Integration of Maui Scheduler with LSF at NCSA

Michael Pflugmacher
& Michael Shapiro

Abstract

This paper describes the motivation and integration of the Maui Scheduler with Platform Computing's LSF batch queuing system at NCSA.

Introduction

NCSA is working with Maui Scheduler developers to integrate their scheduler with Platform's LSF batch system on our multiple Origin 2000 machine cluster. The talk will cover motivation for doing the integration and the basic interfaces to make the system function. It will also cover enhancements made to the scheduler to facilitate NCSA's batch system requirements and touch on a few extensions that we hope to have added.

History and Motivation

NCSA has used Platform Computing's LSF batch system for several years. We have made extensive use of elim extensions as well as a separate Job Monitor Daemon, for which a paper was presented at CUG 98 in Denver, to better control batch jobs and machines. Enhancements were made to schedule jobs on CPU, memory, and time factors. Other enhancements were integrated to allow us to implement policies that allow for dedicated resources to be available within a time window or as dedicated jobs were submitted. These features were wired together within the framework that local scripts and LSF elim extensions allowed.

We were then tasked with the ability to schedule batch jobs on NCSA's Origin machines that would start in sync with jobs at other Alliance site machines via the Globus meta-queuing system. This would require a guaranteed start time for these jobs. Our configuration of LSF was not going to allow its' preemption mechanism to be useful or politically correct in that it would suspend running jobs to make room. This also was not a scheduled event. It is an immediate priority solution. Investigation of other queuing systems was initiated.

The Maui Scheduler was found to have the major features we were looking for. It was originally written to be the job scheduler for IBM's Loadleveler on an SP2 system. It was being ported to the PBS batch system. We met with the Maui Scheduler developers and decided to integrate with our current LSF instead of a total replacement for user interface and the ability to retain many local batch system utilities that were integrated to LSF. The developers were also very willing to work with us for not only the basic integration, but to also implement additional features and functionality in a very timely manor. We have also met with LSF developers with the same requirements. While they were receptive to the needs, they were not able to make commitments or provide the features in the timeframe.

Maui Features

Several features with new or preferred functionality

- Advanced Reservations
  The ability to reserve CPUs and memory into the future for jobs that will be submitted. This includes creating reservations for accounts, users, and specific jobs.

- Job based priorities
  Priorities are assigned to each job based on quality of service desired and requested resources. The priorities can also be configured to dynamically increase as the job waits in the queue.

- Schedule CPU and Memory resource without lockout
  The scheduler makes a reservation for the highest priority job. All jobs can be configured to increase their priority as a function of queue wait time, thus eliminating job lockout when the job reaches a high enough priority.

- Backfill/reservations based on job requirements
  Backfill allows machine resources to be utilized by smaller and shorter jobs while the scheduler is waiting for the resources to be available for a job with a reservation.

- Short pools
  Provide a defined amount of system resources that jobs of “short” runtime can utilize for quick turn-around.

- Simulations
  The Maui scheduler has a simulation mode that allows the administrator to make configuration changes and test the change against actual job history. This allows the effects to be measured and evaluated before users see the effects live.

Lost Functionality

Several LSF scheduler features that we lost with the Maui integration:

- Suspend/Resume based on load/Resources
  The ability for the system to reduce or resume its’ number of batch jobs based on resource usage levels.

- Run/Dispatch Windows
  LSF windows on queues cannot be supported with the current Maui scheduler. We used this feature to hold certain queues to only run on the weekends.

- Preemption
  The ability for a higher priority pending job to cause a lower priority running job to suspend free resources for the higher priority job to utilize.

Additional Features

The next phase of the Maui scheduler integration into the LSF batch system at NCSA is planned to recover lost and adds new features.

- Deep Reservations
  Enables the scheduler to set reservations for multiple jobs simultaneously. This includes features like startdate, enddate, and dedicated queue to get next available slot.

- Suspend/Resume Handling & Scheduling
  Allow higher QOS jobs to suspend lower priority jobs to minimize high priority queue time. We plan to use this to guarantee that jobs with advanced reservations will have all requested resources when there run window arrives.

- Data Pre-Stage
  Allow for the triggering of a procedure that would prepare input data for the batch job before it started on the batch machine. In NCSA's case, this would allow us to issue a stage command to the Unitree mass storage system to get all files for the job staged from tape to disk cache.

- Estimated Start Times
  Provide the ability to give an estimated start time to each queued job based the current scheduling cycle. It is acknowledged that a job of higher priority or machine availability can skew that estimate.

Integration Components

Wrapper for the LSF bsub command

The batch policies and the Maui integration have required that several of the bsub command options be enforced, modified, or rejected. This is part of the functionality of a wrapper script that front-ends LSF's bsub command.

Required Options

Requires the user to specify the number of CPUs (-n), the amount of memory (-M) and the wall clock time (-W). They are passed to Maui and to our job monitor daemon (JMD). These values are not passed to LSF. LSF handles these values differently than we desire. –n requires job slots. –M imposes a per-process memory limit rather than a per-job limit. –W is enforced by LSF and means RUN+SUSP time-- we need it to include data transfer time while the job is running.

Modified Options

Accepts the –m and –R options. Both options are passed to Maui. The –m option is also passed to LSF, but the –R is not. Both –m and –R are used only to do host selection for a pending job. The –R could interfere with resume requests.

Rejected Options

Options that influence job start times (e.g., -b, -t , and -w) are rejected. They do not mesh well with Maui scheduling and advanced reservations.

The various options are passed to Maui and JMD by setting Unix environment variables during the bsub. For example, BSUB_LIMIT_NPE is set to the number of CPUs requested.

Resource manager interface between Maui and LSF

The Maui scheduler interfaces with external queuing systems using client/server architecture. A stand-alone server called the resource manager implements four services.

(1) get host information
(2) get jobs information
(3) run a job
(4) kill a job

The LSF commands (e.g., bhosts, bqueues, bjobs, brun, bkill, etc) are used to gather most of the information needed by Maui. The job requirements are determined by reading the job scripts in the LSF *logdir/info* directory and parsing the commands that set the jobs' environment variables. The resource manager also gets data-transfer-time for running jobs from the JMD and suspend-time from LSF in order to dynamically modify a job's wall clock limit. It gets some host information via the IRIX array services (although this could have been implemented using elim/lsload).

Summary

The initial trial of the Maui scheduler integration with LSF on the NCSA Origin cluster proved to have a few problems. A couple of bugs with the code were quickly resolved. But it was also concluded that the NCSA policies with Maui's default one deep reservation scheme would allow some jobs to be locked out. It was decided that the deep reservation feature discussed above was needed to resolve the problem. The developers, as of this writing are currently adding this feature to the Maui scheduler. We plan to have the improved version of the scheduler active late June timeframe.

Acknowledgments

The Maui Scheduler was written and is supported by the Maui High Performance Computing Center. David Jackson of the Maui Center is the primary developer and interface with NCSA for this project. The LSF interface to Maui programs were written by Michael Shapiro of NCSA with overall guidance and review by myself, the High Performance Computing Systems Group, and the Computer Policy Committee.