



AberdeenGroup

Sun's Grid Computing
Solutions Outdistance
the Competition

An Executive White Paper

May 2002

Aberdeen Group, Inc.
One Boston Place
Boston, Massachusetts 02108 USA
Telephone: 617 723 7890
Fax: 617 723 7897
www.aberdeen.com

Sun's Grid Computing Solutions Outdistance the Competition

Executive Summary

Sun Grid Engine (SGE), SGE Enterprise Edition (SGEEE), and Platform Computing's Load Sharing Facility (LSF) are three distributed resource management software solutions. The source codes for SGE and SGEEE are available in the Grid Engine open source project sponsored by Sun Microsystems and hosted by CollabNet. The Grid Engine open source project was launched in June 2001. Reference releases are available via downloads from the Grid Engine open source project (www.gridengine.sunsource.net).

The downloads are 100% compatible with the Sun products, SGE and SGEEE, apart from a different binary code license. SGE is free and available only through downloads from www.sun.com/gridware. SGEEE is available from Sun and its resellers only on CD at a cost that is dependent on grid size.

SGE and LSF provide comparable functionality and are suitable for what Sun refers to as cluster grids (one-project, one-department grids). SGEEE provides distributed resource management in more sophisticated grids — i.e., enterprise grids (multiple projects or departments, one organization) — and provides the basic functionality for global grids (i.e., collections of enterprise grids that cross organization boundaries). SGE also provides basic functionality for global grids, but not as much as SGEEE.

Moving from single departmental grids to more sophisticated grids requires human cooperation. SGEEE implements the concept of policies for specifying how humans will cooperate in multidepartment, multiproject grids. That is, policies define how computer resources are distributed among projects and people. For example, some users may not want to share their CPUs (central processing units) on certain days of the week or with specific groups. As a result, policies are implemented at a level above the distributed resource management layer.

SGEEE's policy module provides the following four benefits, which neither SGE nor LSF provide:

1. It introduces new utility computing-like parameters for scheduling. Scheduling jobs is not based on priority alone. With SGEEE, a user, team, department, or project can receive a resource allocation, for a period of time, based on some percent of the total resources available. SGEEE will ensure that the assigned percentage of resources is available to the jobs within that project or for a user, team, or department.
2. It enables collaboration. Users and project teams can negotiate resource assignments that can vary from week to week. For example, a project team may get 10% of the resources this week and 30% of the resources next week. This type of negotiation allows project teams to better manage the start and completion of their projects and allows a project team

with a “hot” project to negotiate sufficient resources to ensure that the project can finish on time.

3. It offers management alignment of compute power (cumulative usage over time) to specific projects that have high importance.
4. The more you pay the more you get. Cumulative resource usage can be proportional to budget contributions.

Sun and Platform Computing have taken different approaches to developing grid computing solutions. Sun delivers grid computing solutions with necessary functionality integrated within a single product such as SGE/EE; whereas, Platform's approach is to provide a base product (LSF Base combined with LSF Batch) and then surround it with other products to complement the functionality in the base offering. Even with the purchase of additional Platform products such as LSF MultiCluster and LSF Parallel (beyond LSF Base and LSF Batch), LSF cannot provide the functionality available in SGEEE.

In this *Executive White Paper*, Aberdeen analyzes SGE, SGEEE, and LSF and draws the following conclusions:

- SGE and LSF provide comparable functionality and provide basic resource management for cluster grids.
- SGEEE is the most comprehensive, cost-effective grid computing solution currently on the market, one that is capable of providing resource management for enterprise grids and aspects of global grids.
- Sun's most serious competitors in grid computing — HP (Hewlett-Packard) and IBM — are relying heavily on third-party proprietary products from companies such as Platform Computing versus building their own. It is Aberdeen's perspective that Sun, by leveraging the contributions of the Grid Engine open source project, is able to provide superior functionality at a lower cost.
- While Sun is ahead of its competition, building its own grid products, it is incorporating concepts developed by the leading research organizations (of which Sun is a primary participant) in grid computing — the open source Globus project (www.globus.org), the Global Grid Forum (GGF), and the Distributed Resource Management Application API (DRMAA) working group (which Sun initiated with Intel and Veridian) within GGF.

What Is Grid Computing?

Today, computing is often concerned with collaboration, sharing of data files and databases, and other means of interaction across departments within an organization and across organizations. These requirements have led to new ways of performing application development and deployment. The requirements have been,

for the past few years, primarily the concern of developers of distributed systems for scientific and technical research. Work within this community has led to the formation of grid computing technologies.

