

Get cozy with your data sources using LSF queue technology: One LSF Cluster to Rule them All

Steven B. Garrison, Sr., Bank of America, Chief Platform Engineer, SAS Center of Excellence

ABSTRACT

Do you have data sources located all over the world? Is your user data travelling needlessly across long distances? Do you experience network lag because your production SAS grid cluster is not housed in the same physical location as each of your data sources? The answer is not to create multiple production SAS environments, one in each data location. The solution is to expand your grid cluster to span multiple data centers using dedicated LSF queues for each 'remote' server(s).

In an attempt to save licensing and resource costs, many global corporations are moving towards a single enterprise SAS grid versus multiple internal SAS environments. Many of these legacy SAS environments were purposely co-located beside critical data sources to provide the fastest possible response time when pulling or pushing data to those sources of data. So, how does an organization architect a single enterprise SAS grid utilizing data sources located all over the world, without compromising performance?

One way to eliminate network latency is by cohabitating compute nodes with your critical data. LSF queue technology makes this possible. Having a queue assigned to each worker node in each location where critical data is housed, there is no network lag because there is no traversing of distance. The server performing the work is in the same location as the data source. A single production LSF cluster can span the globe to submit a SAS job from Virginia to be executed on a server in Tokyo (or wherever a critical data source may be located).

This use of LSF queue technology creates a single production cluster to "rule them all."

INTRODUCTION

This document is designed for SAS grid administrators that have an in-depth understanding of the underlying LSF technology that makes their grid function. We will discuss how to use Platform LSF queue technology to spawn jobs from one datacenter and execute them on a compute node in another distinct datacenter. This will allow SAS grid administrators to co-locate servers next to data sources and eliminate potentially increased response times.

Readers should be familiar with the below SAS grid/Platform LSF terms.

Grid Basics review

Cluster- A group of computers (hosts/servers/nodes) running Platform LSF that work together as a single unit, combining computing power and sharing workload and resources. A cluster provides a single-system image for a network of computing resources. Hosts can be grouped into clusters in a number of ways. A cluster could contain:

- All the hosts in a single administrative group
- All the hosts on one file server or sub-network
- Hosts that perform similar functions

Job- A unit of work that is run in the Platform LSF system. A job is a command submitted to LSF for execution. LSF schedules, controls, and tracks the job according to configured policies. Jobs can be complex problems, simulation scenarios, advanced calculations, or anything that requires compute power.

Queue- A cluster-wide container for jobs. All jobs wait in queues until they are scheduled and dispatched to hosts/servers. Queues do not correspond to individual hosts; each queue can use all server hosts in the cluster, or a

configured subset of the server hosts. When you submit a job to a queue, you do not need to specify an execution host. LSF dispatches the job to the best available execution host in the cluster to run that job. Queues implement different job scheduling and control policies.

CURRENT GRID ARCHITECTURE

Our current SAS grid environment consists of a grid client gateway server, a grid master control server (which also contains the Metadata tier) and a collection of application/compute servers. These compute/application nodes are what I like to refer to as 'worker' nodes. The worker nodes actually run the jobs submitted into the cluster from the grid master control server via the grid client gateway (or a SAS client, such as Enterprise Guide). This collection of hardware typically resides within the same physical data center.

Let's use the below diagram (Figure 1) of our production cluster named PROD_cluster. This cluster will be our baseline and is our enterprise solution. It is important to note that in our PROD_cluster there is one LSF queue in use, appropriately named the PROD queue. Currently all jobs that are submitting from the Grid Client Gateway are launched onto any node within the PROD queue from the Grid Master Control Server.

This PROD queue spans all of the worker nodes within the PROD_cluster of servers; ie- when jobs are submitted, they are defaulted to the PROD queue that submits jobs to any of the 10 worker nodes (ATL_worker1-ATL_worker10).

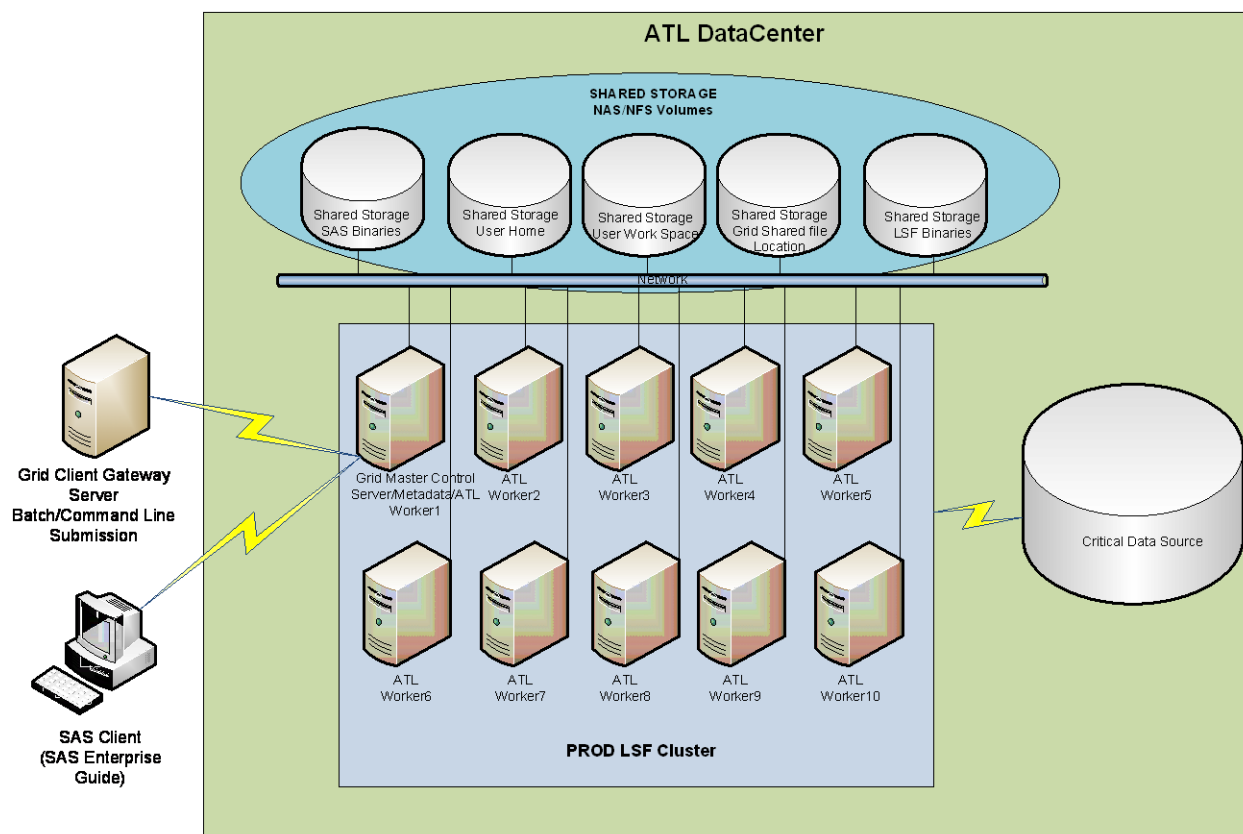


Figure 1. ATL Datacenter specifications

DIRECTIVE- ASSIMILATE REMOTE SAS INSTALLATION

In order to save licensing costs and overhead of multiple SAS installations within our large corporation, management has given us a new directive: consume another remote SAS installation. As we all know, most SAS installations are built on top of or next to their critical data sources. Simply, the co-location of the SAS environment with its critical data source ensures much better performance because there is no need for the data to traverse distance. This model is spreading throughout corporate America and presents a unique problem for SAS grid administrators: How do you maintain the integrity of one true production platform, without sacrificing performance, even when you have critical data sources housed in multiple data centers?

