Paper 367-2011

Effectively Implementing SAS® Grid Architectures in Conjunction with Non-Grid Aware Processes

William Nasuti, SAS Institute Inc., Cary, NC

ABSTRACT

SAS® Grid Computing provides enterprise class governance which allows many SAS® applications to run within a centrally managed grid infrastructure. Employing SAS® Grid Manager affords enterprise organizations the opportunity to utilize their underlying hardware resources more efficiently. In practice, there are many situations where grid nodes are running non-grid aware processes in conjunction with grid aware processes. In order to alleviate this inefficiency, SAS Grid Manager provides a plethora of controls to ensure that resources are governed in an enterprise environment. This process is very straightforward when the governing grid environments are comprised of exclusive grid aware processes. The utilization of non-grid aware processes introduces unique challenges. This paper presents and discusses a viable architecture to improve on grid environments that consist of both grid aware and non-grid aware processes.

INTRODUCTION

SAS Grid Manager is often mistakenly referred to as a single entity, but in reality, it has many facets. SAS Grid Manager provides an abundant functional toolset that can improve the performance and resource utilization of SAS applications and programs. Before delving into specific architectural details surrounding the SAS Grid, it is important to build a foundational understanding of SAS Grid Manager by discussing the different ways SAS Grid Manager enriches performance and resource utilization.

SAS Grid Computing employs multiple computers to improve hardware utilization and reduce processing times. This method is often referred to in the High Performance Computing (HPC) umbrella. SAS Grid Manager computing is built on the Platform™ LSF™(a third-party product) that enables the SAS Grid to manage workloads across multiple platforms. By employing this third-party product, SAS can be decoupled from the underlying operating system (OS), and as a result, SAS applications can manage workloads more efficiently for many virtual and physical platforms.

Here are several examples of how you can utilize this technology with SAS.

- **Multi-User Workload Balancing** – By submitting jobs to the grid rather than to an individual machine, the workload can be spread among all the machines in the grid. Using a grid provides a plethora of enterprise class IT governance. For example, it can ensure that machines are not overloaded, and workloads are distributed based upon resource requirements. It also allows for assigning priorities based on users or applications. It can even be used to suspend or preempt jobs. Multi-user workload balancing enables SAS to ensure that customers are fully utilizing their SAS resources.
- **Parallelized Workload Balancing** – This adds a whole new dimension to SAS processing. Instead of running a job serially, you can divide the jobs into smaller pieces that can be run in parallel. Dramatic performance gains can be achieved through utilization of this technology. In some cases, programs that used to take days to run can now be run in minutes.
- **Distributed Enterprise Scheduling** – Some jobs have to run on a regular basis and are very computational intensive. These jobs are typically run on the batch server and scheduled to run during off hours when there is less interactive load for the SAS resources. By integrating enterprise scheduling with the other grid technologies, it is now possible to ensure that customers are efficiently utilizing their SAS resources even during off hours.
- **High Availability** – By employing a load balancing switch or corporate Domain Name System (DNS), SAS Grid Manager provides a high availability platform for many SAS servers which makes it simple to configure a primary and failover metadata server, remote services, or many other SAS servers.
- **Scalability/Maintainability** – SAS Grid Manager in UNIX environments provides unique opportunities by affording the ability to share configuration and binary directories. This sharing simplifies scalability across the grid by simplifying the addition of grid nodes. Maintainability is simplified due to the ease of adding and removing grid nodes.

In summary, SAS Grid Manager is an exciting technology that will afford SAS customers the ability to take full advantage of their SAS resources across the enterprise.

As previously mentioned, SAS Grid Manager is based on the LSF platform that provides underlying support in governing the hardware resources. By architecting queues based on needs, usage patterns, and available hardware, implementations of SAS Grid Manager can be tuned to ensure maximum hardware performance, efficiency, and utilization. For example, in situations where jobs are predominantly CPU bound, the SAS Grid Manager is easily configured to not dispatch jobs to nodes that are experiencing CPU utilizations above a certain limit. Jobs that exceed the CPU threshold will be suspended. This ability to configure the SAS Grid Manager confirms that jobs consume resources based on their priority and resource requirements. The LSF platform has many dynamic indices to choose from, so you can tune a grid for virtually any situation.

