7m	1.1m
4340	Sleep5_Job_16	RUNNING	2	sasdemo	-	-	sasserver02	batch	01-31 15:00:59	01-31 15:02:02	-	1.1m	0.7m
4341	Sleep5_Job_17	RUNNING	2	sasdemo	-	-	sasserver01	batch	01-31 15:00:59	01-31 15:02:02	-	1.1m	0.7m
4342	Sleep5_Job_18	RUNNING	2	sasdemo	-	-	sasserver03	batch	01-31 15:01:00	01-31 15:02:02	-	1m	0.7m
4343	Sleep5_Job_19	RUNNING	2	sasdemo	-	-	sasserver05	batch	01-31 15:01:00	01-31 15:02:02	-	1m	0.7m
4344	Sleep5_Job_20	RUNNING	2	sasdemo	-	-	sasserver04	batch	01-31 15:01:00	01-31 15:02:02	-	1m	0.7m
4345	Sleep5_Job_21	RUNNING	2	sasdemo	-	-	sasserver02	batch	01-31 15:01:00	01-31 15:02:22	-	1.4m	0.4m
4346	Sleep5_Job_22	RUNNING	2	sasdemo	-	-	sasserver03	batch	01-31 15:01:00	01-31 15:02:22	-	1.4m	0.4m
4347	Sleep5_Job_23	RUNNING	2	sasdemo	-	-	sasserver05	batch	01-31 15:01:00	01-31 15:02:22	-	1.4m	0.4m
4348	Sleep5_Job_24	RUNNING	2	sasdemo	-	-	sasserver04	batch	01-31 15:01:00	01-31 15:02:22	-	1.4m	0.4m
4349	Sleep5_Job_25	RUNNING	2	sasdemo	-	-	sasserver01	batch	01-31 15:01:00	01-31 15:02:22	-	1.4m	0.4m
4350	Sleep5_Job_26	PEND	1	sasdemo	-	-	batch	batch	01-31 15:01:00	-	-	2m	0m
4351	Sleep5_Job_27	PEND	1	sasdemo	-	-	batch	batch	01-31 15:01:01	-	-	2m	0m

Figure 26: Looking at Jobs in RTM

After waiting for RTM to poll LSF and update its data, you can see (in Figure 26) that the jobs are dispatched in "bunches" of 5 (one on each machine), every 20 seconds (first at 15:01:01, then 15:01:21, then 15:01:42, and so on). This continues until all of the job slots (5 per machine, total of 25) are taken. At that point, the remaining 15 jobs stay pending in the queue, until some slots free up. In this situation, it takes 5 times 20 seconds = 100 seconds to get all of the jobs started.

This illustrates very well the inherent overhead of the Grid. This is definitely not a bug in the software: These sleep times are there to allow time for the dispatched jobs to start executing so that LSF has an accurate view of resource

RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

usage. This ensures that things do not happen too quickly so that LSF can keep tight control over what is going on.

This overhead can nevertheless be an issue, if either it is critical that jobs start very quickly or if all Grid jobs are so short (less than a few minutes) that this overhead adds up to vast amounts. In this case, you can reduce the MBD and SBD sleep times to speed things up. For example, by bringing it down to 5 seconds, you should get a different behavior: However, most grid environments have jobs with multiple run-time characteristics. In this case, the default settings provide the best dispatching and distribution of jobs across the grid.

```
[sasdemo@sasserver03 sasinst]$ bparams
Default Queues: normal
Job Dispatch Interval: 5 seconds
Job Checking Interval: 5 seconds
Job Accepting Interval: 5 seconds
[sasdemo@sasserver03 sasinst]$
```