Before we solve the issue, let's understand the smaller remote SAS installation to be absorbed into the enterprise SAS GRID.

Remote Legacy SAS environment Specifications

The BOS legacy SAS environment was created in the BOS datacenter because of the DB2 database that is located within BOS datacenter (Figure 2). This is a common scenario for larger companies that have many SAS installations. Each SAS installation is usually built on top of or physically next to its primary data source. The BOS SAS installation was co-located with the DB2 data to increase performance between the BOS SAS environment and the BOS DB2 database. Other SAS installations, such as the enterprise SAS grid PROD_cluster experience network latency when connecting to the BOS DB2 data because they do not reside in the BOS data center.

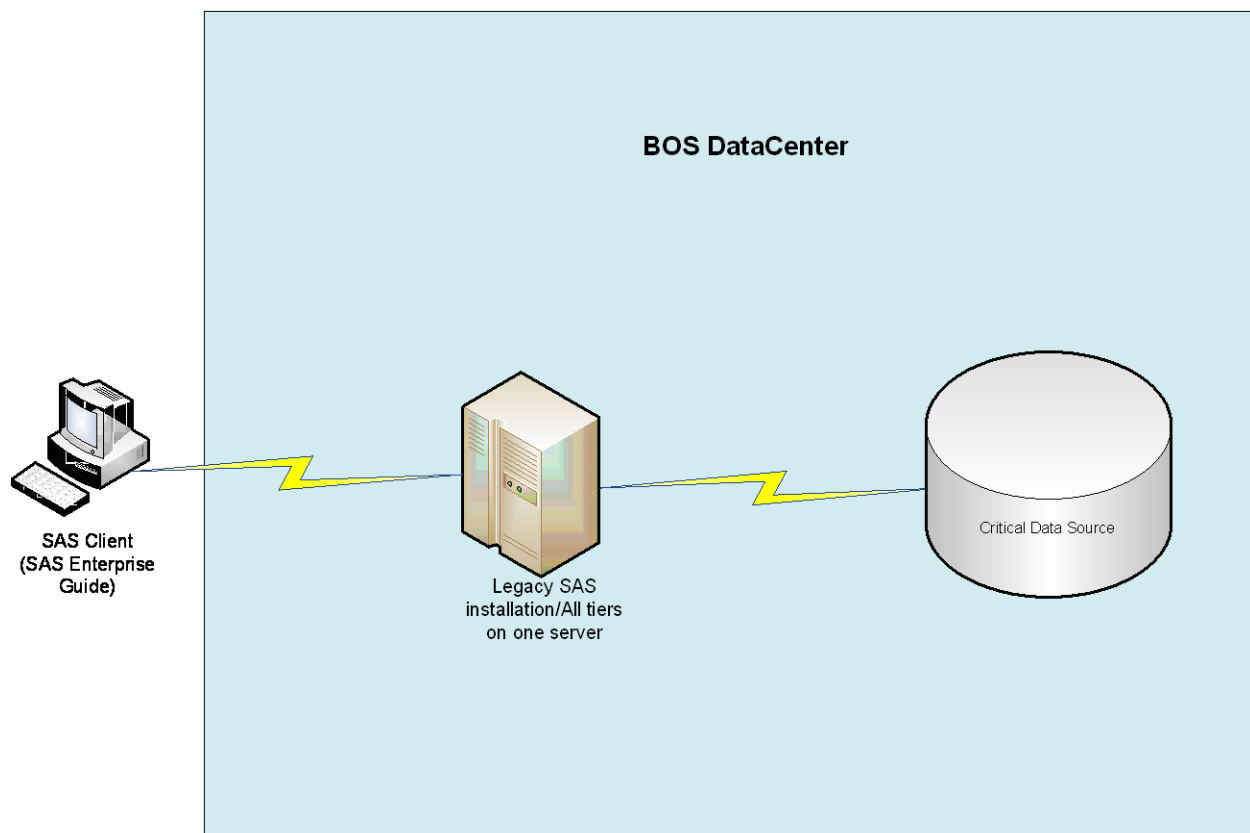


Figure 2. BOS Remote Datacenter specifications

TIME FOR A CHANGE

To accommodate the absorption of the remote BOS SAS installation into the existing SAS grid enterprise environment, we must first change our current PROD_cluster configuration. We'll want to change the name of one of current worker nodes to meet each datacenter server's naming standard. We also want to ensure that we are not violating our existing license structure. Our existing servers are prefaced with ATL for identification of hardware purposes between the datacenters (ATL_worker1- ATL_worker10). Because our ATL worker node is now moving to BOS, the name of the server will change. Thus the ATL_worker10 host name will change to BOS_worker1.

After each update to the configuration files, you will need to run lsadmin/badmin reconfig's to pick up the changes.

Step One: Update lsfc.conf

First, we need to update the list of servers that comprise the PROD_cluster. This can be changed in the lsfc.conf file found in the \$LSF_ENVDIR directory.

Below is the current list of servers for the PROD_cluster before the change of the server name due to his change of location.

```
LSF_SERVER_HOSTS="ATL_worker1 ATL_worker2 ATL_worker3 ATL_worker4 ATL_worker5
ATL_worker6 ATL_worker7 ATL_worker8 ATL_worker9 ATL_worker10"
```

This value should reflect the new name of the old server by updating the LSF_SERVER_HOSTS entry below. ATL_worker10 has been updated with the new compute node name of BOS_worker1

```
LSF_SERVER_HOSTS="ATL_worker1 ATL_worker2 ATL_worker3 ATL_worker4 ATL_worker5
ATL_worker6 ATL_worker7 ATL_worker8 ATL_worker9 BOS_worker1"
```

Step Two: Update lsbf.hosts

The next file that needs to be changed is the \$LSF_ENVDIR/lsbatch/PROD_cluster/configdir/lsbf.hosts. Below you can see that the file contains a list of the servers that are eligible to run jobs. The below lsbf.hosts file does not reflect the change of the new remote server in the BOS datacenter.