Non-grid aware processes run outside of the grid's governance, so the grid cannot marshal them. Processes running outside of the grid's governance have the potential to wreak havoc on the processes that are grid controlled. For example, if a very CPU intensive non-grid aware job is run after a high priority job has been dispatched to the same node, it is possible for the grid job to be starved of resources while the non-grid job consumes all the available CPU resources. In order to alleviate the possibility of this negative interaction, the grid must be partitioned in a way that reserves resources for the grid without resulting in a significant decrease in hardware utilization (for example, reserved resources not being utilized). Initially, grids configured in this manner will require additional monitoring and tuning, especially with regard to the partitioning and tuning of the grid. However, in the long term, these environments have the potential to improve performance, efficiency, and utilization.

ARCHITECTURE CONSISTING OF GRID AND NON-GRID AWARE PROCESSES

When partitioning a grid that consists of nodes running grid and non-grid aware processes, one viable architecture segregates the non-grid aware processes, so these processes run on a subset of nodes within the grid. This architecture consists of two sets of nodes within a cluster. One set of nodes will be configured to exclusively run grid aware processes, so the processes running on this set of machines will be under the full governance of the grid. By configuring the grid to send all "high priority" or "interactive" jobs to this set of nodes, the response time can be safeguarded by applying full grid governance. The second set of nodes will consist of both grid aware and non-grid aware processes. This set of nodes will be configured to handle lower priority, non-interactive batch type jobs and will also run non-grid aware processes. The dynamic indices will be configured with additional margins to verify that resources are in essence "reserved" for the non-grid aware processes. The following figure provides a visual conception of this architecture.

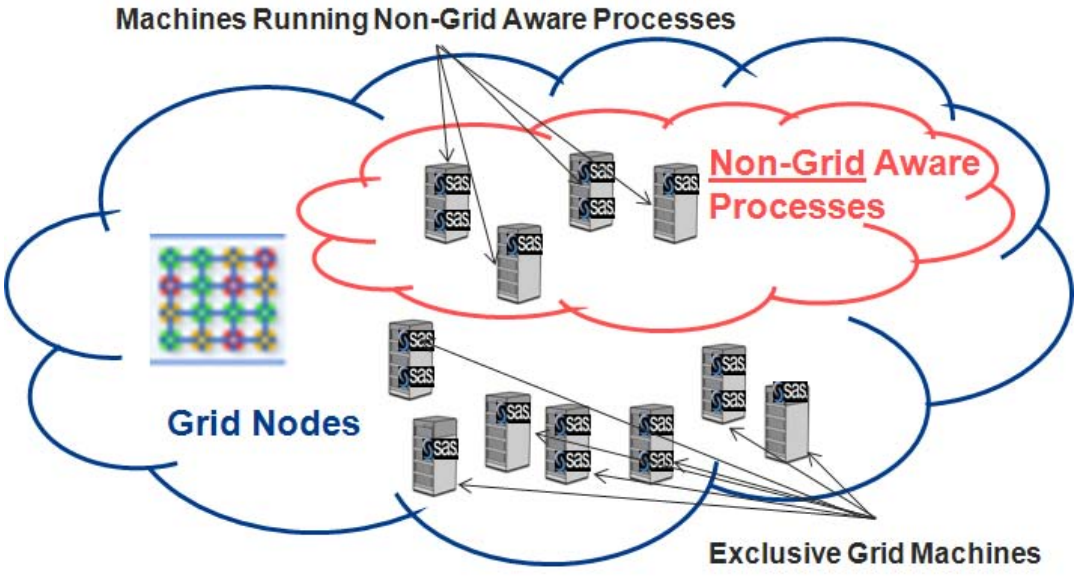

Figure 1 - Grid and Non-Grid Aware Processes

ARCHITECTURE OVERVIEW

A working grid must be configured in order to implement the architecture described in the previous figure. In order to simplify this environment, the grid will consist of a master grid controller (MC) and two additional nodes. These machines are referred to as MC, Node 2, and Node 3. It is important to keep in mind that in an enterprise environment, nodes would be added to predefined groups (which will be introduced shortly) in order to facilitate

