Issues in Selecting a Job Management System

By Omar Hassaine - CPRE Engineering-HPC

Sun BluePrints™ OnLine - January 2002

http://www.sun.com/blueprints

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303 USA
650 960-1300 fax 650 969-9131

Part No.: 816-3299-01
Revision 1.0, 10/22/01
Edition: January 2002

Please
Recycle

Issues in Selecting a Job Management System

The Job Management System (JMS) is one of the key software components needed to administer a typical compute site. Evaluating, selecting, and deploying a JMS has always been a daunting task for system administrators and data center managers. This article addresses this critical issue by giving the reader insight into how to evaluate and compare Job Management System products.

"*Whose* job you run in addition to *when* and *where* it is run, may be as important as *how many* jobs you run!" This expression typically defines the function of a Job Management System at large compute sites. No matter how fast computing power grows, it is inevitable that there is not enough to satisfy the user community which keeps always asking for more. The High Performance Computing (HPC) segment of the computing marketplace has long recognized this challenge and fostered a class of tools that promotes and effectively optimizes the sharing of resources across clusters of systems. These tools are frequently referred to as Resource Management Software, Queuing Systems, or Job Management System, which is the preferred name used throughout this article.

The reader is first introduced to the most important JMS features and requirements followed by a brief architectural overview of each JMS product. The article then presents a comparison of the three most popular JMSs available on Sun platforms and concludes with general recommendations on how to select the most appropriate JMS in a typical compute site. This article assumes that the reader is somewhat familiar with a particular JMS. The novice reader is strongly urged to consult the references in the bibliography when appropriate to be able to follow the contents of this document.

JMS Feature Classification

This section gives a classification that includes the most important JMS features required by typical HPC sites. The JMS requirements mentioned in this section have been distilled and modified from an extensive list that was published in the NAS technical report[7]. The curious reader will also find a detailed description of each requirement in the referred NAS document. The features are classified into five separate categories:

1. Job Management System

- Must operate in a heterogeneous multi-computer environment.
- Should inter-operate with OS level check pointing, providing the ability for the JMS to restart a job from where it left off and not simply from the beginning.
- Must include a published API to every component of the JMS.
- Must be able to enforce resource allocations and limits.
- Must support multiple instances and versions of the JMS software to simultaneously run on the cluster.
- Should provide administrative hierarchy.
- Should be scalable.
- Must provide integration status with the Sun HPC ClusterTools™ software.
- Should make the source code available to the user community.
- Must be compliant with the POSIX 1003.2d “Batch Queuing Extensions for Portable Operating Systems” standard.

2. Resource Manager Requirements

- Must be parallel aware.
- Must support MPI- based parallel programs.
- Must support User-level check pointing/restart.
- Should provide a history log of all jobs.
- Should provide asynchronous communication between application and Job Manager with a published API.
- Must provide support for authentication/security system.
- Should provide a mechanism to allow reservations of any resource.

3. Scheduler Requirements

- Must be highly configurable.
- Must provide well known scheduling policies such as first-in first-out, shortest job first, fair sharing, etc.

- Must be separable from JMS. A site needs the ability to both modify and replace the scheduler.
- Must schedule multiple resources simultaneously.
- Must be able to change the priority, privileges, run order, and resource limits of all jobs, regardless of the job state.
- Support for coordinated scheduling.

4. Queuing System Requirements

- Must support job migration: The ability to suspend a job or part of a job, and move it to a different node or set of nodes for completion.
- Must provide for restricting access to the batch system.
- Must be fault tolerant and reliable.
- Must provide support for complete administrative functionality.
- Should provide support for jobs to be submitted from one cluster and run on another.
- Should be resilient to dynamic reconfiguration of the system.
- Must provide support for interactive-batch jobs.

5. Other Useful Capabilities

- Should provide a graphical interface to all JMS components.
- Should be able to support both hard and soft limits.
- Should support job accounting.
- Should provide wide area network support.
- Should provide Job array support.