Begin Host

HOST_NAME	rlm	ls	MXJ	DISPATCH_WINDOW	tmp	pg	ut
default	()	()	!	()	()	()	()
ATL_worker1	()	()	300	()	()	()	0.95
ATL_worker2	()	()	300	()	()	()	0.95
ATL_worker3	()	()	300	()	()	()	0.95
ATL_worker4	()	()	300	()	()	()	0.95
ATL_worker5	()	()	300	()	()	()	0.95
ATL_worker6	()	()	300	()	()	()	0.95
ATL_worker7	()	()	300	()	()	()	0.95
ATL_worker8	()	()	300	()	()	()	0.95
ATL_worker9	()	()	300	()	()	()	0.95
ATL_worker10	()	()	300	()	()	()	0.95

End Host

Update the \$LSF_ENVDIR/lsbatch/PROD_cluster/configdir/lsbf.hosts file to reflect the new hostname of the BOS server. Using our example we'll change ATL_worker10 to BOS_worker1.

Begin Host

HOST_NAME	rlm	ls	MXJ	DISPATCH_WINDOW	tmp	pg	ut
-----------	-----	----	-----	-----------------	-----	----	----

```

default      ()      ()      !      ()      ()      ()      ()
ATL_worker1  ()      ()      300    ()      ()      ()      0.95
ATL_worker2  ()      ()      300    ()      ()      ()      0.95
ATL_worker3  ()      ()      300    ()      ()      ()      0.95
ATL_worker4  ()      ()      300    ()      ()      ()      0.95
ATL_worker5  ()      ()      300    ()      ()      ()      0.95
ATL_worker6  ()      ()      300    ()      ()      ()      0.95
ATL_worker7  ()      ()      300    ()      ()      ()      0.95
ATL_worker8  ()      ()      300    ()      ()      ()      0.95
ATL_worker9  ()      ()      300    ()      ()      ()      0.95
BOS_worker1  ()      ()      300    ()      ()      ()      0.95
End Host

```

Step Three: Update lsb.queues

The following queue definitions are similar to what you will find in your \$LSF_ENVDIR/lsbatch/PROD_cluster/configdir/lsb.queues file. There are multiple 'out-of-the-box' queue definitions defined below of normal, owners, priority, short and idle. We can see that queues have the following attributes associated with them:

- Name, which uniquely identifies the queue
- Priority, where a larger integer is a higher priority
- Queue limits, that restrict hosts, number of jobs, users, groups, processors, etc.
- Standard UNIX limits: memory, swap, process, CPU, etc.
- Scheduling policies: FCFS, fairshare, preemptive, exclusive
- Administrators
- Run conditions
- Load-sharing threshold conditions, which apply load sharing to the queue
- UNIX nice(1) value, which sets the UNIX scheduler priority

```

Begin Queue
QUEUE_NAME=normal
PRIORITY=30
DESCRIPTION=For normal low priority jobs, running only if hosts are lightly loaded.
NICE=20
End Queue

```

```

Begin Queue
QUEUE_NAME=owners
tmp=1/0
DESCRIPTION=For owners of some machines. Only users listed in the HOSTSsection can
submit jobs to this queue.
PRIORITY=43
pg=4/15
io=30/200
swp=4/1
r1m=1.2/2.6

```

```

NICE=10
End Queue

Begin Queue
QUEUE_NAME=priority
PRIORITY=43
NICE=10
PREEMPTION=PREEMPTIVE
DESCRIPTION=Jobs submitted for this queue are scheduled as urgent jobs. Jobs in this
queue can preempt jobs in lower priority queues.
End Queue

Begin Queue
QUEUE_NAME=short
PRIORITY=35
DESCRIPTION=For short jobs that would not take much CPU time. Scheduled with higher
priority.
r1m=0.9/
CPULIMIT=15
NICE=20
End Queue

Begin Queue
QUEUE_NAME=idle
DESCRIPTION=Running only if the machine is idle and very lightly loaded.
it=10/1
PRIORITY=20
pg=4.0/15
r15s=0.3/1.5
r1m=0.3/1.5
NICE=20
End Queue

```

THE PROD QUEUE

Below is our custom PROD queue definition found in the \$LSF_ENVDIR/lsbatch/PROD_cluster/configdir/lsb.queues file. We can see that the PROD queue takes highest priority and spans across servers ATL_worker1 - ATL_worker10. The HOSTS variable will need to be updated to drop the renamed remote worker node name. The new worker node name will be used in the bosprod queue, yet to be defined.

```

Begin Queue
QUEUE_NAME=prod
PRIORITY=100
RERUNNABLE=NO
HOSTS= ATL_worker1 ATL_worker2 ATL_worker3 ATL_worker4 ATL_worker5 ATL_worker6
ATL_worker7 ATL_worker8 ATL_worker9 ATL_worker10

```

```
NICE=0
End Queue
```

THE BOSPROD QUEUE

After updating the \$LSF_ENVDIR/lsbatch/PROD_cluster/configdir/lbqueues file, the HOSTS entry within the PROD queue definition should reflect the drop of the ATL_worker10 server name. Now, the PROD queue will run on all hosts within the PROD_cluster except the new BOS_worker1 node. The intent is for only the BOS users to run on the bosprod queue. However, this example will allow for existing PROD queue users to change their queue by changing the value of a global variable.

A new bosprod queue has been created. It's important to note the preemption option to the prod queue here.

Jobs in this queue can preempt jobs from specified lower priority queues only, or from all lower priority queues by default if the parameter is specified with no queue names.

If you specify a list of lower-priority preemptable queues, you must enclose the list in one set of square brackets. We have defined the bosprod queue to be of slightly lower priority and preemptable by the prod queue. Don't forget to run 'lsadmin reconfig' and 'badmin reconfig' after any changes to the LSF configuration files.

```
Begin Queue

QUEUE_NAME=prod

PRIORITY=100

RERUNNABLE=NO

HOSTS= ATL_worker1 ATL_worker2 ATL_worker3 ATL_worker4 ATL_worker5 ATL_worker6
ATL_worker7 ATL_worker8 ATL_worker9

NICE=0

End Queue
```

```
Begin Queue

QUEUE_NAME=bosprod

PRIORITY=99

PREEMPTION=PREEMPTABLE[prod]

HOSTS=BOS_worker1

RERUNNABLE=NO

NICE=0

End Queue
```

THE MAGIC OF QUEUE ASSIGNMENT USING ESUB SCRIPTING

Now that we've added the configured hardware to the BOS datacenter, updated our server list, updated our 'prod' queue definition and created our 'bosprod' queue definition, it's time to add the logic for queue assignment. This is a critical piece that will allow the BOS user's jobs to run on the remote worker node in the BOS datacenter without specifying a queue at job submission.

An esub command is an administrator-written executable (or script) that can be used to validate, modify, or reject jobs. The esub command is put into \$LSF_SERVERDIR where Platform LSF checks for its existence when a job is submitted, re-started, or modified. If LSF finds an esub command, the command is run. Whether the job is submitted, modified, or rejected depends on the logic that's built into the esub command. Figure 3 depicts a typical user job flow, if an esub script is found it is used. For this example, a user's queue may be updated based on the existence of the esub script and the contents of a global variable.