administrative tasks. If at all possible, the grid should be architected in a manner that shares binary and configuration directories. *As always, when considering a grid architecture never underestimate the importance of a shared file system that is accessible by all machines in the cluster. SAS recommends 30-50 megabytes per second per core. This requirement is critical to ensure the performance of a grid.* Once the grid is up and running, the design should include portioned nodes by continuing the concept of two groups of grid nodes. The following high-level tasks must be performed in order to architect and implement this design concept:

1. Install LSF on all nodes, starting with the master controller.
2. Install SAS.
3. Configure the application server definitions that consist of workspace server and grid server definitions (as a minimum).
4. Define and configure LSF host groups.
5. Define and configure LSF queues.
6. Analyze, establish, and implement dynamic indices after reviewing the environment.

ARCHITECTURE DETAILS

This environment consists of two groups of nodes as depicted in the following table. Group one is referred to as "hybrid," and group two is referred to as "exclusive." The hybrid group consists of nodes that are configured to run both grid and non-grid controlled processes. The exclusive group of nodes is configured to run only grid controlled processes. These logical definitions are defined in Platform LSF which utilitizes the *HostGroup* definition in the `lsb.hosts` configuration file.

Table 1 - Groups of Nodes

Group name of nodes	Grid Aware Processes	Non-Grid Aware Processes
exclusive_h	Yes	No
hybrid_h	Yes	Yes

Metadata application server definitions are utilized to guarantee that grid-enabled jobs are sent to the appropriate group as depicted in Table 2 – Metadata Logical Definitions. The application server definitions are referred to as SASAppHybrid and SASAppExclusive. These application server definitions include a Logical Grid server which sends jobs to their respective queues, referred to as the hybrid_q and the exclusive_q. By defining these queues in the options of the logical grid server, it is possible to ensure that all grid jobs submitted to these application servers are sent to the appropriate grid queue. Once submitted to the appropriate queue, the queue definitions are configured to direct the jobs to the appropriate host group as defined in the grid.

Table 2 - Metadata Logical Definitions

Metadata Application Server Component	Metadata Application Server Component "Additional Options Setting"
SASAppHybrid	queue=hybrid_q
SASAppExclusive	queue=exclusive_q

From a SAS client perspective, aspects of the SAS Grid Manager can be leveraged to varying degrees as depicted in Table 3 – Manager Clients. The methodology utilized to reference a specific application server varies across clients. For example, in SAS programs, the application server is directly referenced as grdsvc_enable(_all_,server=SASAppExclusive) or grdsvc_enable(_all_,server=SASAppHybrid. Clients, such as SAS® Data Integration Studio and SAS® Enterprise Miner™, access the application server indirectly via the SAS

workspace server they are configured to connect through. The SAS client is the starting point for the SAS Grid Manager Job Flow.

Table 3 - SAS Grid Manager Clients

<table>
<tr><th colspan="3">SAS Grid Manager</th></tr>
<tr><th>Distributed Enterprise Scheduling</th><th>Workload Balancing</th><th>Parallelized Workload Balancing</th></tr>
<tr><td>SAS® Data Integration Studio</td><td>SAS Data Integration Studio</td><td>SAS Data Integration Studio</td></tr>
<tr><td>SAS® Web Report Studio</td><td>SAS® Enterprise Guide®*</td><td>SAS® Enterprise Miner™</td></tr>
<tr><td>SAS® Marketing Automation</td><td>SAS Workspace Server</td><td>SAS® Risk Dimensions®*</td></tr>
<tr><td>SAS® Marketing Optimization</td><td>Any SAS Program*</td><td>JMP® Genomics 4.0</td></tr>
<tr><td>Any SAS program*</td><td>SAS® Stored Processes**</td><td>SAS® Demand Forecasting for Retail 4.2</td></tr>
<tr><td colspan="2" rowspan="2">* With modifications or wrapper
** With Limitations – running the SAS® Stored Process through the workspace</td><td>Any SAS program*</td></tr>
<tr><td>SAS® Stored Processes**</td></tr>
</table>

After the job is submitted to the grid through one of the SAS clients, it is forwarded to a specific queue as defined in the grid definition for the application server. The queue defined in the Additional Options field in the Grid definition cannot be overridden by the client. The following figure is a visual depiction of how grid jobs are sent to specific groups of nodes on a grid.

