Moab User Training

UCRL-PRES-228869

Blaise Barney
Don Lipari
February 27, 2008
Livermore Computing Training
Topics

- Background and Goals of the Tri-Lab Workload Manager
- Moab Overview, Features, Concepts and Terminology
- Moab Configuration and Livermore Computing Policies
- Resource Usage Accounting
- Using Moab – Common Tasks
- More Moab Commands
- LCRM Wrappers and Translation Guide
- Further Information
- Hands-on Exercises
Background and Goals of the Tri-Lab Workload Manager
Current State of Resource Management at LLNL

Livermore Computing Resource Management (LCRM) System
- An enterprise-wide workload manager developed, enhanced and maintained for 15 years by LLNL developers
- Roughly 150K lines of C code
- Schedules jobs over a disparate collection of machines
- Jobs are submitted using the “psub” command

Simple Linux Utility for Resource Management (SLURM)
- An open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux (and AIX) clusters started 5 years ago by LLNL developers
- Roughly 180K lines of C code
- Manages a queue and launches jobs on a single cluster
- Jobs are submitted by the workload manager or by using the “srun” command

IBM’s LoadLeveler
- Machines running LoadLeveler are either being retired or have been converted to run SLURM
A Brief Digression into the Distinction between a Resource Manager and a Workload Manager

- A resource manager manages a queue of batch jobs for a single cluster. The resource manager contains the job launch facility as well as a simple FIFO job queue. It also has hooks into the high-speed interconnect. Examples of resource managers are SLURM, LoadLeveler, and TORQUE.

- A workload manager is a scheduler that ties a number of resource managers together into one domain. This allows a job to be submitted from one machine and run on a different cluster. The workload manager also implements the policies that govern job priority (e.g., fair-share), job limits, and consolidates resource collection and accounting. Examples of workload managers are LCRM, LSF, and Moab.
Motivation for Adopting a Common Workload Manager - Goals

- To create a more uniform environment for users to submit, status, and control their batch jobs
- To move toward the day when users can submit jobs at one lab and run at another lab just as easily as they do at the users’ home labs
- To standardize the collection and reporting of computing resource usage
- To foster a collaboration among administrators from the three labs and with the vendor that serves to address common problems, share solutions, and develop a collective expertise
- To reduce duplication of efforts in training, documentation, and customer support
- To unify our procurement activities to leverage our combined business to achieve the best value at the best terms
What Stays and What Changes

- All installations of LoadLeveler are either retired or replaced by SLURM
- SLURM becomes the sole resource manager installed on Livermore Computing systems
- Open source development of SLURM continues for the foreseeable future
- LCRM is replaced by Moab - a third party, commercial product
The Tri-Lab Workload Manager Selection Committee

- Formed in the Spring of 2005
- Met periodically to identify the goals and requirements for a workload manager
- Evaluated the workload managers each lab was currently using and how well they met the needs of users
- Created and issued an RFP for a workload manager
- Evaluated the bids
- Selected Cluster Resources, Inc.’s Moab Cluster Suite in August 2006
Cluster Resources, Inc. and Moab

- CRI is a strong company led by some of the industry’s most innovative and respected leaders in computing resource management.
- Moab is a policy-based grid management solution that integrates scheduling, managing, monitoring and reporting of workloads across independent clusters.
- Moab simplifies management across diverse groups, hardware, storage, networks, licenses, resource managers and operating systems.
- Moab represents a new direction for LLNL and LANL.
- CRI agreed to enter into a collaboration with the Tri-Labs.
The Tri-Lab - CRI Collaboration Model

- Full access to Moab source code
- Direct access to CRI’s most senior code developers
- Tri-Lab has significant influence in Moab product development
- Tri-Lab tool and plug-in development is encouraged
- Tri-Lab Moab administrators share common goals and collaborate on solutions
The Tri-Lab Support Model

- Written and approved in December 2006
- Establishes the process for addressing issues that could be encountered in running Moab
- Includes definitions of problem severity and process for escalation
- Identifies the process for developing tools that serve the individual needs of the three labs
- Identifies the process for merging the requirements and concerns of the three labs into a single voice of direction to CRI
What a Workload Manager Does

- Schedules jobs on across a domain of machines
- Sets job priority that determines the order of job execution
- Provides a fair-share mechanism to insure resources are delivered commensurate to buy-in
- Enforces the established policy such as:
  - Job duration limits
  - Job size limits
  - Qualities of service (normal, standby, expedited)
- Collects and reports computing resource usage and statistics
LCRM and Moab

- LCRM evolved organically over the years to suit the needs of LC users and management “like a glove”
- Moab offers a sizeable number of configuration parameters and models of use
- Our approach was to configure Moab to provide the service and functionality that LCRM users expect
Replacing LCRM with Moab

In transitioning LC users from LCRM to Moab, considerable effort was made to perpetuate
- Existing policies
- Concepts and models of use
- Fairness
- Performance
- Easy user account setup and administration

Important goal was to provide transparency for legacy LCRM job scripts
Support for Legacy LCRM Job Scripts

- Users will be able to submit LCRM job scripts using the psub command.
- LCRM jobs scripts submitted through psub will actually be sent to Moab for scheduling.
- The PSUB_ environment variables will be set in the job’s execution environment.
- Users will be able to status their jobs using the pstat command.
- Users will also have the full set of native Moab commands available for use.
**LCRM Wrapper Scripts**

- psub
- palter
- pstat
- prm
- phold
- prel
Job Scripts

- Users who wish to submit legacy LCRM job scripts to Moab will use the psub command.
- The psub command will understand #PSUB directives (psub options) in existing LCRM job scripts.
- Users who elect to use Moab’s msub command will use msub options (prefixed by #MSUB).
- All jobs submitted to Moab, either by msub or psub will be able to be statused by pstat or Moab’s “checkjob” command.
Transitioning from LCRM to Moab

- Will be very gradual
- Maiden voyage for Moab was on atlas in March 2007
- Moab will be installed on all new LC machines
- Once users and administrators are comfortable with Moab’s operation and performance, existing LC machines will be gradually migrated to Moab
  - LC test-bed machines
  - OCF machines
  - SCF machines
Tri-Lab Uniformity

- We are working toward the day when the interface to the batch scheduler across the ASC Tri-Labs is uniform and based on Moab's native command set.