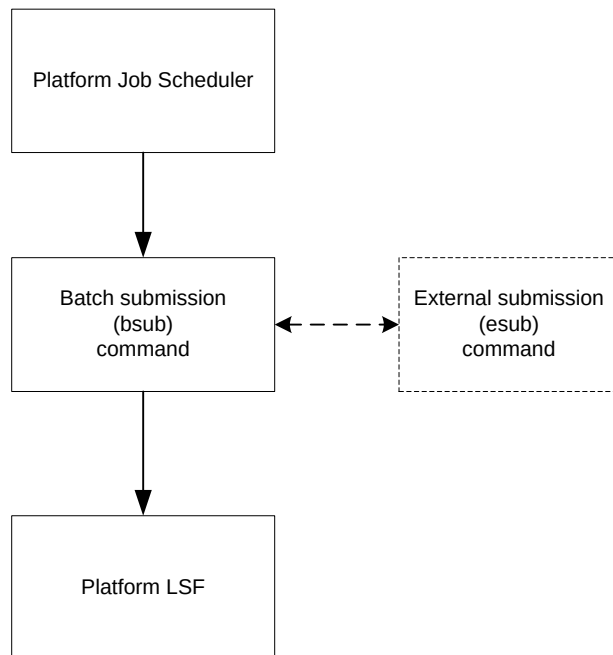


Figure 3. Process flow for job submission with esub script

Below is an esub script example of how to create and add a variable to a user's .bash_profile and then based on the value within, the esub script will use the variable for queue assignment (determine job submission to which host using the queue). Existing prod queue users will not see any difference in job runs and will not have to set the variable. If the \$USTG variable is not set, the prod queue is used as the default for missing entries. In this example, we have added a variable to the environment named \$USTG to the BOS user's .bash_profile with the following code:

```
export USTG=bosprod
```

The below code is the esub script using the logic to determine which queue to be assigned based on the value of \$USTG: if \$USTG='bosprod', then submit the job into the bosprod queue, if not use the prod queue. Note that any existing prod queue user has the ability to update their .bash_profile to include the \$USTG variable and point to the newly created bosprod queue.

```
#!/bin/sh

. $LSB_SUB_PARM_FILE

exec 1>&2

USTGLCASE=`echo $USTG | tr 'A-Z' 'a-z'`

echo 'USTG = "$USTGLCASE"' > $LSB_SUB_MODIFY_FILE
echo 'LOGNAME = "$USERLCASE"' >> $LSB_SUB_MODIFY_FILE

if [ "$USTG" = "bosprod" -o "$USTG" = "prod" ];
then
# submit the job to the appropriate queue as specified by the user
echo "Executing job in the $USER_STAGE environment."
echo "LSB_SUB_QUEUE = \"$USTG\" " > $LSB_SUB_MODIFY_FILE
else
# stage is not specified or unknown; assume production job
echo "Executing job in the prod environment."
echo 'LSB_SUB_QUEUE = "prod"' > $LSB_SUB_MODIFY_FILE
fi
```

The above logic will inspect the variable of \$USTG, if it's the default of prod (set in the global /etc/profile) or if it's blank, the default prod queue will be used. There is no change required for a user to continue to use the prod queue. If the \$USTG variable has been set in the BOS user's profile then the queue will change appropriately to 'bosprod'.

NOW IT'S TIME TO TEST BATCH SUBMISSION

At this point, we have:

- Physically moved and renamed our configured hardware from the ATL datacenter to the BOS datacenter (host name change from ATL_worker10 to BOS_worker1)
- Updated our lsf.conf and lsb.hosts files with our new server information
- Changed our existing PROD queue to remove our remote BOS worker node by updating lsb.queues
- Created a new BOSPROD queue assigned to the BOS worker node by updating lsb.queues
- Run lsadmin reconfig and badmin reconfig commands to pick up any configuration changes
- Created an esub (external submission) script containing the logic to switch our queue based on the existence of a variable in the BOS user's .bash_profile

We are ready to submit a batch job into the PROD_cluster as a BOS user. From the output of the bjobs command, we can see that the job was submitted into our Grid Master Control Server (ATL_worker1) and then because of the existence of the \$USTG variable of 'bosprod', the esub script assigns the job to the bosprod queue and is executed on the remote BOS_worker1 compute node.

The job is submitted from the ATL datacenter, then successfully executed on the BOS remote server. The user experiences identical performance as they did when their SAS installation was located in the BOS datacenter.

```
[root@ATL_workerGCG platcom1]# bjobs -u all -a
```

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
7419	produser1	RUN	prod	ATL_worker1	ATL_worker8	atl_datagrab	Jul 17 16:02
7978	produser4	RUN	prod	ATL_worker1	ATL_worker5	atl_datagrab3	Jul 23 17:00
7991	BOSuser3	RUN	bosprod	ATL_worker1	BOS_worker1	BOS_datagrab	Jul 24 08:18
8088	produser5	RUN	prod	ATL_worker1	ATL_worker1	atl_datagrab	Jul 25 10:07
8090	produser7	DONE	prod	ATL_worker1	ATL_worker1	atl_datagrab	Jul 25 18:00

WE'RE ALL SET FOR BATCH JOB SUBMISSION, ONTO THE CLIENT TOOLS

Implementing Grid Option Sets

A grid options set is a collection of grid options, SAS options, and required resources. Grid options sets are then mapped to a particular SAS client application and one or more specific metadata identities. The purpose of a grid options set is to enable a SAS grid administrator to define a collection of options in SAS metadata that map to one or more SAS client applications. Those options are automatically applied to the workload that is submitted to the grid based on the identity of the user accessing the client application. SAS client applications such as the SAS Grid Manager Client Utility have been enhanced to support grid options sets.

We have successfully changed the behavior of our PROD_cluster to use a new queue of bosprod queue when appropriate by implementing the esub script. But now we need to perform the same type of logical queue assignment using the SAS client tools. This can be accomplished by creating a new grid option set.

First let's take a look at our existing entry for our PROD queue, below. Navigate to the setting by:

1. Open SAS Management console
2. Choose the Server Manager option
3. Choose the default application server name of SASApp
4. Choose SASApp- Logical Grid Server
5. Select the options tab for the SASApp- Grid Server