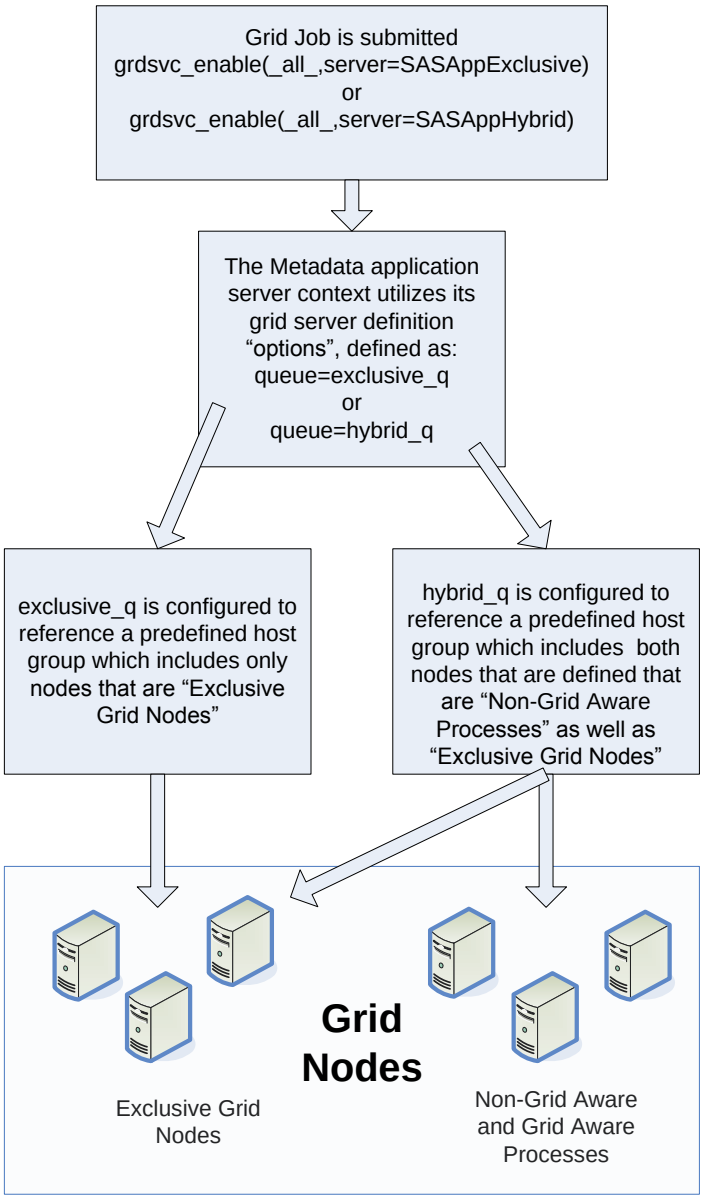

Figure 2 - High Level Grid Job Flow

NODE GOVERNANCE DISCUSSION

The architecture then necessitates the adjustment of the dynamic indices on the nodes in order to ensure that jobs are governed as efficiently as possible. One way to implement these controls is by modifying the `lsb.hosts` configuration file. The `lsb.hosts` configuration file was chosen because it supports the following configuration controls on a host-by-host basis:

1. The ability to control the total number of jobs run.
2. The ability to control the maximum run by individual users.
3. The ability to establish load conditions to control if jobs are run.
4. The ability to specify time windows in running jobs.

