Optimizing Solaris™ Resources Through Load Balancing

By Tom Bialaski - Enterprise Engineering

Sun BluePrints™ Online - June 1999

http://www.sun.com/blueprints
Optimizing Solaris™ Resources Through Load Balancing

Sun’s workstations have become more and more powerful over the years, making it possible to handle increasingly complex engineering tasks and to complete those tasks more quickly than ever before. The engineers who use these workstations are highly skilled and represent a critical resource at the companies where they work. Consequently, the investment in state-of-the-art workstations is usually considered vital, since it enables the engineering staff to perform at peak efficiency.

Since engineers do not work 24 hours a day, there are many hours when these powerful workstations sit idly. Several techniques have been devised to take advantage of the unused CPU cycles. These techniques range from launching simple shell scripts using cron(1M), to deploying sophisticated load balancing products from commercial vendors.

In this article, I will examine one commercially available product called LSF (Load Sharing Facility) from Platform Computing Corporation. I will explain how LSF can be used as a resource management tool for running technical batch applications such as simulations.

When does it make sense to deploy LSF?

The engineering process is usually an iterative one, requiring simulations to be run many times with different data as input. If more simulations can be performed within a project’s time constraints, the quality of the engineered product increases correspondingly. For example, designing a new Integrated Circuit (IC) requires building a computer model of the IC and running a suite of functional tests against it. The more tests that can be run, the greater the chance of discovering bugs.

To determine if your engineering environment could benefit from a product such as LSF, ask yourself the following questions:

- Are there many more jobs than there are CPUs to run them on?
Do the jobs vary greatly in the length of time it takes to run them?
Are some jobs more critical to run then others?
Do different groups with different needs share the same computer resources?
Do jobs frequently fail because of insufficient resources or fail to complete in time?
Are system administrators spending lots of time writing and debugging scripts used to run batch jobs?
Are interactive users and batch jobs competing for the same set of resources?

If you answered yes to a majority of these questions, then your environment is probably a good candidate for LSF. LSF improves the throughput and turnaround time of resource intensive applications.

So how does LSF work?

To use LSF, you need to configure a cluster of computers (or hosts), each running the LSF software. One computer in the cluster is designated as the master host. The master host maintains information about all other hosts in the cluster. Batch jobs can be run on all the computers in the cluster which are designated as execution hosts. Each execution host runs an LSF process that monitors its current workload and reports back to the master host.

The following diagram illustrates a typical LSF cluster:

![LSF Cluster Diagram]

Computers in the cluster can take on multiple roles, for example acting as both submission hosts and execution hosts. They accept jobs only when they are not in use for interactive work. Since LSF itself imposes little overhead, it does not interfere with interactive work.
What resources does LSF look at?

LSF places resources into two categories: static and dynamic. Static resources describe the basic configuration of an execution host, such as the OS (e.g., Solaris 7), an attached high-speed interconnect, or a node-locked application license. These parameters are useful in determining which execution hosts meet the resource requirements of the batch jobs in the queue. For example, if a batch job requires 1GB of RAM to run (as specified in the job’s profile), LSF waits until an execution host with 1GB of RAM or greater becomes available, and then submits the job for execution on that machine.

Dynamic resources change over time as the workload on the execution host changes. Examples of dynamic resources include the current CPU utilization rate and the amount of available virtual memory. These resources are measured by an LSF process which is installed on the execution host. The load measurements are fed back from all execution hosts to the master host, which uses the information to determine what execution host is the best candidate to run the next batch job.

Sharing the Load

It would be nice if all batch jobs were of equal importance and the users who launch them were conscientious about sharing resources with other users. This would certainly make scheduling batch jobs a whole lot easier. However, in the real world, not all tasks have the same priority, and not all users are considerate of others.

Recognizing this reality, LSF provides several ways to schedule and prioritize batch jobs. This allows a system administrator great flexibility in establishing resource sharing policies. Preemptive scheduling can be used to make it possible for a high priority job to suspend a long running low priority job, and then restart that low priority job later. A fair share scheduling policy can be established by limiting the number of concurrent jobs an individual user can run. This prevents aggressive users from placing all their jobs at the head of the queue, forcing other users to wait. Exclusive scheduling is also available on a per host basis, so that only certain jobs are allowed to run on certain hosts.

Planning for High Availability

Batch jobs usually run unattended over nights and weekends. If a failure occurs, it is often not discovered until the following morning, and the failed job may not be rerun until the subsequent evening. To address this situation, LSF includes several High Availability (HA) features which can be coupled with the Sun(TM) Cluster software to implement a robust highly available environment for running batch jobs.
The LSF software, itself, is highly available on many fronts. If the current master host fails, a backup host is automatically promoted to be the new master host. If an execution host fails, it is taken off line, and no more jobs are sent to it. A failed job can be restarted automatically on a different execution host. LSF also supports applications checkpointing, and includes library which application developers can use to insert checkpoints at various locations in their code. Applications written in this manner can resume execution at the most recent checkpoint, making it unnecessary to restart them from the beginning.

Batch jobs typically require large amounts of data (for both input and output). Since the data must be available to all the execution hosts, it is usually hosted by NFS(TM) mounted volumes. To prevent an NFS server from becoming a single point of failure, you can configure a pair of servers with the high availability NFS services available with Sun Cluster Software. In this situation, if an NFS server fails, the other server automatically takes over.

Summary

A system administrator who is well versed in writing shell scripts and administering cron(1M) can easily run a small number of batch jobs on a limited number of Sun’s workstations. However, in situations where the job mix becomes more complex due to organizational consolidation or a mandate to optimize resources, a commercial workload management product like LSF may make sense. Reduced administration costs and increased resource usage can make this a wise approach.

More information on LSF and companion products can be found at Platform Computing’s Web site (http://www.platform.com).

Author’s Bio: Tom Bialaski

Tom joined Sun Microsystems in 1984 as a Systems Engineer and has been providing network computing solutions to customers since then. He is currently a PC interoperability specialist and has recently received his MCSE certification from Microsoft.