In general, the specific problem that grid computing tries to solve is coordinated resource sharing in dynamic, multi-institutional virtual organizations (VOs). Examples of VOs include service providers, manufacturers, and organizations involved in collaborative problem solving. This description of the problem that grid computing tries to solve aligns perfectly with Sun's notion of global grids, which span organizations. However, cluster grids and enterprise grids are also real examples of grid computing. At Sun, grid computing is defined as the pooling of resources into virtual systems. Sun introduces scaling with cluster grids, enterprise grids, and global grids.

Sharing is concerned with direct access to computers, software, data, and other resources, i.e., the type of sharing required for collaborative problem solving. Sharing is highly controlled, with resource providers and consumers defining what is shared, who is allowed to share, and the conditions under which sharing occurs. Sharing relationships must be flexible for sophisticated and precise levels of control of how shared resources are used. On the other hand, the relationships must be definable so that they can be implemented and enforced.

Resource sharing policies are conditional statements. That is, a resource owner makes resources available subject to constraints on when they can be used, how they can be used, and for what they can be used. The implementation of constraints requires mechanisms for expressing policies, for establishing the identity of a consumer or resource, and for determining whether or not the use of a resource is consistent with the specified sharing relationships. Any policy mechanism must be capable of handling relationships that can vary dynamically over time, in terms of the resources involved, the nature of the accesses permitted, and the participants to whom access is permitted. In addition, a new participant (individual, group, or organization) must be able to "discover" the nature of relationships that exist at any given time. For instance, a new participant must be able to determine what resources are available for it to access, the quality of the resources, and the policies that govern access. A single resource may be shared in several ways, possibly in different ways for each participant.

Today, distributed computing technologies do not address many of the concerns listed for sharing resources. The Internet, for example, addresses information interchange among computers, but it does not address the types of flexible policies required for sophisticated resource sharing at various computer sites by individuals, groups, and organizations. That is where grid computing becomes important. In the past few years, researchers within the grid community have produced protocols, services, and tools that address the challenges of resource sharing. These

technologies include security solutions, resource management protocols, and policies that support secure remote access to computing and data resources, etc.

Members of the Globus Project (www.globus.org) have published an Open Grid Services Architecture (OGSA) proposal. The OGSA proposal outlines interfaces to grid computing software that comply with Web services standards. The objective is to take advantage of Web services properties such as service description and discovery. If this proposal is adopted, common grid services like job scheduling, authentication, failure detection, and migration of data will all be accessed through standard Web services architecture. Web services are a natural fit with the underlying services for grid computing because Web services aim to connect applications across large heterogeneous networks using Web-related standards like Web Services Definition Language (WSDL) and eXtensible Markup Language/Simple Object Access Protocol (XML/SOAP).

While grid computing originated in the research world and few, if any, commercial grids are in use today, it is Aberdeen's perspective that grid computing will also become an important way to deploy commercial applications including e-Business applications. The jump to the commercial world is not difficult because groups work in teams and share resources just like scientific researchers do, and they want the same three things:

1. Efficient use of hardware and workload balancing;
2. Quality of service — which nodes are working, which nodes are not working, route around overloaded nodes, etc.; and
3. Flexibility — virtual access to resources.

Grids are a potential solution to an organization's lack of compute power and to a more efficient use of existing resources. In many companies, desktop utilization is relatively low, on the order of 20% or less, and there is often a duplication of resources across groups. Grids, via distributed resource management software, aggregate available compute resources and deliver compute power as a network service. Grids increase productivity by matching workload and resources. And, perhaps more importantly, grids make it as easy to use many CPUs as to use one, allowing the user to be much more productive.

Software Requirements for Grid Computing

Grid computing requires a collection of software features that contribute to managing the resources in heterogeneous, distributed computing environments. It is Aberdeen's perspective that sophisticated grid computing solutions must address the following:

- *Coordinated resource management* — managing the use of shared resources to best achieve an enterprise's goals such as productivity, timeliness, level of service, etc.
- *Dynamic policy mechanisms* — mechanisms capable of handling relationships that can vary dynamically over time, in terms of the resources involved.
- *Discovery* — participants must be able to determine what resources are available for access, the quality of resources, etc.
- *Dynamic scheduling* — the capability to remove resources from a low priority job and give them to a higher priority job so the higher priority job can complete execution. This is done without killing the lower priority job. If no other higher priority jobs are remaining, then the resources can be given to lower priority jobs to continue execution.
- *Controlled sharing of resources* — enforce policies that constrain access according to group membership, ability to pay, etc.
- *High-level policy administration* — distribution of computer resources among projects and people versus the lower level concept of distributing resources among jobs.
- *Security* — security services define standard functions for identifying individuals in communicating parties, encrypting messages, and so forth.
- *Heterogeneous, distributed computing environments* — grids that span projects and organizations almost always utilize platforms from several suppliers.
- *Checkpointing capability* — ability to move jobs from host to host during execution without restarting.
- *Open standards* — standards-based solutions facilitate extensibility, interoperability, portability, and code sharing.
- *Adjust to various customer attitudes toward grid computing* — architectures should be flexible enough to satisfy distributed applications and satisfy user application requirements versus the grid architecture's forcing a structure on the applications.
- *Differences with respect to high-performance computing (HPC)* — grid computing and high-performance computing are intertwined, but they are not the same. Some HPC applications may suffer using some grid computing approaches, e.g., bandwidth- and latency-dependent applications deployed across multicluster grids or grids spanning distances.

Market Potential for Grids

Grids have the potential to impact the marketplace in a number of ways by enabling more complex simulations and modeling efforts, expanding and sharing access to databases, and speeding communication between collaborators working on common projects.

Today, grids are in the very early adopter stage with the primary use coming from research, engineering, science, and areas such as Electronic Design Automation (EDA), life sciences, and others. The primary drivers are resource optimization, access to resources, cost sharing, and improved models for managing resources.

Sun's SGE and SGEEE

Sun has two grid computing solutions — SGE 5.3 and SGEEE 5.3. SGE is suitable for cluster grids, and SGEEE is capable of handling enterprise grids and some aspects of global grids. When Sun announced SGE in September 2000, the product already had a five-year history. Sun acquired Gridware, developer of the initial product, in July 2000.

The front-end development for both SGE and SGEEE is done in the Grid Engine open source project (www.gridengine.sunsource.net). Sun does not deviate from the source code produced via the Grid Engine project for releases of SGE/EE. Reference releases, which are functionally identical to SGE and SGEEE at a point in time, are available via the Grid Engine project. SGE and SGEEE are both made from the same source tree in the Grid Engine project and share internal components. When Sun decides to release a new version of SGE and SGEEE, it brings a stable build of Grid Engine software into the Sun quality assurance process and documents and productizes the software under the SGE/EE brands.

For those users who download SGE from www.sun.com/gridware and who buy SGEEE, Sun delivers controlled upgrades via patches (through support services contracts). Sun does not provide support contracts on the binaries available in the Grid Engine open source project. But most enterprise customers want a supported version and are willing to pay for support. Anyone — including Sun's competitors — can use the Grid Engine open source project code to make their own version of Grid Engine.

Product Comparisons

Functionality and feature comparisons for SGE, SGEEE, and LSF are presented in the following sections, and Appendix A contains detailed overviews of the three grid product offerings.

Comparison of SGE 5.3 and LSF 4.2

Both SGE and LSF provide functionality for what Aberdeen considers to be basic resource management — both solutions are suitable for supporting cluster grids,

but not enterprise or global grids. One of the major differences between SGE and LSF is that SGE is free and LSF Base plus LSF Batch costs \$995 per CPU for Unix and \$399 per CPU for Linux. Support for SGE is available from Sun via the Web and via a software-only support contract with Sun Enterprise Services. Support is also available from Sun partners and resellers.

Another difference between SGE and LSF is that SGE is a solution with all the functionality required to provide resource management for cluster grids integrated within a single product offering. Platform Computing's approach is to provide a base product (LSF Base with LSF Batch) and surround it with other offerings — LSF MultiCluster, LSF Parallel, LSF JobScheduler, LSF Make, and LSF Analyzer — to complement the functionality in the base offering.

LSF is built in layers. For this reason, LSF is composed of several individual products. The base system (LSF Base) provides a basic level of services that allow users to perform dynamic load sharing and distributed processing. However, if sophisticated job scheduling and resource allocation policies are necessary — as they frequently are in grid computing — more complex scheduling must be built on top of LSF Base using LSF Batch. (LSF Batch is a batch system built on top of LSF Base to provide distributed batch job scheduling services.) That is the reason LSF Base and LSF Batch are both necessary to provide a base product for grid computing. The disadvantage of Platform's approach is that the user has to pay additional costs for each of the separate offerings. A description of the additional LSF products can be found in Appendix A.