Figure 27: Confirming the Changes to the Sleep Times

JobID**	Job Name	Status	State Changes	User ID	CPU Usage	CPU Effic	Execution Host	Running Queue	Submit Time	Start Time	End Time	Pend	Run
4469	Sleep5_job_1	DONE	2	sasdemo	0.2s	0.07%	sasserver04	batch	01-31 15:24:02	01-31 15:24:04	01-31 15:29:05	0m	5m
4470	Sleep5_job_2	DONE	2	sasdemo	0.18s	0.06%	sasserver01	batch	01-31 15:24:02	01-31 15:24:04	01-31 15:29:04	0m	5m
4471	Sleep5_job_3	DONE	2	sasdemo	0.21s	0.07%	sasserver02	batch	01-31 15:24:02	01-31 15:24:04	01-31 15:29:05	0m	5m
4472	Sleep5_job_4	DONE	2	sasdemo	0.19s	0.06%	sasserver03	batch	01-31 15:24:02	01-31 15:24:04	01-31 15:29:05	0m	5m
4473	Sleep5_job_5	DONE	2	sasdemo	0.18s	0.06%	sasserver04	batch	01-31 15:24:03	01-31 15:24:09	01-31 15:29:10	0.1m	5m
4474	Sleep5_job_6	DONE	2	sasdemo	0.18s	0.06%	sasserver01	batch	01-31 15:24:03	01-31 15:24:09	01-31 15:29:09	0.1m	5m
4475	Sleep5_job_7	DONE	2	sasdemo	0.18s	0.06%	sasserver02	batch	01-31 15:24:03	01-31 15:24:09	01-31 15:29:10	0.1m	5m
4476	Sleep5_job_8	DONE	2	sasdemo	0.19s	0.06%	sasserver03	batch	01-31 15:24:03	01-31 15:24:09	01-31 15:29:10	0.1m	5m
4477	Sleep5_job_9	DONE	2	sasdemo	0.22s	0.07%	sasserver05	batch	01-31 15:24:03	01-31 15:24:09	01-31 15:29:10	0.1m	5m
4478	Sleep5_job_10	DONE	2	sasdemo	0.2s	0.07%	sasserver04	batch	01-31 15:24:03	01-31 15:24:14	01-31 15:29:15	0.2m	5m
4479	Sleep5_job_11	DONE	2	sasdemo	0.18s	0.06%	sasserver01	batch	01-31 15:24:03	01-31 15:24:14	01-31 15:29:15	0.2m	5m
4480	Sleep5_job_12	DONE	2	sasdemo	0.18s	0.06%	sasserver02	batch	01-31 15:24:03	01-31 15:24:14	01-31 15:29:15	0.2m	5m
4481	Sleep5_job_13	DONE	2	sasdemo	0.2s	0.07%	sasserver03	batch	01-31 15:24:04	01-31 15:24:14	01-31 15:29:15	0.2m	5m
4482	Sleep5_job_14	DONE	2	sasdemo	0.21s	0.07%	sasserver05	batch	01-31 15:24:04	01-31 15:24:14	01-31 15:29:15	0.2m	5m
4483	Sleep5_job_15	DONE	2	sasdemo	0.19s	0.06%	sasserver04	batch	01-31 15:24:04	01-31 15:24:19	01-31 15:29:20	0.3m	5m
4484	Sleep5_job_16	DONE	2	sasdemo	0.19s	0.06%	sasserver01	batch	01-31 15:24:04	01-31 15:24:19	01-31 15:29:20	0.3m	5m
4485	Sleep5_job_17	DONE	2	sasdemo	0.2s	0.07%	sasserver02	batch	01-31 15:24:04	01-31 15:24:19	01-31 15:29:20	0.3m	5m
4486	Sleep5_job_18	DONE	2	sasdemo	0.2s	0.07%	sasserver03	batch	01-31 15:24:04	01-31 15:24:19	01-31 15:29:20	0.3m	5m
4487	Sleep5_job_19	DONE	2	sasdemo	0.22s	0.07%	sasserver05	batch	01-31 15:24:04	01-31 15:24:19	01-31 15:29:20	0.3m	5m
4488	Sleep5_job_20	DONE	2	sasdemo	0.2s	0.07%	sasserver04	batch	01-31 15:24:04	01-31 15:24:24	01-31 15:29:25	0.3m	5m
4489	Sleep5_job_21	DONE	2	sasdemo	0.17s	0.06%	sasserver01	batch	01-31 15:24:05	01-31 15:24:24	01-31 15:29:25	0.3m	5m
4490	Sleep5_job_22	DONE	2	sasdemo	0.18s	0.06%	sasserver02	batch	01-31 15:24:05	01-31 15:24:24	01-31 15:29:25	0.3m	5m
4491	Sleep5_job_23	DONE	2	sasdemo	0.19s	0.06%	sasserver03	batch	01-31 15:24:05	01-31 15:24:24	01-31 15:29:25	0.3m	5m
4492	Sleep5_job_24	DONE	2	sasdemo	0.2s	0.07%	sasserver05	batch	01-31 15:24:05	01-31 15:24:24	01-31 15:29:25	0.3m	5m
4493	Sleep5_job_25	DONE	2	sasdemo	0.2s	0.07%	sasserver05	batch	01-31 15:24:05	01-31 15:24:29	01-31 15:29:30	0.4m	5m
4494	Sleep5_job_26	DONE	2	sasdemo	0.19s	0.06%	sasserver01	batch	01-31 15:24:05	01-31 15:29:07	01-31 15:34:07	5m	5m
4495	Sleep5_job_27	DONE	2	sasdemo	0.19s	0.06%	sasserver04	batch	01-31 15:24:05	01-31 15:29:07	01-31 15:34:07	5m	5m
4496	Sleep5_job_28	DONE	2	sasdemo	0.18s	0.06%	sasserver03	batch	01-31 15:24:05	01-31 15:29:07	01-31 15:34:08	5m	5m
4497	Sleep5_job_29	DONE	2	sasdemo	0.15s	0.05%	sasserver02	batch	01-31 15:24:06	01-31 15:29:07	01-31 15:34:08	5m	5m
4498	Sleep5_job_30	DONE	2	sasdemo	0.17s	0.06%	sasserver01	batch	01-31 15:24:06	01-31 15:29:12	01-31 15:34:12	5.1m	5m
4499	Sleep5_job_31	DONE	2	sasdemo	0.2s	0.07%	sasserver04	batch	01-31 15:24:06	01-31 15:29:12	01-31 15:34:12	5.1m	5m
4500	Sleep5_job_32	DONE	2	sasdemo	0.2s	0.07%	sasserver03	batch	01-31 15:24:06	01-31 15:29:12	01-31 15:34:12	5.1m	5m
4501	Sleep5_job_33	DONE	2	sasdemo	0.22s	0.07%	sasserver05	batch	01-31 15:24:06	01-31 15:29:12	01-31 15:34:12	5.1m	5m
4502	Sleep5_job_34	DONE	2	sasdemo	0.19s	0.06%	sasserver02	batch	01-31 15:24:06	01-31 15:29:12	01-31 15:34:12	5.1m	5m
4503	Sleep5_job_35	DONE	2	sasdemo	0.19s	0.06%	sasserver01	batch	01-31 15:24:06	01-31 15:29:17	01-31 15:34:17	5.2m	5m