The configuration control used in this example is the *load_index* which allows for the specification of *loadSched/loadStop*. The first parameter, *loadSched,* defines the scheduling threshold for the load index, and the

second parameter, *loadStop,* defines the suspending threshold. If the *loadSched* threshold is exceeded, jobs are not dispatched to that particular node. Additionally, jobs are suspended on that particular node if the *loadStop* threshold is exceeded as well.

There are many dynamic load indices monitored by Platform LSF to which the *load_index* controls can be applied. See the following table for descriptions of the load indices.

Table 4 - Load Indices

Load_Index	Description	Order
io	Disk i/o (KB per second)	Increasing
It	Interactive Idle time (Minutes)	Decreasing
mem	Available memory (megabytes)	Decreasing
Pg	Paging rate (pages/second)	Increasing
r15s	15 second queue length	Increasing
r1m	1 minute queue length	Increasing
r15m	15 minute queue length	Increasing
Swp	Available swap space (megabytes)	Decreasing
Tmp	Available temp space (Disk Space in `/tmp` in megabytes)	Decreasing
Ut	CPU utilization (1 minute CPU utilization)	Increasing

It is impractical to define the "best" set of dynamic indices to apply to a given environment. Every environment is different and must be monitored, analyzed, and tuned in order to determine the best set of controls. For example, assume an environment consists of both grid-enabled and non-grid enabled processes that are CPU bound. In this environment, ut is a percentage measurement that monitors the CPU utilization and acts as an appropriate dynamic indice A ut measure of 0% indicates no load on the CPU. A ut measure of 100% indicates the CPU is operating at full capacity.

To configure the grid to utilize the ut measurement, it is essential to modify the `lsb.hosts` configuration file. First, analyze and determine the appropriate settings. As discussed previously, there are two host groups of machines. The first group (exclusive) consists of machines running exclusively grid aware processes. The second group (hybrid) consists of non-grid aware processes. The exclusive group's grid processes are monitored and controlled by the grid. The group of non-grid aware processes are monitored but not controlled by the grid (because the grid does not have the ability to control non-grid aware processes).

To set the ut on the exclusive group, you typically start with a load index setting of .8/.9 (*loadSched/loadStop).* This setting indicates that the grid will stop dispatching jobs to this node when the CPU utilization exceeds 80% and will start suspending jobs when the CPU utilization exceeds 90%. The loads must be monitored continuously during the initial setup and tuned appropriately because the process is dependent on the typical CPU load of the running processes. For example, if a typical process consumed approximately 10% of the CPU utilization, it may be appropriate to set the *loadSched* to .6 (60%) in order to preclude a single job dispatch resulting in a job suspension.

The ut setting on the hybrid group of nodes running non-grid aware processes requires additional consideration. First, an assessment must be performed to develop an understanding of the CPU load for the non-grid aware processes on individual nodes. For example, one node may be running an Oracle server, jvm (java virtual machine), or any other process outside of the SAS grid. It is also helpful to review the timeline of the CPU resource demands and to adjust the CPU utilization expectations accordingly. For example, the Oracle server may have a very high demand from 7PM to 7 AM and very little CPU utilization from 7AM to 7 PM. Without an understanding of the resource utilization patterns across time, it is difficult to tune non-grid aware hosts. SAS Grid Manager provides Report, Track, Monitor RTM, a tool utilized to facilitate the load analysis, among other things. (This paper will expand on RTM later).

In this example, assumptions are necessary for illustrative purposes. First, assume that each node running non-grid aware processes runs a single non-grid aware process that typically consumes no more than 40% of the CPU

utilization. (Although on occasion, it is possible for this process to exceed 40 %.) This CPU load varies intermittently and runs 24 hours a day. The grid architecture must ensure that this node provides the 40% CPU resource overhead associated with these non-grid aware nodes (which must be compensated for). Initially, behave conservatively and set the *loadSched* to 50%, so grid jobs are not submitted to the hybrid group of nodes when the load exceeds 50%. Setting the *loadStop* is more involved as it is necessary to refrain from starving the non-grid process of resources, as well as to ensure resources are not underutilized. In this case, it is known that the non-grid aware process consumes 40% of CPU utilization, and it also known that grid jobs will not be dispatched to the node when the CPU utilization exceeds 50%. Setting loadStop to 100% provides ample resources for the non-grid aware process and also provides some headroom for the process dispatched earlier. Therefore, set the ut of the hybrid hosts at .5/1.