SGE and LSF both provide the following functionality:

- *Batch processing* — submit jobs and process them as soon as the resources are available;
- *Dynamic allocation of resources* — allocate resources as they are required and release them when not needed;
- *Fault tolerance* — automatically resubmit jobs that fail;
- *Failover capability* — grid continues to operate if one or more hosts fail;
- *User-specifiable resources* — at submission time, a user can specify resources needed to complete a job;
- *Resource location independence* — the user does not know or care where the compute resources are located in the grid;
- *Job status* — users want to know what is happening to their job at any given time;
- *Host status* — system administrators need to know the utilization and up/down status of all hosts in a grid;

- *Centralized management* — system administrators need the capability to manage an entire grid from one application, which is not the same as having a single machine from which to manage a grid; and
- *Suspend/resume jobs* — the user (and system administrator) has the capability to halt a job and restart it later without losing the work already completed.

Table 1 provides a comparison matrix of important features for SGE 5.3 and LSF 4.2.

The concept of queues in SGE and LSF requires some explanation because they affect scheduling in the two product offerings. When a job enters an LSF queue, it must remain in the queue until its execution is completed, unless a system administrator manually moves it to another queue. LSF queues are distinct from hosts. All LSF jobs in a high-priority queue run before any jobs in a lower priority queue are started.

SGE queues are an expression of what resources are available on each machine in a grid. When a job is submitted to an SGE-based system, the SGE scheduler takes into account the order in which the job was submitted, what queues (hosts) are available, and the priority of the job. The scheduler places all jobs in a single pending list and continuously re-evaluates the availability of resources and priority of all jobs in the grid. If a host has the resources available, SGE will automatically move the highest priority job to a queue on that host to begin execution immediately. That is an advantage for SGE (and SGEEE) with respect to utilization of resources and throughput.

An analogy for LSF queues is a grocery store where customers select a queue and wait in line to be checked out. An analogy for SGE queues is a hospital emergency room where selection of those served next is made on a number of parameters, with the list frequently re-sorted.

SGE Versus SGEEE

SGEEE contains all of the functionality and features that SGE has plus the important concept of policies that permit it to adapt to general grid computing models such as enterprise and global grids. SGEEE is substantially different than SGE (and LSF or other similar competitive products). The primary difference is SGEEE's policy module.

Any SGE grid can be upgraded to SGEEE by upgrading the master host in the grid. SGEEE requires that the master daemon run on a Solaris (2.6 to 9.0)/SPARC master host; whereas, the master daemon in an SGE grid can run on Solaris (2.6 to 9.0)/SPARC, Solaris (2.6 to 8.0)/x86, Linux/SPARC, or Linux/Intel. Master daemons in both SGE and SGEEE are 100% compatible to execute daemons from Grid Engine open source project builds such as AIX, HP-UX, IRIX, Linux, and others.

SGEEE Versus LSF

Aberdeen does not view LSF as a competitor to SGEEE because LSF does not provide the policies required to manage resources in enterprise grids. LSF is priced at \$995 per CPU for Unix. For more than 100 licenses, SGEEE is priced at approximately \$300 per CPU. Not only is SGEEE much less expensive than LSF, but it also contains significant functionality that LSF does not provide.

SGEEE Versus Aberdeen's Software Requirements for Grid Computing

When the functionality and features that SGEEE provide are compared with the list of software requirements that Aberdeen considers important for grid computing, SGEEE fares very well. For instance, it is Aberdeen's perspective that SGEEE is more than adequate for controlling and resolving resource sharing in dynamic VOs where collaborative problem solving is a must. SGEEE's policies are capable of defining what is shared, who can share, etc. And, importantly, policies defined by the use of SGEEE's policy mechanism can be implemented and controlled.

SGEEE provides for dynamic scheduling and controlled sharing of resources. It thrives in heterogeneous, distributed computing environments; is based on open standards; provides an adequate degree of security across departments and organizations; and is flexible enough to allow jobs to be automatically re-directed to new hosts when resources become available.

Table 1: Comparison of Important Features — SGE Versus LSF

Features	SGE 5.3	LSF 4.2
Cost	Free	\$995 per CPU for LSF Base and LSF Batch for Unix; \$399 per CPU for Linux
Dynamic scheduling	Yes	No
Transparent to application	Transparent to application if compatible with the underlying operating system	Transparent to application if compatible with the underlying operating system
Scripts for job submission	Scripts specific to the environment are required	Scripts specific to the environment are required
Ease of installation	Very easy to install, takes one to two minutes per host	Takes about a half-day to create a grid of 50 to 100 hosts
Ease-of-use	Easy to use (based on information collected at download time)	More difficult to use than SGE (based on download information)
Scalability	There are separate daemons for the master and scheduling functions that can run on separate CPUs in a dual processor host, allowing more jobs to be processed simultaneously	Places master and scheduler functions in one daemon, creating potential delays in scheduling