The above feature requirements will be the basis to analyze and compare the three JMS packages in the *JMS Comparitive Analysis* section.

JMS Architectural Overview

A typical JMS is a software package that sits in a layer above the operating system. This section provides a brief introduction to the software architecture of the three JMSs discussed in this article.

Sun™ Grid Engine Architecture

The Sun™ Grid Engine package was formerly named Codine before Sun's acquisition of Gridware. It is a software package that provides a batch queuing framework for a large variety of architectures.

FIGURE 1 briefly describes the software architecture of the Sun Grid Engine JMS. The master daemon (`cod_qmaster`) and the separate scheduler (`cod_schedd`) both live on the master host. A shadow host is ready to take over if the master host fails. Every execution node in the cluster has a communication daemon (`cod_commd`) and an execution daemon (`cod_execd`).

This execution daemon creates a shepherd process whose mission is to spawn the task to be finally executed. The shepherd process will terminate upon completion of the task. The execution daemon is also used to report load information to the scheduler. The communication daemon is used for all communications between the

components of the Sun Grid Engine product. More information about the Sun Grid Engine architecture is found in the documentation set[4] and in the freely available web-based course[5].

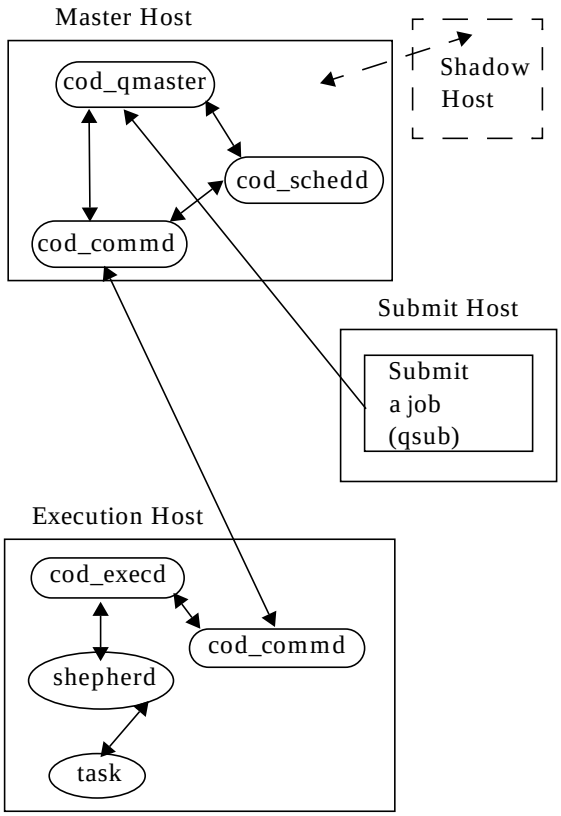

FIGURE 1 Sun Grid Engine Architecture

Platform's Load Sharing Facility

Load Sharing Facility (LSF) is the second JMS researched in this article. LSF is another batch queuing system with a rich set of features that allow system administrators to effectively optimize the use of the available resources at their computer sites.

FIGURE 2 below is a description of the LSF high level software architecture by illustrating a parallel batch job submission. LSF is also a client/server architecture based on a set of daemons that allows it to perform its functions. The master daemon (`mbatchd`) typically picks up a request and retrieves a list of suitable hosts

from the load information manager (`lim`) daemon. The slave daemon (`sbatchd`) in the first execution host inherits its task from `mbatchd`(8) and starts the parallel application manager (`pam`), a process that enables the execution of a parallel program.

`pam` starts the remote execution server (`res`) on each execution host allocated to the batch job. Finally `res` starts the tasks on each execution host. For interactive parallel submission, the user's request is directly submitted to `pam` instead of `mbatchd`. `pam` will in turn query the master `lim` for the job placement. Further information about LSF can be found in the documentation set at Platform's web site[2].