ARCHITECTURE IMPLEMENTATION DETAILS

1. Update the Grid Server properties of each application server to submit jobs to the desired queue in LSF. After the desired application server is defined in the metadata, open SAS® Management Console, and drill down into the Logical Grid Server to the Grid Server. Right-click and select Properties. In the SASApp – Grid Server Properties dialog box, select the Options tab. Update the Additional Options text box as depicted in the following figure.

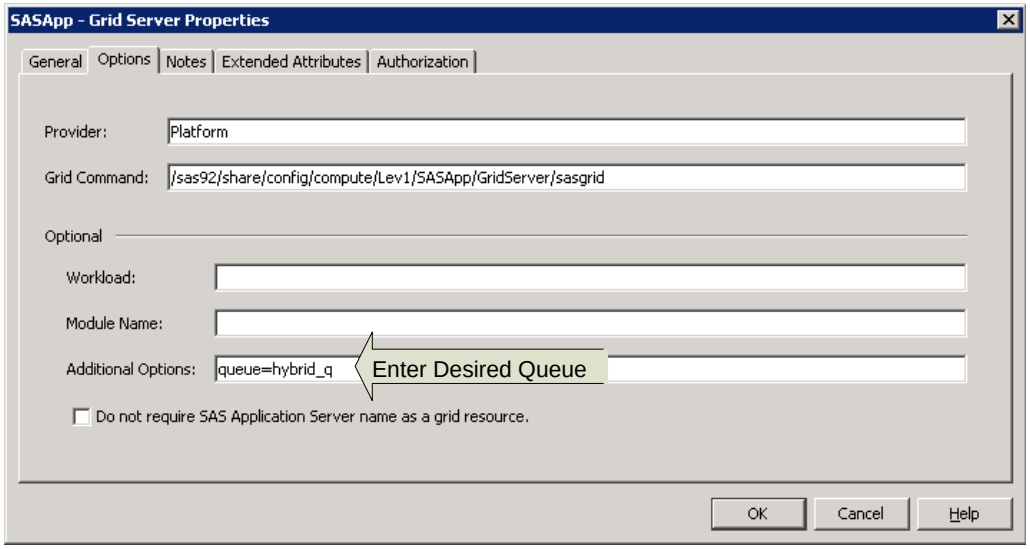

Figure 3 - Grid Server Properties

2. Definition of the host groups and hosts in the LSF configuration files involves modifications to the `lsb.hosts` file. Modification is a two-step process. First, add the host groups by using the machines introduced previously (exclusive_h and hybrid_h). Secondly, incorporate the dynamic indices controls that were developed in the previous section.

   a. To add Host Groups, open the `lsb.hosts` file for editing, and add the following code. Then, save the file.

```
Begin HostGroup
GROUP_NAME          GROUP_MEMBER        #GROUP_ADMIN  # Key words
#hgroup1            (hostA hostD)       #()     #Define a host group
exclusive_h         (node2)
hybrid_h            (node3)
End HostGroup
```

   b. To add the dynamic indices to the individual hosts, Open the `lsb.hosts` file for editing and add the following code. In this case, the ut for the exclusive host is set at .8/.9 and for the hybrid host at .5/1. Then, save the file:

```
Begin Host
HOST_NAME     tmp  DISPATCH_WINDOW   MXJ      pg     r1m    ls     ut
default        ()       ()            5        ()      ()    ()     ()
node2          ()       ()            !        ()      ()    ()   .8/.9
node3          ()       ()            !        ()      ()    ()   .5/1
```

```
End Host
```