Figure 28: Jobs Are Now Dispatched Faster

Figure 28 clearly shows that now that the sleep times have been lowered down to 5, jobs are dispatched much more quickly, and it only takes 25 seconds to dispatch all the jobs that there are slots for. It is then required to wait until the first one (jobID 4469) finishes at 15:29:05 for more jobs (4494-4497) to start.

LOAD TESTING

Until now, most of the programs submitted were variations on sleep commands. While this is useful to test parallelism and concurrency of jobs, it does not give a good picture of how the hardware, controlled by LSF, reacts to real SAS workload being performed. It is however not very difficult to build some simple SAS programs that would simulate the load users could put on the environment. The ones provided below are included as illustrations:

RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

```
%macro stepsort(reps,recs);  
%do i=1 %to &reps;  
  data work.a1;  
    do a=1 to &recs;  
      j=ranuni(&i);  
      output;  
    end;  
  run;  
  Proc sort data=a1 out=work.a2;  
    by j;  
  run;  
  
  data work.b1;  
    do b=1 to &recs;  
      j=ranuni(time());  
      output;  
    end;  
  run;  
  Proc sort data=b1 out=work.b2;  
    by j;  
  run;  
  
  data c;  
    merge a2 b2 ;  
    by j;  
  run;  
  
%end;  
%mend;  
  
%stepsort(10,100000000);
```

Figure 29: Content of the StepSort.sas Program

The SAS program in Figure 29 is meant to repetitively perform DATA steps and procedures to simulate the load that batch-type programs could inflict on the machines. By submitting this program 50 times through SASGSUB, you can see how the Grid reacts in Figure 30. The load across the cluster stays well below the cluster capacity, thanks to host thresholds that keep machines from being overloaded. As can be seen on the left hand side, this is done by suspending (Sys Suspended) some running jobs when the load on the host comes close to the limit. At the bottom of the graphs, you can see in the weekly view the type of recurrent processing that seems to happen nightly on a scheduled basis.

RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

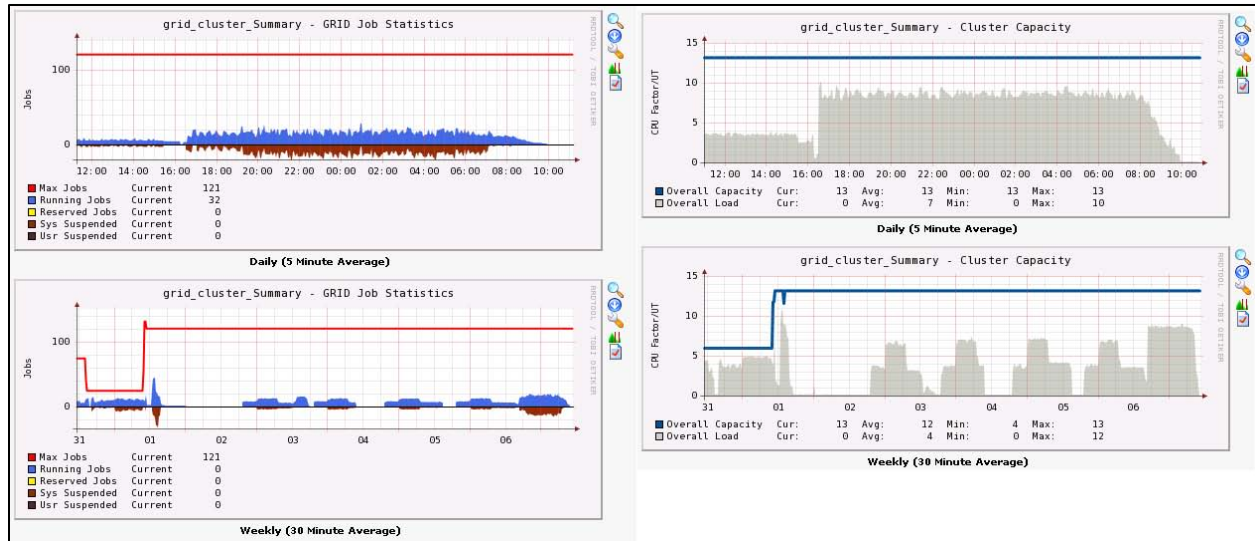


Figure 30: Job Statistics and Cluster Capacity with StepSort Program

```
/* this sas program aims at simulating the activities of an EG user */
%macro EG_sim(type,dur);
%if "&type"="cpu" %then %do;
  data _null_;
  stt= datetime();
  i=1;
  do until (i = 0);
    i=i+1;
    j=i**2;
    now= datetime();
    elap=(now-stt) ;
    if elap > &dur then do;
      put "cpu intensive for " elap "seconds";
      i=0;
    end;
  end;
end;
run;
%end;
%if "&type"="sleep" %then %do;
  data _null_;
  call sleep(&dur*1000);
  put "slept for &dur seconds";
run;
%end;
%mend EG_sim;

options mprint;

**sample execution of the macro **;
%EG_sim(cpu,10);
%EG_sim(sleep,10);

%let daylength=8; * hours *;

data day;
length type $20.;
l=0;
do until (l > &daylength*3600);
  i+1;
```

RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

```

    stt=l;
    if mod(i,2)=1 then do;  *odd;
        type="cpu ";
        l=l+ranuni(time())*2000;
        put i= l=;
    end;
    else do;
        type="active";
        l=l+ranuni(time())*100;
    end;
    end=l;
    dur=end-stt;
    output;
end;
run;

proc print;
run;

data _null_;
set day;
cmd=compress('%EG_sim('||type||","||dur||");");
call execute(cmd);
run;

```

Figure 31: Content of the eg_sim.sas Program

In the previous example, StepSort had a true batch profile: it runs continuously, from top to bottom, until the whole program is done. That is not the way all of the users work, and especially not when they are in the early stages of developing a new Enterprise Guide Project or some ETL processes. In those situations (interactive use) the user typically runs small steps many times, with some thinking and analyzing time, until they move on to the next step in their process. This means that in spite of their SAS session running for many hours, it only performs “real” work for minutes.

To simulate that behavior, the eg_sim.sas program alternates between cpu-intensive steps and idle steps for about eight hours.

In order to see how the Grid reacts to a large number of interactive users, you can use a shell script like the one in Figure 32.

```

##  this shell script is used to quickly submit multiple jobs to the grid
GSUB='/sas93/config/compute/Levl/Applications/SASGridManagerClientUtility/9.3/sasgs
ub -METAPASS {SASENC}46EE8513422F11D859B25CC6'
PROG_FOLDER=/sas93/resources/paper/sas_programs/

for i in `seq 1 20`;
do
    x=`printf "%02d" $i`
    OPT="-GRIDJOBPTS 'queue=interactive_q'"
    sudo -u griduser$x -i $GSUB -gridsubmitpgm $PROG_FOLDER/eg_sim.sas -
GRIDJOBNAME EG_A_sim_$x $OPT
    sudo -u griduser$x -i $GSUB -gridsubmitpgm $PROG_FOLDER/eg_sim.sas -
GRIDJOBNAME EG_B_sim_$x $OPT
    sudo -u griduser$x -i $GSUB -gridsubmitpgm $PROG_FOLDER/eg_sim.sas -
GRIDJOBNAME EG_C_sim_$x $OPT
    sleep 3
done

```

Figure 32: Submitting Interactive Program

This time, jobs are submitted as different users (using the UNIX command sudo). Each user (griduse01 to griduser20) submits 3 jobs (A, B, and C) to the queue named Interactive_q. This type of processing is obviously

RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

putting less pressure on the machines, as most of the time is spent idling. That can also be seen by looking at the graph from a single job in Figure 33.