Features	SGE 5.3	LSF 4.2
Enterprise grid-like capability	Not available in SGE, but available in SGEEE	Limited form using LSF MultiCluster at an additional cost of \$200 to \$300 per node
Specify queue name at job submission	Not applicable	Name for a specific queue can be submitted with the job
Array memory for submitting large numbers of jobs at the same time	Yes	Yes
Globus support	Supported under SGE/EE	Yes
Parallel job execution	Yes	With LSF Parallel at an additional cost
Heterogeneous computing environments	Supports AIX, HP-UX, IRIX, Solaris, Tru64 UNIX, Linux	Supports AIX, HP-UX, Solaris, Tru64 UNIX, Linux, Mac OS X, Windows
Master host configurations	Each grid requires a master host. SGE master daemon must run on Solaris/SPARC, Solaris/x86, Linux/SPARC, or Linux/Intel. SGEEE master daemon must run on Solaris/SPARC	Each grid requires a master host. No platform restrictions on master host.
Job checkpointing	Yes	Yes
GUI support	Yes	Yes
Command line interface	Yes	Yes
Error logging	Yes	Yes
Required resources can be requested at job submit time	Yes	Yes
Job accounting, e.g., submit, execution times	Yes	Yes
Job arrays	Yes	Yes
SNMP agent support	No	Yes
Adding/removing hosts without shutdown	Yes	Yes

Source: Aberdeen Group, May 2002

Aberdeen Conclusions

While grid computing originated within the scientific and technical market segments, it is just as appropriate for commercial applications and any type of computing where sharing of files and databases as well as collaborative forms of interaction across projects, department, and organizations are important. With SGE 5.3 and SGEEE 5.3, Sun is able to provide the functionality required for today's grids. And, as grid computing moves more and more to the forefront, Sun, with SGEEE 5.3 and its concept of policies, is positioned to be the leading supplier of grid computing products for the future.

Sun's most serious competitors in the scientific and technical market segments where grid computing originated — HP and IBM — are trailing Sun in the development of grid product offerings. While Sun has jumped out front in the grid market with its own products — SGE 5.3 and SGEEE 5.3 — the competition is relying on third-party offerings such as Platform Computing's LSF. Aberdeen concludes that LSF 4.2 falls far short of SGEEE 5.3 in the functionality required to create sophisticated grids, and LSF 4.2 costs significantly more per CPU than does SGEEE 5.3.

Sun is positioned to respond to new grid computing requirements through its customer base and through its involvement with the open source Globus project (www.globus.org) and the GGF. Sun was the first sponsor of GGF in 1999. Sun is convinced that standards and open source are fundamental for the success of grid computing and suppliers and users. Sun was the first systems supplier to place key grid technology — the Grid Engine product suite — into open source. The Globus-based OGSA architecture is based on the same Web services standards that Sun ONE (Open Net Environment) Web Services are based on. Sun is providing input as an active member of the OGSA working group based on its experience with the large Sun ONE installed base.

Sun also initiated the DRMAA working group within the GGF. This effort promises to give commercial application vendors a “write once” interface to utilize whatever DRMAA-compliant resource management system the customer has deployed, thus reducing deployment effort for end-users. A number of manufacturers of resource management systems are working to complete this API in 2002.

APPENDIX A

Product Descriptions for SGE 5.3, SGEEE 5.3, and LSF 4.2

SGE 5.3

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
www.sun.com
(650) 960-1300

SGE 5.3

In SGE/EE terminology, an administrator is a user who is allowed to fully control SGE/EE. An operator is a user with administration privileges who is not allowed to change queue configurations. An owner is a user who is allowed to suspend jobs in queues that he or she owns or disable owned queues. A user can manipulate only the jobs that he or she owns.

SGE Overview

SGE is layered above the operating system and requires no alterations to applications. SGE can be used with any type of server — dedicated or shared, compute farms, and desktop systems. SGE is suitable for developing cluster grids.

The basic function of SGE is to match available resources in a grid with users' requests. SGE supports both batch jobs and interactive jobs. A batch job is a shell script that can be executed without user intervention, and it does not require access to a terminal. An interactive job is a session started with SGE commands that open an X-terminal window for user interaction or provide the equivalent of a remote login session.

Computational resources are delivered to user jobs by SGE based on resource policies (not to be confused with the policies for people and projects available in SGEEE) specified by the grid cluster owner organization's technical and management staff. SGE uses the policies to examine available computational resources within the grid, gathers these resources, and then allocates them to jobs in a manner that optimizes their usage across the cluster grid.