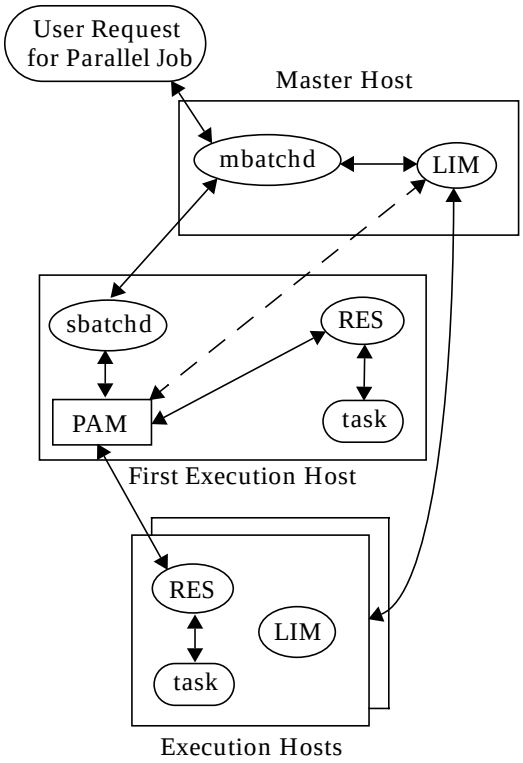

FIGURE 2 LSF Software Architecture

Portable Batch System

The Portable Batch System (PBS) is the third and last JMS that this article is covers. PBS is also a batch queuing and workload management system that operates on networked, multi-platform UNIX® environments.

The PBS high level software architecture is depicted by the diagram in FIGURE 3 below. A job request is picked up by the `pbs_server` daemon which talks to the scheduler (`pbs_sched`) to schedule the job according to the site adopted policy. The scheduler gets information about the resources from `pbs_mom` to make a final scheduling decision. The job is then dispatched from `pbs_server` to `pbs_mom` for execution on the same host or forwarded from the local `pbs_mom` to a remote `pbs_mom` for execution in another host.

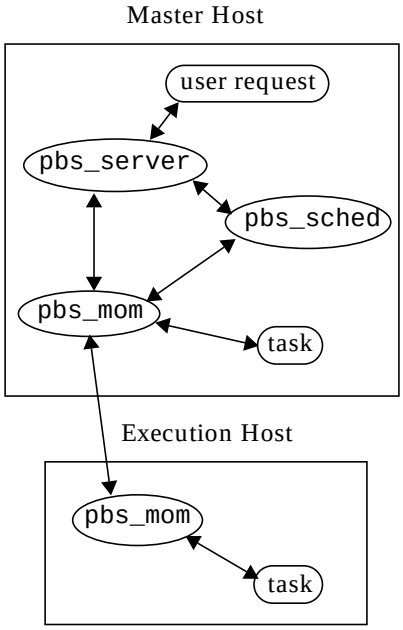

FIGURE 3 PBS Software Architecture

The interested reader is referred to the documentation[3] for further technical details about PBS.

Sun HPC ClusterTools Software

A compute site that uses a JMS and also uses parallel programming will more likely need additional software such as the Sun HPC ClusterTools software which provides the support for the Message Passing Interface (MPI). FIGURE 4 illustrates an overview of Sun HPC ClusterTools software.

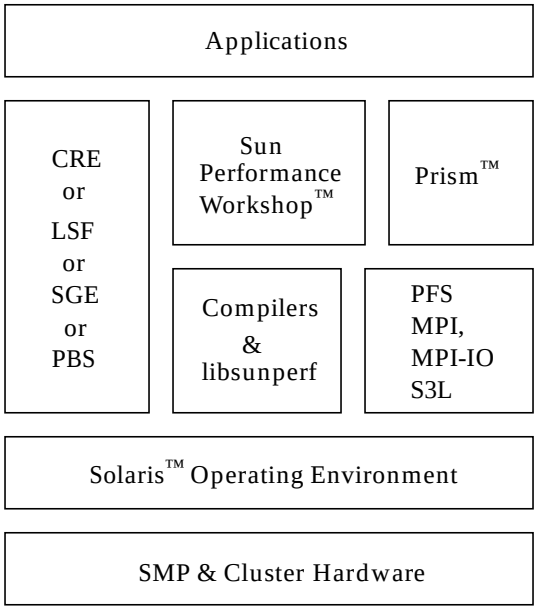

FIGURE 4 Sun HPC ClusterTools Software Overview

The Sun HPC ClusterTools[1] product consists of the following components:

- Cluster Runtime Environment (CRE)
- Parallel Filesystem (PFS)
- Message Passing Interface (MPI)
- MPI I/O
- Prism™ Programming Environment
- Scalable Scientific Subroutine library (S3L)

CRE is the critical component of this product that is briefly described in the next section due to the nature of its interaction with the JMS used.

Cluster Runtime Environment

The Cluster Runtime Environment (CRE) is an integrated component of the Sun HPC ClusterTools software. It provides the job launching and load balancing capabilities for MPI based C/C++/Fortran programs. CRE can be used in conjunction with one of the JMSs described above. FIGURE 5 is a brief software architecture diagram that illustrates a job launching scenario using CRE.

The `mprun`(1) command can be executed from any node of the cluster to launch the job that will execute `a.out` on the different nodes. The above scenario is described in detail in the Sun HPC ClusterTools Administration guide[1].

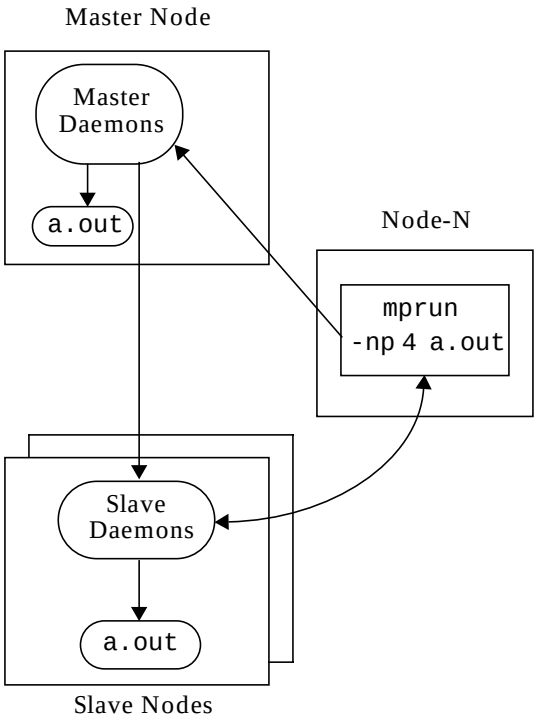

FIGURE 5 CRE Architecture

JMS Comparative Analysis

This section is devoted to comparing the three JMSs. This comparison will not stress the common features found in the three JMSs; rather, the discussion will be particularly focused on the features that differentiate them.

General JMS Features

TABLE 1 gives a brief comparison of the general features followed by a detailed explanation of how each JMS supports these features.

TABLE 1 General JMS Features Comparison

Features \ JMS	SGE	LSF	PBS
Heterogeneous platform support	Unix flavors only	Unix & NT	Unix flavors only
Multi-cluster support	No	Yes	Somewhat
System level Checkpoint restart when underlying OS supports it	Yes	Yes	Yes
User level Checkpoint restart	Yes	Yes	No
Large computational grid support	Somewhat	Somewhat	No
Massive Scalability(jobs & nodes)	Yes	Yes	Yes
Parallel job support with Sun HPC ClusterTools	Loose Integration	Tight Integration	Loose Integration
Distribution format of end product	Binary and Source	Binary Only	Source
Freely available	Yes	No	Yes
Posix 1003.2d compliance	Yes	No	Yes

Platform offers the LSF MutiCluster product that allows for the resource sharing among heterogeneous computer clusters. PBS allows for linking multiple clusters together which gives the ability to route and forward jobs between them. The

restriction is that one single scheduling policy is enforced across multiple clusters by PBS. The Sun Grid Engine product is released mainly to be used in one computer cluster due to the support issues that result when using the multi-cluster feature.