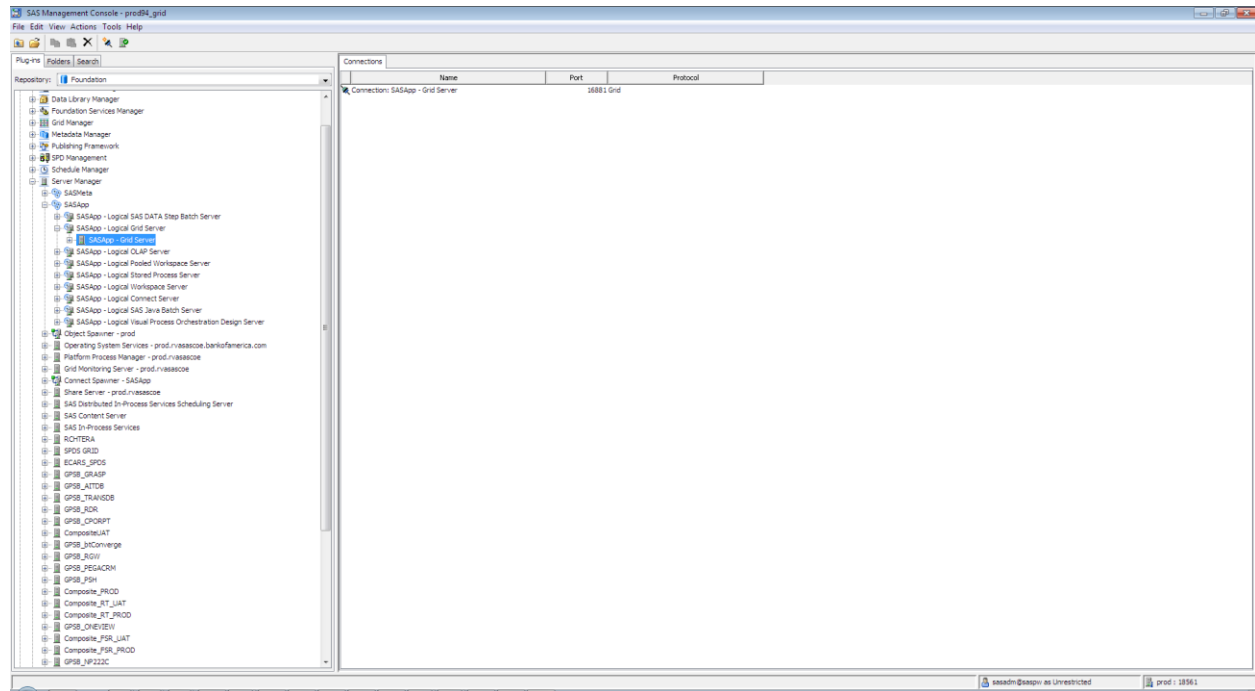


Figure 4. SASApp- Grid Server

SASApp - Grid Server Properties

General Options Notes Extended Attributes Authorization

Required Fields

Provider: Platform

Grid Command: /sas/9.4/prod/config/Lev1/SASApp/GridServer/sasgrid

Optional Fields

SAS Options:

Required Resources: SASApp

Grid Options: queue=prod

Grid Options Set Mappings

#	SAS Application /	Mapped Identities	Grid Options Set
---	-------------------	-------------------	------------------

New Edit Delete

Advanced Options

OK Cancel Help

Figure 5. SASApp- Grid Server Options tab with default queue specified

In Figure 5, we can see that by specifying the queue=prod value in the Grid Options field, all user workspace sessions launched will run on the SASApp application via the prodqueue. This means that even with the current changes to batch job submission process, the SAS client tools (such as SAS Enterprise Guide) will run their code on the ATL_worker1- ATL_worker9 worker nodes.

There are two tasks associated with creating a new grid option set for our new bosprod queue users: creating the grid option set and specifying that each application is grid capable.

Creating a New Grid Option Set

First we will create new Grid Option Set Mappings by selecting the 'New' button from the Options tab of the SASApp-Grid Server.

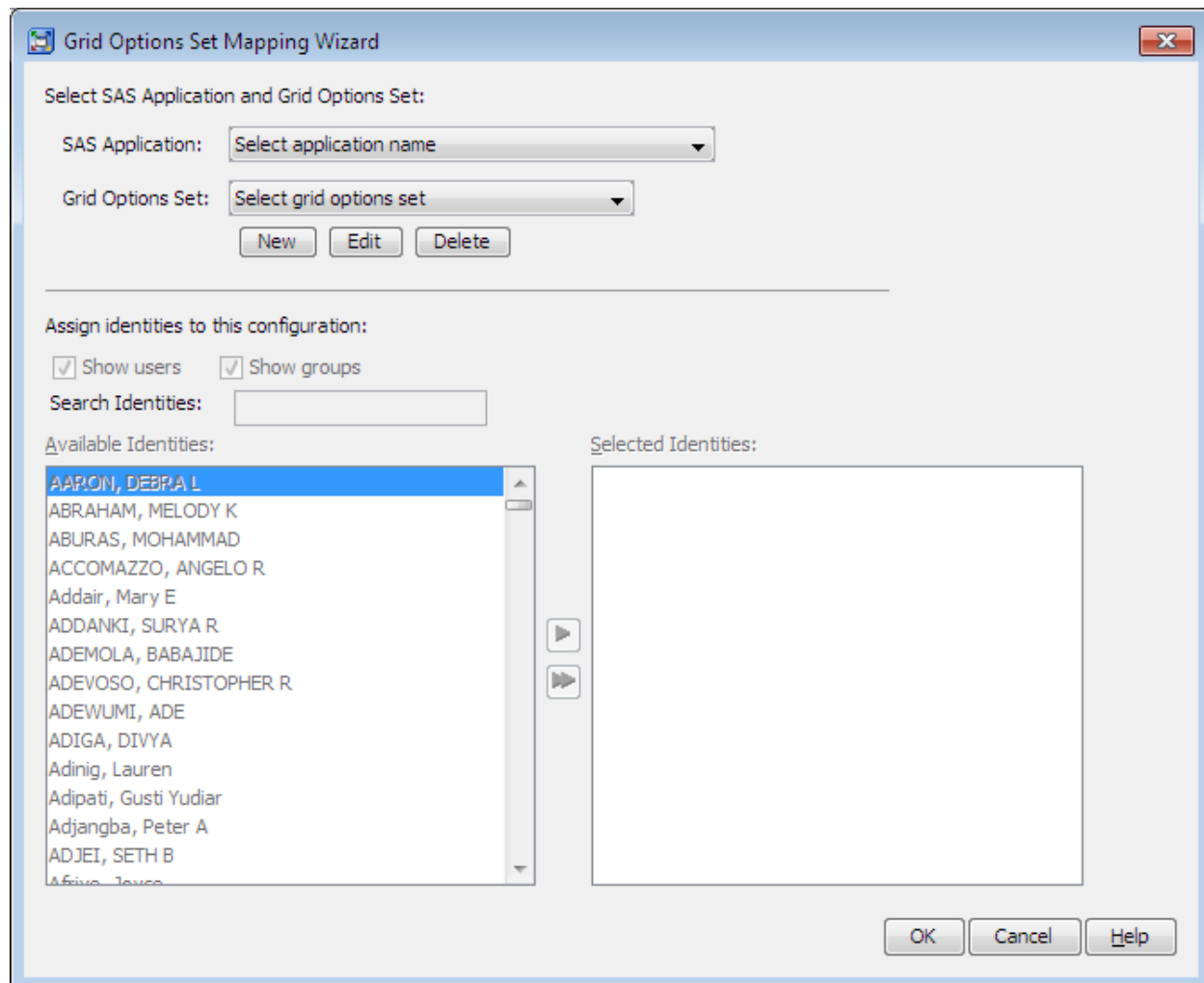


Figure 6. Creating New Grid Option Set

Next, select the application name of SASApp and create a new Grid Option Set of 'bosprod'. We'll also create a new user group to house the BOS users named BOSPROD_QUEUE. After the changes, we should have a new grid option set with the following attributes in Figure 7.