SGE Job Flow

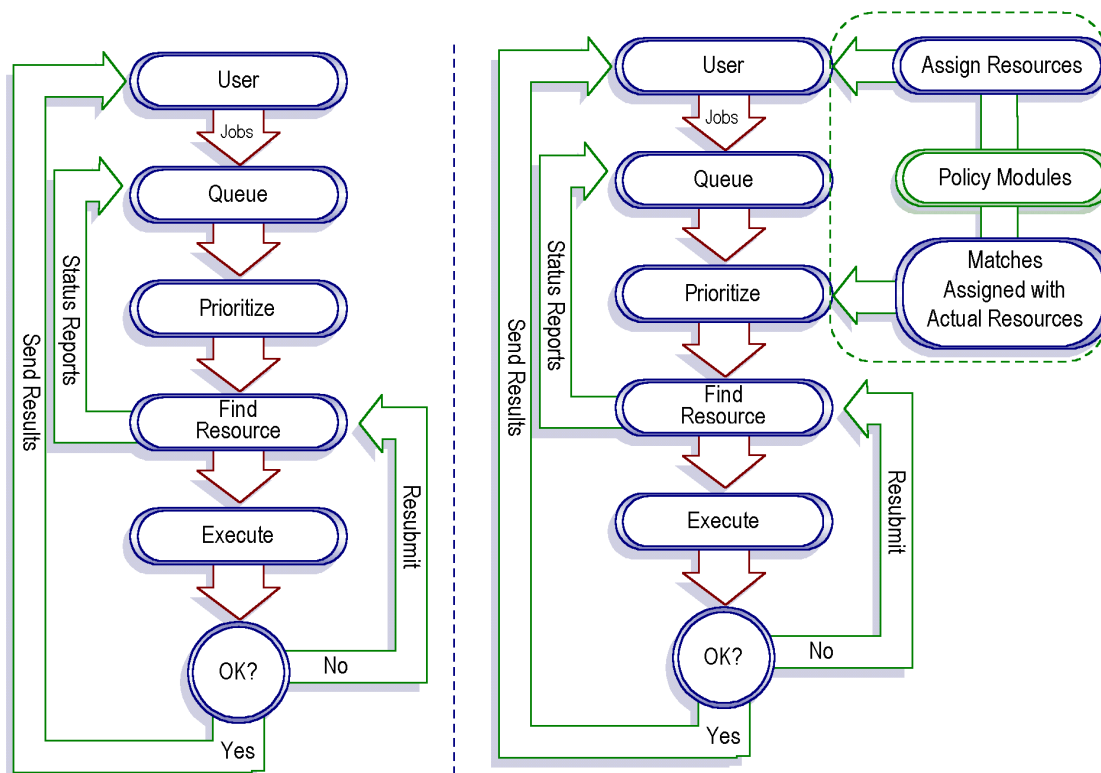
Users can submit batch jobs, interactive jobs, and parallel jobs to SGE. SGE supports checkpointing jobs — jobs that can migrate from host to host within a grid without user intervention and based on load demand on the SGE system. Checkpointing is a procedure that saves the execution status of a job into a checkpoint area for the job, permitting the job to be aborted and restarted later without loss of information and already completed work.

At a high level, SGE works in the following manner (Figure 1):

- Accepts jobs from users;
- Places jobs in a computer holding area until they can be executed;
- Sends them from the holding area to a host where they can be executed;
- Manages them during execution; and
- Logs a record of their execution when they are finished.

A user who submits a job to SGE specifies a requirement profile for the job along with user identification and a priority number. The requirements profile is a

Figure 1: Job flow for SGE (Left) and SGEEE (Right)



Source: Sun Microsystems, May 2002

statement of attributes associated with the job such as memory requirements, operating system required, etc. SGE schedules the most important jobs first. Based on the contents of the requirement profile, SGE dispatches the job to a suitable queue associated with an appropriate host server on which the job will be executed. SGE uses load balancing to spread workload among available servers.

SGE Components

SGE is composed of several components — hosts, daemons, queues, and client commands. An SGE grid cluster is composed of four types of host nodes — master, execution, administration, and submit. The master host controls all SGE activity. It runs the master daemon and the scheduler daemon. Implementing the master and scheduling functions in two separate daemons allows these two functions to run on different CPUs in the same host, thereby improving scalability. These two daemons control all queues and jobs and maintain tables about the status of queues and jobs, about user access permissions, etc. By default, the master host is also an administration host and submit host. Execution hosts are nodes that have permission to execute (and submit) SGE jobs, and they have queues associated with them.