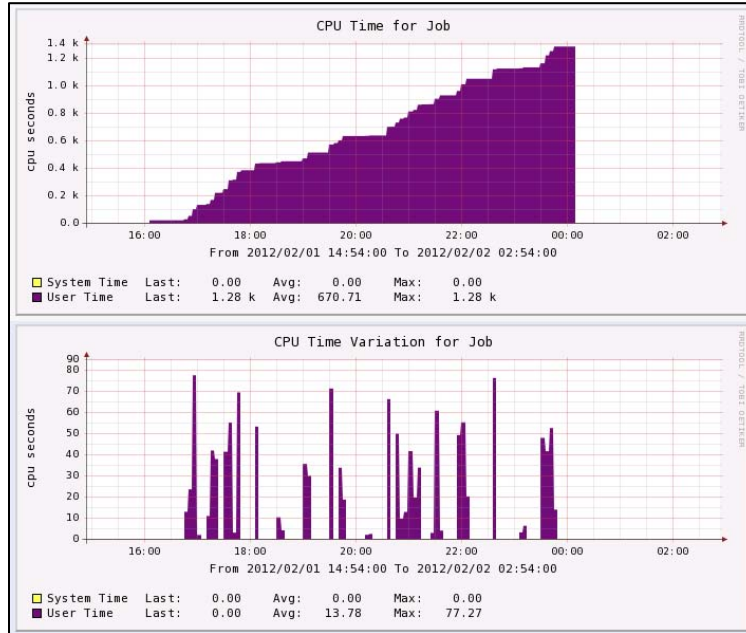


Figure 33: RTM Graph of Single Run of eg_sim.sas

VERIFYING AN ADVANCED LSF CONFIGURATION

LSF is a very versatile piece of software that can be configured to work in many different ways. Take for example the configuration described in Figure 34. The SAS Grid Administrator probably had good reasons to create it in this way, but it is easy to overlook something and end up with unexpected results. It is better if you find that out before the users start using the environment.

lsf.cluster.grid cluster

<<..>>

Begin Host

HOSTNAME	mem	server	swp	model	rlm	type	RESOURCES
sasserver01	()	1	()	!	()	!	(mg SASApp)
sasserver02	()	1	()	!	()	!	(mg SASApp)
sasserver03	()	1	()	!	()	!	(mg SASApp)
sasserver04	()	1	()	!	()	!	(mg SASApp)
sasserver05	()	1	()	!	()	!	(mg SASApp)
sasclient01	()	0	()	!	()	!	()
sgcwin02013	()	1	()	!	()	!	(mg SASApp)
sgcwin02014	()	1	()	!	()	!	(mg SASApp)
sgcwin02015	()	1	()	!	()	!	(mg SASApp)
sgcwin02016	()	1	()	!	()	!	(mg SASApp)
sgcwin02017	()	1	()	!	()	!	(mg SASApp)
sgcwin02018	()	1	()	!	()	!	(mg SASApp)

End Host

<<..>>

lsb.hosts

<<..>>

Begin Host

HOST_NAME	ut	r15s	MXJ	tmp	DISPATCH_WINDOW
default	0.8/0.9	()	12	()	()
sasserver01	0.4/0.5	()	1	()	()

RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

```

End Host

Begin HostGroup
GROUP_NAME      GROUP_ADMIN      CONDENSE      GROUP_MEMBER
GridExternal    ()      N      (sasserver02 sasserver05)
GridOnly        ()      N      (sgcwin02013 sgcwin02014 sgcwin02015 sgcwin02016
sgcwin02017 sgcwin02018 sasserver03 sasserver04)
End HostGroup
<<..>>

lsb.queues
<<..>>
Begin Queue
  QUEUE_NAME=interactive_q
  #if time(5:17:00-1:8:00 18:00-8:00)
    NICE=10
  #else
    NICE=20
  #endif
  #if time(1:8:00-1:18:00 2:8:00-2:18:00 3:8:00-3:18:00 4:8:00-4:18:00 5:8:00-
5:18:00)
    PREEMPTION=PREEMPTIVE[batch_q]
  #endif
  #if time(5:17:00-1:8:00 18:00-8:00)
    PRIORITY=30
  #else
    PRIORITY=60
  #endif
  UJOB_LIMIT=2
  HOSTS=GridExternal GridOnly
  RERUNNABLE=NO
End Queue

Begin Queue
  QUEUE_NAME=batch_q
  #if time(5:17:00-1:8:00 18:00-8:00)
    NICE=20
  #else
    NICE=10
  #endif
  #if time(5:17:00-1:8:00 18:00-8:00)
    PREEMPTION=PREEMPTIVE[interactive_q]
  #endif
  #if time(5:17:00-1:8:00 18:00-8:00)
    PRIORITY=60
  #else
    PRIORITY=30
  #endif
  JOB_ACCEPT_INTERVAL=3
  HOSTS=GridExternal GridOnly
  PJOB_LIMIT=0.5
  RERUNNABLE=YES
End Queue

```

Figure 34: An Example of Advanced LSF Configuration

In order to test this configuration's behavior, a shell script is created, to submit both the StepSort and eg_sim programs multiple times:

```

## this shell script is used to quickly submit multiple jobs to the grid
GSUB='/sas93/config/compute/Levl/Applications/SASGridManagerClientUtility/9.3/sasgs
ub -METAPASS {SASENC}46EE8513422F11D859B25CC6'

```

RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

```

PROG_FOLDER=/sas93/resources/paper/sas_programs/

badadmin qinact batch_q interactive_q

for i in `seq 1 50`;
do
    OPT="-GRIDJOBPTS 'queue=batch_q'"
    sudo -u sasdemo -i $GSUB -gridsubmitpgm $PROG_FOLDER/stepsort.sas -GRIDJOBNAME
stepsort_$i $OPT
done

for i in `seq 1 20`;
do
    x=`printf "%02d" $i`
    OPT="-GRIDJOBPTS 'queue=interactive_q'"
    sudo -u griduser$x -i $GSUB -gridsubmitpgm $PROG_FOLDER/eg_sim.sas -
GRIDJOBNAME EG_A_sim_$x $OPT
    sudo -u griduser$x -i $GSUB -gridsubmitpgm $PROG_FOLDER/eg_sim.sas -
GRIDJOBNAME EG_B_sim_$x $OPT
    sudo -u griduser$x -i $GSUB -gridsubmitpgm $PROG_FOLDER/eg_sim.sas -
GRIDJOBNAME EG_C_sim_$x $OPT
done

badadmin qact batch_q interactive_q

```

Figure 35: Submitting Multiple Jobs Using a Script

This script first inactivates both queues (interactive_q and batch_q) so that they can receive jobs but not dispatch them. It then submits the StepSort program 50 times to the batch_q queue as the sasdemo user. It then proceeds to submit 60 jobs to the interactive_q queue. These 60 jobs are the same job (eg_sim), submitted by 20 different users (griduser01 to griduser20) 3 times each. Finally, once all of these jobs have been submitted, both queues are activated again, so that they start dispatching these 110 jobs that are pending in the queues.

The following figures show how some of the LSF settings can be confirmed by looking in RTM at how the jobs are running.












<< Previous		Showing Rows 1 to 11 of 11 [1]											Next >	
Actions	Host Name	Cluster	Type	Model	Load/Batch	CPU Fact	CPU Pct	RunQ Im	Max Slots	Num Slots	Run Slots	SSUSP Slots	USUSP Slots	Reserve Slots
	log sgcwin02014.race.sas.com	grid_cluster	X86_64	Intel_EM64T	Ok:Closed-Busy	60	59.96%	0.86	12	5	5	0	0	0
	log sgcwin02015.race.sas.com	grid_cluster	X86_64	Intel_EM64T	Ok:Closed-Busy	60	71.53%	1.34	12	5	5	0	0	0
	log sgcwin02016.race.sas.com	grid_cluster	X86_64	Intel_EM64T	Ok:Closed-Busy	60	58.94%	0.60	12	5	5	0	0	0
	log sgcwin02017.race.sas.com	grid_cluster	X86_64	Intel_EM64T	Ok:Closed-Busy	60	59.67%	0.76	12	5	5	0	0	0
	log sasserver02	grid_cluster	X86_64	Intel_EM64T	Ok:Ok	60	58.30%	0.60	12	5	5	0	0	0
	log sgcwin02018.race.sas.com	grid_cluster	X86_64	Intel_EM64T	Ok:Ok	60	66.89%	1.03	12	5	5	0	0	0
	log sasserver03	grid_cluster	X86_64	Intel_EM64T	Ok:Closed-Busy	60	60.40%	1.03	12	5	5	0	0	0
	log sasserver04	grid_cluster	X86_64	Intel_EM64T	Ok:Ok	60	57.81%	0.75	12	5	5	0	0	0
	log sasserver05	grid_cluster	X86_64	Intel_EM64T	Ok:Closed-Busy	60	69.87%	1.61	12	5	5	0	0	0
	log sgcwin02013.race.sas.com	grid_cluster	X86_64	Intel_EM64T	Ok:Closed-Busy	60	70.51%	1.18	12	5	5	0	0	0
	log sasserver01	grid_cluster	X86_64	Intel_EM64T	Ok:Closed-Admin	60	4.07%	0.02	1	0	0	0	0	0
<< Previous		Showing Rows 1 to 11 of 11 [1]											Next >	