Before the checkpoint restart features are compared, it is necessary to define them in order to clear any confusion from the reader's mind. The system level checkpoint restart feature allows a machine, which crashes and later restarts, to continue at the same point without loss of data as if no failure had occurred. This feature relies on the strict support of the underlying operating system. The user level checkpoint restart, however, requires that applications link to user level libraries in order for them to be able to restart transparently following a recovery from either a system or user failure.

Be aware that the support for the checkpoint restart feature comes with limitations and restrictions. Check pointed applications that had open socket and/or pipe connections and private stack operation at the time of failure, will not be able to recover properly and need to be restarted from their initial point. The application, however, if written with significant sophistication, can allow checkpoints to be clearly made in the source code which help it recover to the last source checkpoint. The reader is referred to the corresponding documentation of each JMS to find out about the limitations of the user check pointing feature.

System-level checkpoint/restart is supported by each of the three JMS packages wherever it is supported by the operating system, such as on Cray's UNICOS or SGI's IRIX. As far as the user-level checkpoint feature, PBS is the only product that does not currently include user-level checkpoint libraries.

An advanced version of PBS is called PBS Pro[9]. PBS Pro provides support for large-scale computational grids, and an interface to the Globus Grid tool. LSF does support Grid environments through the LSF MultiCluster product. LSF is not integrated to Globus; however, Globus does have the ability to send jobs to LSF through the Globus resource manager interface (GRAM). Sun Grid Engine package is integrated with Globus, Legion, and Punch. Globus and Legion have been supported with Codine and they work as well with the Sun Grid Engine package. Punch has been integrated recently. The integration of the Sun Grid Engine package with these Grid packages is currently the Sun Grid Engine solution for multi-clustering.

The three JMSs claim that they scale up to thousands of nodes. The comparison based on this criterion requires a separate study and is beyond the scope of this article.

The three packages have parallel job support. LSF is tightly integrated with the Sun HPC ClusterTools software which allows for submitting Sun MPI jobs directly from the LSF environment. PBS and SGE, however, currently support only the external launcher mechanism of Sun MPI programs and are not yet fully integrated with the

Sun HPC ClusterTools product. More information about this issue can be found in the paper presented last year[6] and a paper that is due to be presented at the same conference[11].

As far as the distribution format of the package, PBS is currently provided as a source code distribution and LSF is distributed as binaries. The Sun Grid Engine package, however, is currently distributed both as a binary[12] and as source under the Source Code Community License[13]. The Sun Grid Engine package and PBS products are available free of charge, however LSF and PBS Pro are not.

PBS and the Sun Grid Engine package are both compliant with the POSIX 1003.2d "Batch Queuing Extensions for Portable Operating Systems" standard. However, Platform's LSF is not compliant with POSIX 1003.2d.

Scheduler

In this section, the JMS scheduler component is compared across the three covered JMS products with TABLE 2, followed by a detailed explanation.

TABLE 2 JMS Scheduler Comparison

Features \ JMS	SGE	LSF	PBS
Known job scheduling policies	Yes	Yes	Yes
Independent scheduler component	Yes	No	Yes

The three JMS schedulers have the capability to provide most of the known job scheduling policies such as First-In First-Out (FIFO), Fairshare, etc. The current version of the Sun Grid Engine package supports only the default option which implements FIFO.

The high level software architecture described earlier in this article shows that both PBS and the Sun Grid Engine package have the scheduler as a separate process. This makes the scheduler an easy candidate to be substituted and/or enhanced. The LSF scheduler is integrated with the remaining software components and this makes LSF immune from accepting a customized scheduler. Since PBS is distributed as source, it is possible to rewrite the scheduler. Similar to PBS, the Sun Grid Engine package's open source provides a framework[14] for modifying the scheduler.