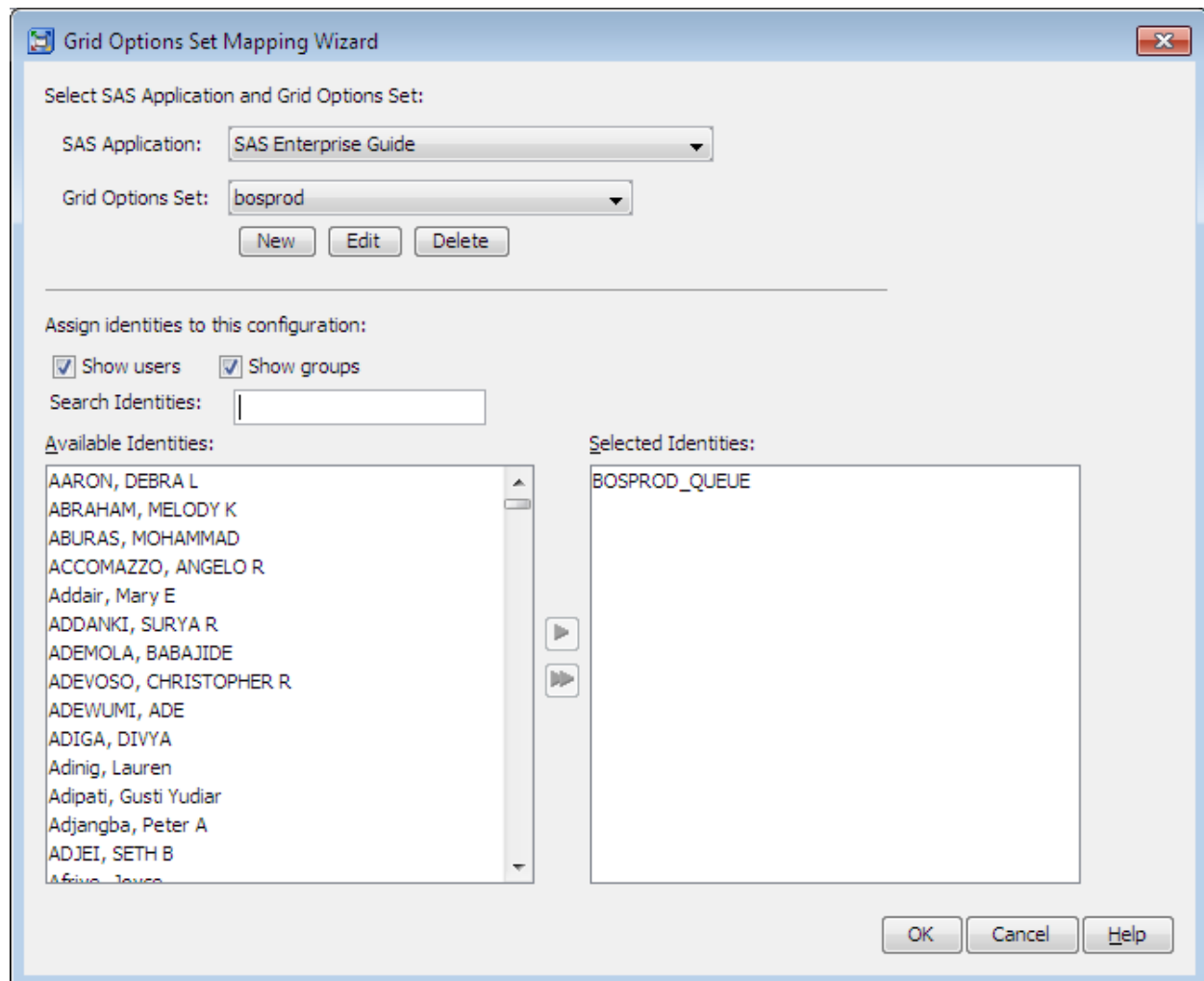


Figure 7. Creating New Grid Option Set cont.

After clicking 'OK', we now have the new grid option set mappings added to the SASApp- Grid Server, identical to Figure 8 below. By creating the BOSPROD_QUEUE user group with the appropriate BOS users within, we can control access to the remote server. All users that need to use the bosprod queue must reside within the BOSPROD_QUEUE user group to submit their code onto the remote server.

SASApp - Grid Server Properties

General Options Notes Extended Attributes Authorization

Required Fields

Provider:

Grid Command:

Optional Fields

SAS Options:

Required Resources:

Grid Options:

Grid Options Set Mappings

#	SAS Application /	Mapped Identities	Grid Options Set
1	SAS Enterprise Guide	BOSPROD_QUEUE	bosprod

New Edit Delete

Advanced Options

OK Cancel Help

Figure 8. Creating New Grid Option Set (cont.)

Specify the application is grid capable

The only applications that you can associate with a grid options set are ones that have been identified as being grid capable. Applications are identified as grid capable either by the application's deployment process or by specifying the `isGridCapable` keyword through the application's property values.

To verify which applications have been identified as grid capable during deployment, you can define a grid options set mapping before you have set the `isGridCapable` keyword on any applications through the application properties. The applications that are listed as available for mapping are the ones that were identified as grid capable during deployment.

For all other applications that you want to map to a grid options set, you must manually identify each of them as grid capable. In order to apply the `isGridCapable` keyword to the correct application when running a job, SAS Grid Manager must be able to match the application name as identified on the application folder with the name that is sent by the application when the object spawner starts the workspace server. If they do not match, then you can use wildcards to help match the names. This situation might occur if the application sends a different name from one invocation to the next or if you have multiple versions of the same application that you want to map to the same grid options set.

Let's specify that the EG application is grid capable. Navigate to the setting:

1. Open SAS Management console

2. Choose the Folder tab
3. Open the System folder, then Applications directory beneath
4. In the properties window, click Add next to the Keyword area
5. In the Add Keyword dialogue box enter isGridCapable (See Figure 9)