An administrative host is a node from which administration commands may be issued. It is responsible for carrying out administrative activity for an SGE cluster grid. Submit hosts are nodes that are permitted to submit batch jobs and query their status. A node in an SGE cluster grid can belong to multiple host classes simultaneously.

The functionality of the SGE system is performed by four daemons. The master daemon maintains tables about hosts, queues, jobs, system load, and user permissions. The master daemon must run on a Linux- or Solaris-based host. The scheduler daemon maintains an up-to-date view of a grid's status. It determines which jobs are dispatched to which queues and then forwards its decisions to the master daemon, which initiates the required actions. The execution daemon is responsible for the queues associated with the host on which it runs, and it periodically forwards the status of jobs and the load on its host to the master daemon. The communication daemon communicates over a well-known TCP (Transmission Control Protocol) port. It is used for all communication among SGE components.

An SGE queue is a container for a class of jobs allowed to execute concurrently on a particular host. Throughout their lifetimes, running jobs are bound to a queue. SGE users do not submit jobs to queues. Users specify the requirement profile of a job, and SGE software dispatches the job to a suitable queue on the host with the lowest workload. The command line user interface is a set of commands that allows users to manage queues, submit and delete jobs, check job status, and suspend/enable queues and jobs.

SGEEE 5.3

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
www.sun.com
(650) 960-1300

SGEEE 5.3

The primary difference between SGE and SGEEE is that SGEEE can consolidate multiple cluster grids into enterprise grids. This capability is provided in SGEEE with the incorporation of policies. These SGEEE policies are a level above job resource allocation. Policies dictate how compute resources are distributed among projects and people, not jobs.

The administrator of an SGEEE grid defines customized policies corresponding to what is appropriate for the use of the grid. Policies are needed to provide the flexibility required for managing resources across multiple projects and across multiple organizations. Project owners using an enterprise grid have varying requirements with respect to how they manage their projects. They need capabilities to negotiate policies, flexibility for manual overrides for unique project requirements, and automatically monitoring and enforcement of policies.

Policies/Tickets

Within SGEEE, tickets are used to distribute the workload. They are assigned to projects, users, and jobs. More tickets mean higher priority and faster execution. The following are four policies available in SGEEE:

1. *Share-tree based*: This policy allocates a percentage of total compute resources to each user or project. If actual cumulative usage allocated over a period of time to each user or project exceeds the assigned value of resource allocation, then "borrowed" resources are "returned" to the other users. For example, if project A in a queue is not using all of its resources, then project B can gain access to them and use as much of A's resources as A allows. When A resumes normal activity, B must return to A a fraction of the borrowed resources. The longer B holds resources borrowed from A, the fewer resources B has to return to A. That is, if B holds some of A's resources for three weeks, then B has to return less than if B holds them for only two days.
2. *Functional*: The functional policy is similar to the share-tree-based policy, but B does not have to return resources "borrowed" from A. This policy forgets about the past.

3. *Deadline*: The deadline policy is a policy that is invoked whenever a job must be finished before or at a certain point in time and may require special treatment to achieve this. An administrator can manually give jobs extra resources to meet deadlines. The resources are relinquished after the deadline. The deadline policy is implemented by redistributing tickets — an administrator can assign a job more tickets to raise its priority so that it can meet its execution deadline.
4. *Override*: The override policy allows administrators to make resource allocation decisions manually instead of automatically by SGEEE. The override policy is implemented by giving additional tickets to jobs, users, or projects to temporarily adjust their relative importance.

SGEEE software uses these policies to examine the available computational resources within the enterprise grid. Then it gathers, allocates, and delivers these resources automatically so that highly optimized resource usage is achieved.

Figure 1 illustrates job flow for SGEEE. The job flows for SGE and SGEEE are similar in some respects; however, the availability of policies for resource management at the people and projects level imposes another level of control on top of resource management for jobs.

LSF 4.2

Platform Computing, Inc.
3760 14th Avenue
Markham, Ontario L3R 3T7
www.platform.com
(905) 948-8448

LSF 4.2

LSF is designed to be a layer on top of existing operating systems. LSF runs on most Unix systems, Linux, Windows 2000, and Cray machines. The base LSF product suite consists of LSF Base and LSF Batch. While LSF Base is a stand-alone offering, it must have LSF Batch to provide the sophisticated job scheduling and resource allocation necessary for grids. LSF Base provides the load sharing and distributed processing for the LSF solution suite. It provides services such as host selection, resource information, and transparent remote execution. LSF-based grids can span departments within an organization. AMD has a 1500-node grid that was created using LSF Base and LSF Batch.