Queuing System

TABLE 3 provides a comparison of the three JMSs queuing components.

TABLE 3 JMS Queuing Component Comparison

Features \ JMS	SGE	LSF	PBS
Support for interactive jobs	Yes	Yes	Yes
Job migration feature	Yes	Yes	Somewhat
Inter-cluster job launching	No	Yes	Yes
Logical queues concept	No	Yes	Yes

The three JMSs support both interactive and batch jobs. They are all highly available and can sustain system failure. They also provide administrator-configurable programs to be run by the JMS before and after a job is executed.

With respect to the job migration, this feature is intimately related to the user level checkpoint restart feature mentioned earlier. Both the Sun Grid Engine package and LSF allow jobs that have been user level check pointed to be migrated from a machine to another within the cluster. However, PBS currently does not provide this feature due to the fact that the user level check pointing feature is not yet supported by this product.

LSF and PBS allow for a job to be launched from a cluster and executed on another cluster. The Sun Grid Engine package does not provide this capability simply because it does not support multi-clustering.

The concept of queues in the Sun Grid Engine package is different from the one in its counterparts. The Sun Grid Engine package queues are defined per host whereas LSF and PBS provide logical queues that can dispatch jobs to a host or to multiple hosts. It is still possible to define logical queues in the Sun Grid Engine package using the complex feature, but it would be nice to have logical queues initially defined.

Other Capabilities

TABLE 4 gives a comparison of the miscellaneous capabilities of the three JMSs.

TABLE 4 Comparing Miscellaneous JMS Capabilities

Features \ JMS	SGE	LSF	PBS
GUI support	Yes	Yes	Somewhat
Job accounting support	Somewhat	Yes	Somewhat
SNMP support	No	Yes	No
Job Array support	Yes	Yes	No

The Graphical User Interface (GUI) for administrators and users is a useful additional feature to have in order to facilitate the administration and usage of the JMS product. LSF and the Sun Grid Engine package provide a comprehensive GUI which is targeted at administrators and users. PBS provides a user's GUI which also includes some administration-related functions available only to PBS-authorized individuals which allows them to stop/start queues, delete any job, etc. PBS also has a system monitoring tool that provides the ability to see the state of nodes in the compute cluster and to determine which jobs are running on each node.

With respect to accounting, the three JMSs provide support for logging job information in accounting files. Platform additionally offers the LSF Analyzer tool which processes the data and produces reports in user friendly format. PBS currently interfaces with the NASA site wide accounting system, ACCT++, which enables multi-system and multi-site resource accounting[10]. SGE provides the command `qacct(1)`that processes accounting data from the Sun Grid Engine product.

LSF is the only one of the three JMS offerings that provides SNMP support which allows for integrating with existing network and system management tools such as the Sun™ Management Center (Sun MC) tool. The Sun Grid Engine package, however does integrate with Sun MC software through the Tcl API. The SGE/SMC interface provides system (queue, host) and job monitoring as well as queue and job controlling facilities for the Sun Grid Engine package from within the Sun MC management console.

The Sun Grid Engine package and LSF do provide the support for job arrays which enables a single job to be rerun with different inputs. PBS does not provide this capability yet.

Recommendations

It is clear that the three JMSs described here exhibit more or less the same basic functionality. They have all matured to a respectable level that they are all currently deployed at complex computing sites and fulfill most of their requirements. It would not be wise in general to recommend a JMS product over another without going through the burdensome selection task that depends heavily on how each product meets the JMS requirements set by compute sites. This article will however recommend a process that may be used as a guideline for site administrators to select the most appropriate JMS for their particular site. The process in question contains the following three main phases:

1. **Produce a list of JMS requirements specific for the site.**

   The “Feature Classification” section in this article is a good starting point for a JMS requirement list that needs to be first produced by the group in charge of the JMS selection project at a typical compute site. The reader who needs a thorough requirement list is urged to consult the NAS technical document[7] which contains a more explanatory and comprehensive list than the one outlined in this article. We can never predict far in the future because computing needs are a moving target at compute sites and a JMS selected today may not be appropriate later, so this requirement needs to be specially taken into account when selecting a JMS.