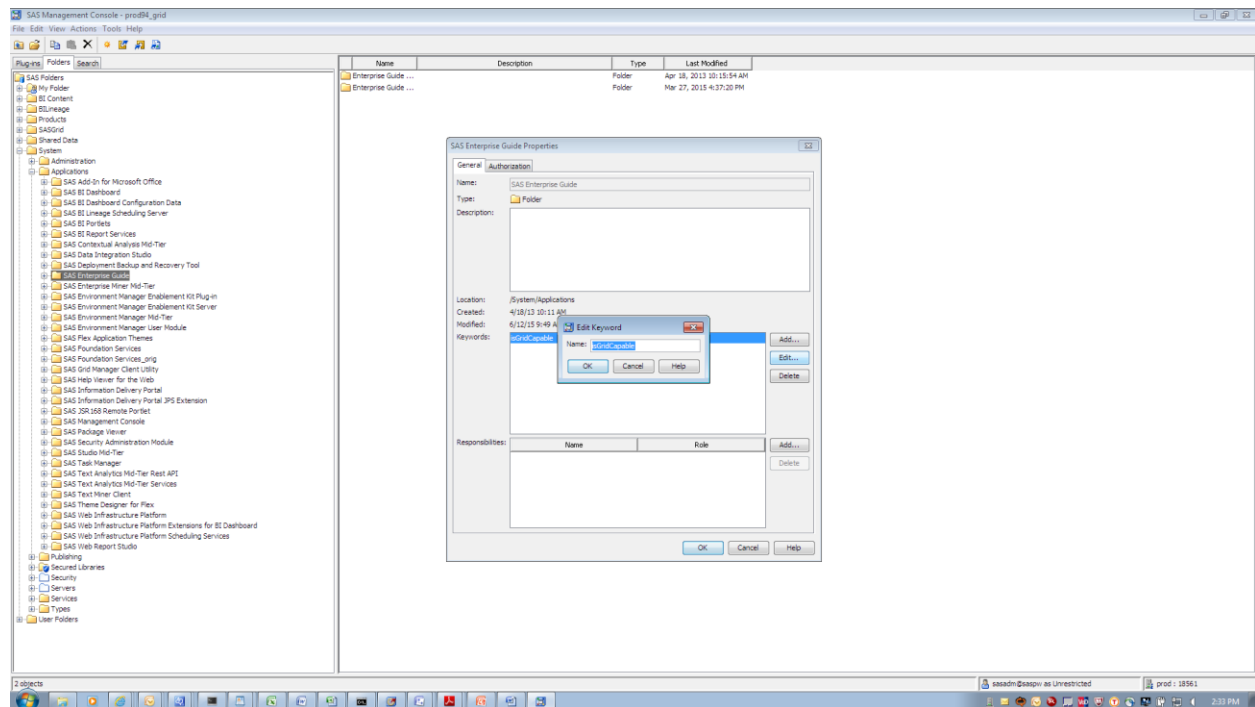


Figure 9. Specifying EG is grid capable

NOW IT'S TIME TO TEST OUR CLIENT TOOLS

To test our changes to the environment and allow our BOS users to run their Enterprise Guide code on the remote server using the bosprod queue, we need to ensure that we have them in the BOSPROD_QUEUE metadata user group. At this point, we have included all of the BOS users in the BOSPROD_QUEUE metadata user group.

After we've updated the group with the appropriate users, let's open the client tool and connect to SASApp (figure 10). If setup correctly, you should be able to validate that the BOS user is indeed running their code on the BOS_worker1 host via SAS Enterprise Guide.

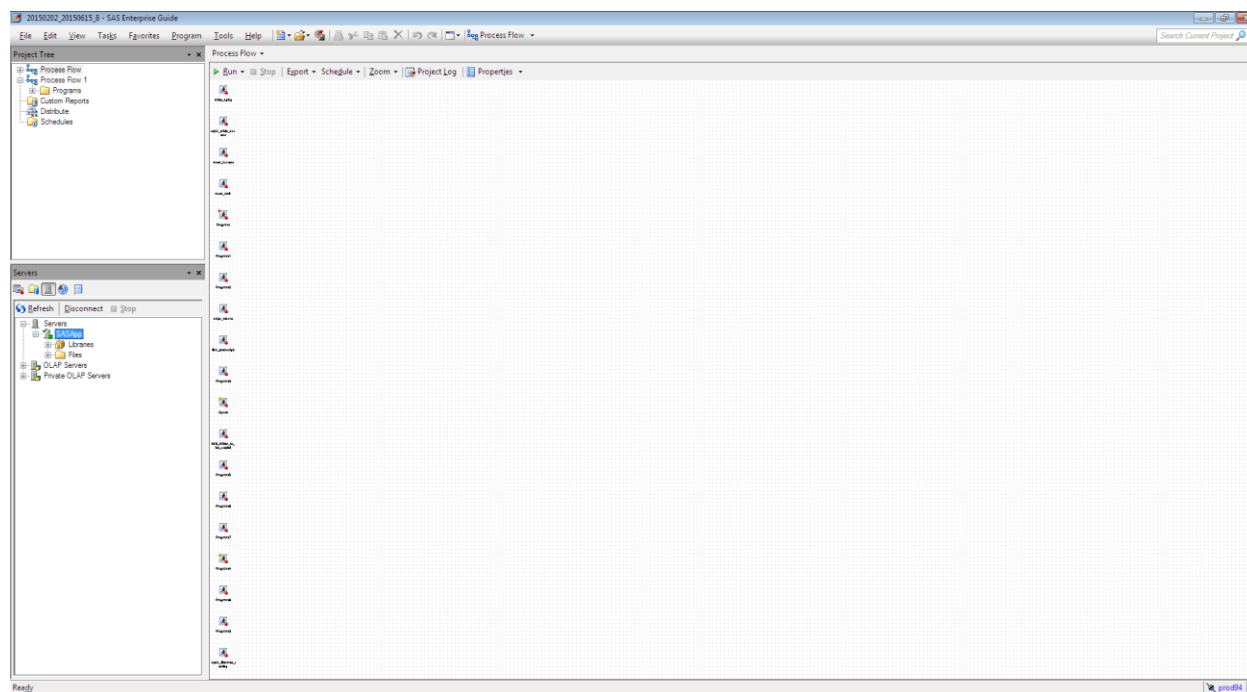


Figure 10. SAS Enterprise Guide session for BOS user

Validate by running the bjobs command for all jobs and users below :

```
[root@lrcha0k7pappr platcom1]# bjobs -u all -a
```

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
7419	produser9	RUN	prod	ATL_worker1	ATL_worker6	*rid:31991	Jul 17 16:02
7978	produser3	RUN	prod	ATL_worker1	ATL_worker7	*Grid:5539	Jul 23 17:00
7991	produser7	RUN	prod	ATL_worker1	ATL_worker1	*xoln.9936	Jul 24 08:18
8092	bosuser5	RUN	bosprod	ATL_worker1	BOS_worker1	*EG.17578	Jul 26 14:10

We can see that bosuser5 from the BOSPROD_QUEUE metadata user group has successfully connected to SASApp using SAS Enterprise Guide and their code is being submitted to the BOS_worker1 remote server.

A LOOK AT OUR END RESULT: ONE CLUSTER TO RULE THEM ALL

Finally, we've implemented all the necessary changes to be able to co-locate our worker node with our critical data sources between the two datacenters of ATL and BOS. We have effectively mitigated the network lag associated with the cross datacenter connectivity because we are able to eliminate the network travel. By physically moving a compute node of ATL_worker10 to the BOS datacenter, the data does not have to traverse the network between the datacenters. The BOS users are able to continue to run their code within the same physical datacenter as the BOS DB2 data using the bosprod queue (that only runs on BOS_worker1). We can see from Figure 11 (below), the new architecture of the PROD_cluster includes the existing hardware worker nodes and the remote server as well. This combination of hardware within the PROD_cluster successfully alters the PROD_cluster to span the datacenters while maintaining the integrity of a single production platform. By maintaining the integrity of the single PROD_cluster instance we have implemented the concept of 'one grid to rule them all'.