Figure 36: Job Distribution per Host

In Figure 36, you can see that all hosts except SASSERVER01 get to run 5 jobs. SASSERVER01 does not run any jobs because it has been closed by an administrator and it is not part of either host group that our queues use. This confirms this machine is effectively prevented from running any Grid Job.

The fact that there are only 5 jobs running on each host can be explained as follows:

- In batch_q, the PJOB_LIMIT is 0.5
 - which means that batch_q can send only one job at a time to one host (each host here has only 2 cores)

RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

- 1 out of the 5 jobs is a StepSort job
- The other 4 are therefore eg_sim jobs running on each host, coming from interactive_q
- In interactive_q, the limit on job slots is UJOB_LIMIT=2
 - That means each user can get only 2 jobs running simultaneously, regardless of hosts
 - There are 20 users, each getting 2 slots, so they occupy a total of 40 slots
 - Spread out over 10 hosts, that would account for the 4 other jobs on each host

<< Previous										Showing Rows 1 to 100 of 113 [1,2]	
JobID**	Job Name	Status	State Changes	Nice Level	User ID	CPU Usage	CPU Effic	Execution Host	Running Queue		
13839	EG_A_sim_01	RUNNING	2	20	griduser01	33s	0.98%	sgcwin02017.race.sas.com	interactive		
13840	EG_B_sim_01	RUNNING	2	20	griduser01	51s	1.52%	sgcwin02016.race.sas.com	interactive		
13841	EG_C_sim_01	PEND	1	20	griduser01	-	-		interactive		

Job Detail	
General Information	
JobId:	13841
Status:	PEND
Jobname:	EG_C_sim_01
Project:	default
License Project:	NULL
Queue:	interactive_q
Cluster Name:	grid_cluster
User:	griduser01
User Group:	
Submission Details	
Submission Time:	2012-02-07 16:16:36
Number of CPU's:	1
Submission Host:	sasserver03
Submission Command:	/sas93/config/compute/Lev1/SASApp/GridServer/sasgrid SASBATCHSASPGM:eg_sim.s as SASWANTJOBINFO:1 "SASBATCHHWKDIR:/sas93/sharedwork/gridshared/griduser01 /SASGSUB-2012-02-07_16.17.14.177_EG_C_sim_01"
Resource Requirements:	SASApp
Output File:	/dev/null
Error File:	/dev/null
Pending/Suspended Reasons	
Reasons:	User has reached the per-user job slot limit of the queue
Pending Time:	54 Minutes

Figure 37: Pending Reason

By zooming in on some of the eg_sim jobs, you can verify that griduser01, for example, has 2 jobs running and one pending. By clicking on the JobID (13841) you can access more detailed information about the job and see that the pending reason for this job is indeed the per-user limit enforced by the queue.

RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

<< Previous Showing Rows 1 to 50 of 50 [1]