3. Lastly, configure the queues to submit jobs only to the appropriate host groups. Previously, it was established that all jobs submitted to the exclusive_q are sent to the exclusive_ host group, and all jobs submitted to the hybrid_q are sent to both exclusive_h and the hybrid_h. After opening the `lsb.queues` file for editing, add the following code. Then, save the file.

```
Begin_Queue
QUEUE_NAME    =      exclusive_q
PRIORITY      =      30
HOSTS         =      exclusive_h
End Queue

Begin_Queue
QUEUE_NAME    =      hybrid_q
PRIORITY      =      20
HOSTS         =      hybrid_h
End Queue
```

In this case, “PRIORITY” was listed in the exclusive _q queue definition and given a higher number relative to the hybrid_q queue definition. A higher number assignment means the greater job priority that the Platform LSF applies to that queue before dispatching it to a node (assuming all other factors like scheduling policies and resources are equal).

PLATFORM RTM

Platform RTM is a Web based graphical user interface based primarily on open source tools. RTM is an acronym which stands for Report, Track, and Monitor. RTM provides a rich graphical dashboard of a SAS Grid Manager environment, including extensive graphing capabilities, alerts, administration, and additional functionality. The following information is available through the RTM interface via the “Host Load” page (However, this is not an exhaustive list of all information available through the RTM interface.):

- Host name
- Type - host type as defined in the cluster definition)
- Status – host status
- RunQ 15 seconds (r15s) – the queue length of the effective CPU run exponentially averaged over the past 15 seconds
- RUNQ 1 minute (r1m)– the queue length of the effective CPU run exponentially averaged over the past 1 minute
- RunQ 15 minutes (r15m)– the queue length of the effective CPU run exponentially averaged over the past 15 minutes
- CPU % (ut)- CPU utilization (1 minute CPU utilization)
- Page Rate (pg) - paging rate (pages/second)
- I/O Rate (io) - disk i/o (KB per second)
- Idle Time – In UNIX environments, idle time of host. In Windows environments, the amount of time that a screen saver has been active.
- Temp Avail. (tmp) – Free space in `/tmp` (G=Gigabyte, M=Megabyte)
- Swap Avail. (swp) – Swap space available (G=Gigabyte, M=Megabyte)
- Mem Avail. (mem) – Physical memory available (G=Gigabyte, M=Megabyte)

Platform RTM is not required to implement an architecture as described in this paper. It acts as a tool which simplifies the analysis of the environment in order to facilitate architecting of appropriate load indices to apply to hosts. The information above is available via the command line by using the `lsload` command without the graphs and GUI-related functionality.

CONCLUSION

In summary, a viable solution and process was introduced to demonstrate how to develop architectures for SAS Grid Manager environments. These environments can consist of both grid aware and non-grid aware processes. This approach logically segregates the grid aware and non-grid aware processes into different groups of nodes. This division, when coupled with dynamic resource load index controls, minimizes the negative effects of non-grid enabled processes on a grid by ensuring that the grid-enabled processes perform as expected, and maximum hardware

resource utilization is achieved. Additionally, the dynamic indices that you can use to monitor hardware resources were introduced, and an example implementation utilizing the ut or CPU utilization was presented. In closing, keep in mind, the power of SAS Grid Manager extends far beyond the functionality introduced in this paper.

RECOMMENDED READING

- SAS Institute Inc. 2011. SAS Grid Computing. Available at http://support.sas.com/rnd/scalability/grid/.
- SAS Institute Inc. 2010. *Grid Computing in SAS 9.2*. 3rd ed. Cary, NC: SAS Institute Inc.
- Platform Computing Inc. 2009. Administering Platform LSF. Available at http://support.sas.com/rnd/scalability/platform/PSS5.1/lsf7.05_admin.pdf.
- Platform Computing Inc. 2009. Platform LSF Command Reference. Available at http://support.sas.com/rnd/scalability/platform/PSS5.1/lsf7.05_command_ref.pdf.
- Platform Computing Inc. 2009. Platform LSF Configuration Reference. Available at http://support.sas.com/rnd/scalability/platform/PSS5.1/lsf7.05_config_ref.pdf.

CONTACT INFORMATION

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

William Nasuti
SAS Campus Drive
SAS Institute Inc.

E-mail: bill.nasuti@sas.com