2. **Match the requirements against the features provided by the candidate JMS.**

   This stage requires that the functionality of each JMS candidate be scrutinized as to how it is really supported. The level of support of each feature can be translated into a score whose range can be determined by how tricky each feature is supported. The JMS comparison section in this article can be very helpful in this evaluation stage particularly for the candidate JMSs that are particularly covered in this article. The tables used in the comparison section can be modified to use a scoring model which reflects the level of support of each feature by each JMS candidate. This will definitely expedite this phase in order to determine the JMS finalist(s) that would be considered for further consideration.

3. **Evaluate the demo version of the selected JMS candidate(s).**

   This last stage is very critical because the selected JMS(s) need to be tested in the environment in which it will eventually be deployed. Once the acceptance criteria have been met during the deployment phase, the decision should be easy to make.

Summary

The overall task of selecting a JMS should not be taken lightly by compute site administrators and managers as it requires a rigorous study that may take weeks if not months to reach a satisfying decision. This article has attempted to educate the reader about how to evaluate and compare the three most popular JMSs that are available on the Sun HPC platform.

The first part of this article outlined the classification of the most important JMS features and requirements. The second portion of this article was devoted to a brief architectural description of the three JMSs and the Cluster Runtime Environment. Finally, the last part portion of this article compared the three JMSs and offered useful recommendations on how to select the most appropriate JMS on a Sun platform.

References

1. Sun HPC ClusterTools 3.1 documentation set, `http://docs.sun.com`
2. LSF documentation set, `http://www.platform.com`
3. PBS documentation set, `http://www.openpbs.org`
4. Sun Grid Engine Manual, `http://www.sun.com/gridware`
5. Sun Grid Engine Web Based Training, `http://www.sun.com/software/gridware/support.html`
6. Burks, Byun, Chansup, and Duncan, *A Comparison of Job Management Systems in Supporting Sun HPC ClusterTools*, SUPerG Fall 2000, `http://www.sun.com/datacenter/superg`
7. NAS Requirements Checklist for Job Queuing/Scheduling Software, `http://www.nas.nasa.gov/Research/Reports/Techreports/1996/nas-96-003-abstract.html`
8. Jones, James Patton and Brickell, Cristy, *Second Evaluation of Job Queuing/Scheduling Software: Phase 1 report*, `http://www.nas.nasa.gov/Pubs/TechReports/NASreports/NAS-97-013/`
9. PBS Pro Web site, `http://www.pbspro.com`

10. NAS System Accounting Web site,
http://science.nas.nasa.gov/Software/Accounting/

11. Byun, Chansup, *Loose Integration of Sun HPC ClusterTools with Sun Grid Engine* SUPerG Spring 2001,
http://www.sun.com/datacenter/superg

12. Sun Grid Engine software,
http://www.sun.com/gridware

13. Grid Engine Web site,
http://gridengine.sunsource.net/

14. Grid Engine Web site, *Scheduling Framework* white paper
http://gridengine.sunsource.net/unbranded-source/browse/~checkout~/gridengine/source/daemons/schedd/schedd.html?content-type=text/html

Author's Bio: Omar Hassaine

Omar Hassaine is a senior HPC engineer in the CPRE-HPC division. Before joining CPRE, he was a technical project leader in the system software group that was involved in two consecutive high-end SPARC servers including the E10K(Starfire™). Before Omar joined Cray which was later acquired by Sun, he was a compiler engineer in the area of source code optimizers for supercompilers at Kuck & Associates. Omar has authored and given several technical presentations at various Sun sponsored events. He helped develop and teach the HPC administration & programming courses. Omar has also designed and co-developed a diagnosis tool that is already being deployed at HPC customer sites.