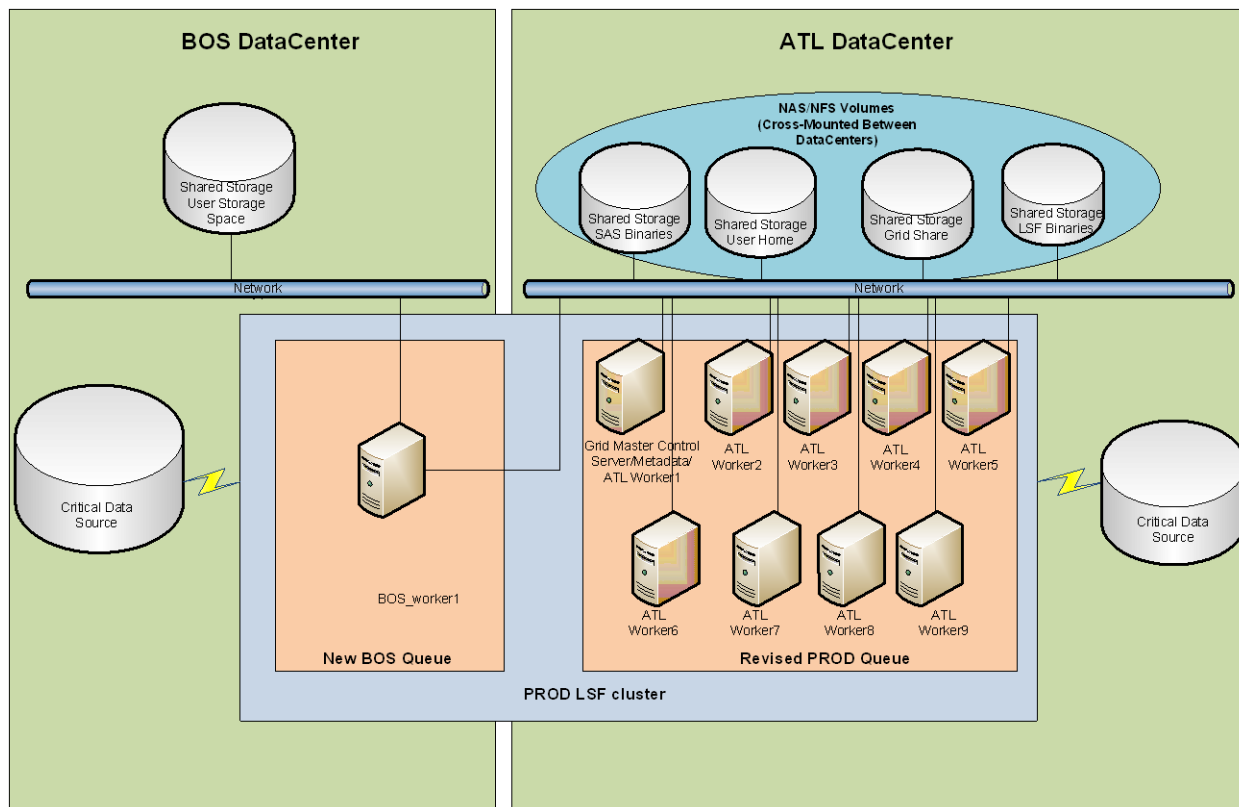


Figure 11. PROD_cluster spanning DataCenters; One Grid to Rule them All

An Important note on storage

Shared Storage

Figure 11(above) depicts the shared storage necessary to make this architecture possible. There are four NAS/NFS data volumes that were cross-mounted between the two datacenters that have little i/o traffic that are critical to this model. They include:

1. The storage volume(s) that includes the SAS executables and configuration files
2. The storage volume that includes the LSF binaries and configuration files
3. The storage volume that houses the 'Grid Share' files
4. The storage volume that includes the user home space

The storage volume(s) that houses the SAS binaries is important to any SAS Grid design as all worker nodes will need to use the same SAS binaries and configuration files. The volume that contains the LSF binaries and configuration files is needed to ensure that identical LSF executables are being shared across the worker nodes. The storage volumes that includes the LSF 'grid shared files' and the user home space is a necessity because the SASGSUB* directory/files will need to be copied from the shared location and back to the user's home space.

The user storage area (Figure 11) was moved from ATL datacenter to the BOS datacenter to avoid network latency between the final landing location of the data and the new remote host of BOS_worker1. All of the jobs submitted to the remote host will potentially write to the user storage location in BOS. When accessed via EG code, this storage volume is local to the remote machine that the code will execute on.

CONCLUSION

Using Platform LSF queue technology SAS grid administrators can completely change the scope and architecture of their existing LSF clusters. By maintaining the integrity of an existing SAS enterprise grid, it is possible to reduce licensing costs associated with multiple SAS installations and eliminate the overhead associated with maintaining multiple production SAS environments. Using this model that relies on LSF queue technology, it's possible to co-locate worker nodes with critical sources of data and prevent network latency by eliminating it.

REFERENCES

- SAS Grid Computing in SAS 9.4, Third Edition
- SAS® 9.4 Intelligence Platform Application Server Administration Guide
- Administering Platform LSF, Version 8.0, Platform Computing Inc.
- Implementing Site Policies for SAS Scheduling with Platform JobScheduler
<http://support.sas.com/resources/papers/JobScheduler.pdf>
- Best Practices for Data Sharing in a Grid Distributed SAS Environment
http://support.sas.com/rnd/scalability/grid/Shared_FileSystem_GRID.pdf

RECOMMENDED READING

Tran, Allen; Williams, Randy; Wong, Alan. 2005. "SAS® Scheduling: Getting the Most Out of Your Time and Resources." Cary, NC: SAS Institute, Inc.

Available at <http://support.sas.com/resources/papers/SASScheduling.pdf>

Howell, Andrew. 2014. "Queues for Newbies – How to speak LSF in a SAS® World." *Proceedings of the SAS Global 2014 Conference*. Cary, NC: SAS Institute Inc.

Available at <http://support.sas.com/resources/papers/proceedings14/1868-2014.pdf>

ACKNOWLEDGEMENTS

The author would like to thank Brian Graham and Glenn Horton from SAS Institute Inc. for their constant support to make these concepts and this paper possible.

The author would also like to thank Laura M. Garrison for reviewing the contents of this paper and making suggestions on how to improve it.

CONTACT INFORMATION

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

Name: Steve Garrison
Enterprise:
Address: 1245 Timber Lake Trl
City, State ZIP: Cumming, GA 30041
Work Phone: 704.
E-mail: steve.garrison@bankofamerica.com, sbgarr3@yahoo.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.