LSF Batch is a distributed batch system built on top of LSF Base to provide batch job scheduling services to users. LSF Batch accepts user jobs and holds them in queues until suitable hosts are available. Host selection is a function of up-to-date load information stored in the load information manager (LIM). LSF Batch holds submitted jobs in a job file until conditions are right for them to be executed.

In addition to LSF Base and LSF Batch, LSF companion products include the following:

- *LSF Analyzer* — provides workload analysis across a cluster of computing resources. It generates charge-back accounting reports and removes bottlenecks and helps tune overall system performance.
- *LSF MultiCluster* — supports resource sharing among multiple LSF grids.
- *LSF Make* — dispatches tasks to multiple hosts to reduce job-processing time.
- *LSF Parallel* — manages parallel job execution.
- *LSF JobScheduler* — provides fault-tolerant, scalable, calendar-driven and event-driven scheduling across server grids.

LSF MultiCluster provides for workload sharing across grids by linking queues in distinct grids together. Each companion product has a price tag associated with it.

LSF projects a network of heterogeneous computers as a single system. LSF, like SGE and SGE4, does not require any alterations to applications. With LSF, jobs

run remotely and behave like jobs run on a local machine. Batch jobs can be run automatically as resources become available. Jobs can be suspended and resumed based on resource availability. In addition, LSF can run sequential and parallel applications and either interactive or batch jobs.

Using LSF, administrators can control access to resources such as the following:

- Who can submit jobs and which hosts they can use;
- How many jobs specific users or user groups can run simultaneously;
- Time windows during which each host can run load-shared jobs;
- Load conditions under which specific hosts can accept jobs or suspend jobs; and
- Resource limits for jobs submitted to specific queues.

LSF supports job checkpointing, permitting job migration to a better host. It supports parallel virtual machine (PVM) and message passing interface (MPI). Several scheduling policies — not to be confused with SGEEE policies — are available for managing batch. These policies include preemptive; preemptable; exclusive; first-come, first-served; and fairshare. Resources can be reserved when a job is submitted, guaranteeing that the job will always have the resources it needs while executing. Or, resources can be gained during job execution, possibly delaying a job's progress during execution because it may have to wait until resources that it needs become available.

Job Flow in LSF

In LSF, a job must be submitted to a queue. A user can name the queue when a job is submitted. When a job is submitted without a queue name, LSF examines the requirements of the job and automatically chooses a queue from a list of default queues. Automatic queue selection is based on user access restrictions (a user may not be permitted to submit jobs to some queues), host restriction (queue must be configured to send the job to all hosts in its specified list), etc.

Queues are not tied to hosts; instead, LSF provides a network-wide view of queues. Jobs can be moved from queue to queue manually. Each time LSF attempts to dispatch a job, it determines which hosts are eligible to run the job. A suitable host is one that has an acceptable load level, has the resource requirements of the job, etc. Each queue has a priority number. Jobs from the highest priority queue are started first. An LSF administrator sets queue priority when the queue is defined. Jobs are dispatched for execution by dispatching those in the highest priority queue first and then in first-come-first-served order within a queue. An administrator can change the order of jobs in a queue.

LSF is designed to be fault tolerant — that is, grid clusters continue to operate even if one or more hosts in the grid are unavailable. Each LSF grid has a master

host that is chosen dynamically. If the current dynamic host becomes unavailable, then another host automatically takes over based on the order in which hosts are listed in the cluster name file.

A job goes through a series of states during its execution phase. Most jobs enter only three states — pend (waiting in a queue for scheduling and dispatching), run (dispatched to a host and running), and done (job completed). A job remains pending until all conditions for its execution are met. Jobs can also be placed in a suspended state by their owners, an LSF administrator, someone with root access, or by LSF itself.

To provide us with your feedback on this research, please go to www.aberdeen.com/feedback.

*Aberdeen Group, Inc.
One Boston Place
Boston, Massachusetts
02108
USA*

*Telephone: 617 723 7890
Fax: 617 723 7897
www.aberdeen.com*

*© 2002 Aberdeen Group, Inc.
All rights reserved
May 2002*

Aberdeen Group is a computer and communications research and consulting organization closely monitoring enterprise-user needs, technological changes and market developments.

Based on a comprehensive analytical framework, Aberdeen provides fresh insights into the future of computing and networking and the implications for users and the industry.

Aberdeen Group performs specific projects for a select group of domestic and international clients requiring strategic and tactical advice and hard answers on how to manage computer and communications technology.