JobID	Name	Status	State Changes	Nice Level	User ID	CPU Usage	CPU Effic	Execution Host	Running Queue
13842	stepsort_1	RUNNING	13	10	sasdemo	19.27m	37.74%	sgcwin02017.race.sas.com	batch_q
13843	stepsort_2	SSUSP	16	10	sasdemo	19.87m	41.10%	sgcwin02016.race.sas.com	batch_q
13844	stepsort_3	RUNNING	5	10	sasdemo	24.52m	43.68%	sgcwin02018.race.sas.com	batch_q
13845	stepsort_4	RUNNING	13	10	sasdemo	19.63m	38.46%	sgcwin02014.race.sas.com	batch_q
13846	stepsort_5	RUNNING	15	10	sasdemo	16.48m	32.12%	sasserver04	batch_q
13847	stepsort_6	RUNNING	17	10	sasdemo	19.15m	38.99%	sgcwin02013.race.sas.com	batch_q
13848	stepsort_7	RUNNING	19	10	sasdemo	14.92m	33.06%	sasserver02	batch_q
13849	stepsort_8	RUNNING	15	10	sasdemo	16.05m	31.45%	sasserver03	batch_q
13850	stepsort_9	RUNNING	17	10	sasdemo	18.83m	38.40%	sgcwin02015.race.sas.com	batch_q
13851	stepsort_10	RUNNING	15	10	sasdemo	13.37m	28.41%	sasserver05	batch_q
13852	stepsort_11	PEND	1	10	sasdemo	-	-	-	batch_q
13853	stepsort_12	PEND	1	10	sasdemo	-	-	-	batch_q
13854	stepsort_13	PEND	1	10	sasdemo	-	-	-	batch_q

Job Detail

General Information			
JobID:	13852	Status:	PEND
Jobname:	stepsort_11		
Project:	default	License Project:	NULL
Queue:	batch_q	Cluster Name:	grid_cluster
User:	sasdemo	User Group:	
Submission Details			
Submission Time:	2012-02-07 16:17:40	Number of CPU's:	1
Submission Host:	sasserver03		
Submission Command:	/sas93/config/compute/Lev1/SASApp/GridServer/sasgrid SASBATCHSASPGM:stepsort .sas SASWANTJOBINFO:1 "SASBATCHWRKDIR:/'sas93/sharedwork/gridshared/sasdemo/SASGSUB-2012-02-07_16.18.17.962_stepsort_11"		
Resource Requirements:	SASApp		
Output File:	/dev/null		
Error File:	/dev/null		
Pending/Suspended Reasons			
Reasons:	The CPU utilization (ut) is beyond threshold: 6 hosts Queue's per-CPU job slot limit reached: 4 hosts Closed by LSF administrator: 1 host		
Pending Time:	1.04 Hours		

Figure 38: Pending reason(s) for the StepSort Job

Figure 38 shows there is no such limit on the number of jobs per user in batch_q, as sasdemo is clearly running many jobs. However, there are still some StepSort jobs that remain pending. By looking more closely at one in particular, you get a detailed view of the reasons for that. Out of 11 possible hosts, none fits the bill: one is closed by the administrator, 6 have a CPU utilization (ut) that is too high to get a new job, and the remaining 4 have reached the PJOB_LIMIT.

These examples show how it is possible to verify and validate parameters without having to involve the users. There are more things that could be checked, such as the NICE levels, and the rotation of some of the values between night and day.

Then again, Grid tuning is more of an art than a science, and nothing fully prepares you for what real SAS users will be doing to the environment. And so, regular monitoring and modifying is the key to a healthy Grid environment.

CONCLUSION

RTM and SASGSUB might look at first glance like two very different products, but as this paper shows, combining them can have very powerful applications. Not only can it help in validating advanced Grid configurations, but it also provides a fairly simple way of stressing hardware. While this is not a definitive guide on RTM or SASGSUB by any stretch of the imagination, this paper should have made you either more familiar with - or a lot more curious about - RTM and SASGSUB.

Finally, and to expand on the topics covered here, it would be interesting to build a test suite in order to stress the component that is the most critical to good Grid performance: the entire storage infrastructure. By building a set of storage-specific ELIMs, passing their information to RTM, and building some io intensive SAS programs, you are able

RTM and SASGSUB, the Power to Know®... what your grid is doing, continued

to verify how the storage reacts when under heavy load, and what needs to be tuned, if anything. Those same ELIMs could then be used to make smarter decisions as to where a job should be run: adding one or more io-based parameters to the usual cpu-utilization one (ut) is sure to yield some interesting results.

RECOMMENDED READING

- SAS Institute Inc. 2011. Grid Computing in SAS 9.3. Available at <http://support.sas.com/documentation/onlinedoc/gridmgr/index.html>
- Platform Computing Inc. 2011. Platform RTM Administrator Guide for SAS (Version 2.0.7). Available at <http://support.sas.com/rnd/scalability/platform/index.html>
- Platform Computing Inc. 2011. ReadMe for Installing Platform RTM for SAS. Available at http://support.sas.com/demosdownloads/sysdep_t1.jsp?packageID=000669

CONTACT INFORMATION

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

Name: Erwan Granger
Enterprise: SAS Institute Inc
Address: SAS Campus Drive
E-mail: Erwan.Granger@sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.