- We recognize the investment in the LCRM interface and we provide backward compatibility with LCRM legacy scripts and the lrm library.

- We help promote user efforts to transition from the LCRM model to the Moab model.

- There are features that can only be found in Moab.

- The LCRM model will be deprecated over the coming years.
Moab Overview, Features, Concepts and Terminology
Moab Grid Model
Moab Architecture
Moab Features

- Offers a large number of job specification options
- Schedules jobs over a collection of independent machines
- Provides status of running and queued jobs
- Achieves a high level of machine utilization
- Collects and reports job usage statistics
- Prioritizes jobs based on a number of factors
- Implements fair-share policy
- Establishes target shares by account
- Imposes LC policy: limits, quality of service, etc.
Moab Features LCRM Does Not Have

- showbf – Shows what resources are available for immediate use
- showstart - displays the estimated start time of a job based a number of analysis types
- Reservations – nodes can be reserved for Dedicated Application Times (DATs)
- Computing resource aware – e.g., run jobs that do not require a Lustre file system
- Jobs can request a wall clock duration range
- Moab can schedule and backfill interactive (pdebug) node partitions
- Moab Access Portal – pending evaluation
Job Specification

- **Credential**
  - User
  - Group
  - Account
  - Class (aka Queue)
  - Quality of Service (QoS)

- **Execution machine(s)**

- **Number of nodes**

- **Wall-clock Limit**

- **Requested features**

- **Request a signal**

- **Request a time to run**

- **Output and Error file locations**

- **Mail at job start and finish**
Job Accounts

- Analogous to LCRM banks
- Form the structure to the fair-share hierarchy

![Diagram of job accounts hierarchy]

- Root 100%
  - LANL 33.3%
    - Project A 50%
  - LLNL 33.3%
    - Project B 30%
  - SNL 33.3%
    - Project C 20%
### Job Class / Queue

- Moab uses the terms Class and Queue synonymously
- Correlates to a SLURM partition
- A SLURM partition is a pool of nodes on a single machine
- Jobs cannot span a SLURM partition
- Jobs that do not specify a queue will be assigned to the host’s default queue

<table>
<thead>
<tr>
<th>Queue</th>
<th>Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login</td>
<td>1-5</td>
</tr>
<tr>
<td>pdebug</td>
<td>6-50</td>
</tr>
<tr>
<td>pbatch</td>
<td>51-1000</td>
</tr>
</tbody>
</table>
Quality of Service

- A Moab job specification that correlates to the LCRM job class
  - Expedite
  - Normal
  - Standby
- Normal is the default everywhere
- The ability to run standby is present on most machines
- The ability to run expedited jobs is granted for special accounts or users
Job Priority

- Refers to the priority for ordering the queue of jobs waiting to run
- Priority ranges from \(-1,000,000,000\) to \(1,000,000,000\)
- The higher the number, the sooner the job will be scheduled to run
- Comprised of three factors
  - Fair-share
  - Technical Targets
  - Queue Wait Time
- Users always have the option to order their own queued jobs relative to each other
Fair-share

- A mechanism of setting priority based on the disparity between how many shares a user has of a machine and how much processor time their jobs have recently consumed.
- The difference between target shares and actual usage determines the fair-share component of the job’s priority.
- Usage has a half life of one week.
Notice of Remaining Time

Job determines when its time is about to expire so that it can save its data and exit gracefully

Two methods are available

Signal
- Request a signal at submission time
- Must write a signal handler
- Moab issues a signal when job has a minimum time remaining

Polling
- Job requests / checks remaining time in polling loop

The polling method is recommended
- Does not tie up a signal or require a signal handler
- Moab does not currently send a signal to standby jobs when they are earmarked for removal
Signal Method

- Job requests a signal via msub option
- Specifies the number of seconds remaining at which to send the signal
- Specifies the signal to send

- `msub -l signal=<sig_num>[@<secs_remaining>]`
- `msub -l signal=<sig_name>[@<secs_remaining>]`
Polling Methods

- Jobs periodically poll to determine wall-clock time remaining until requested duration limit is reached.
- `yogrt_remaining()` – a locally developed library that provides the best performance and accuracy
- `slurm_get_rem_time()` – only when SLURM is the underlying resource manager
- `MCCJobGetRemainingTime()` – native Moab API
Moab Configuration and Livermore Computing Policies
Expedite Quality of Service

- Given a higher scheduling priority
- Are exempt from the following limits
  - Wall-clock limit
  - Node size
  - Node * time limit
Standby Quality of Service

- The lowest scheduling priority
- Will be removed when normal or expedited jobs need the resources
- Are exempt from the following limits
  - Wall-clock limit
  - Node size
  - Node * time limit
Queues

- Identified with SLURM node partitions
- Typically include
  - `pbatch` – batch only queue
  - `pdebug` – interactive sruns allowed
- **Specified with** `msub -q <queue>[@<host>]`
Job Defaults

- Default wall-clock time limit
- Default job node count is 1
- The following limits are not enforced and there is no default applied
  - CPU time limit
  - Process size limit
Job Limits

- **Host Limits**
  - Minimum number of nodes
  - Maximum number of nodes
  - Wall-clock time
  - Node*time product
  - Maximum number of active jobs for consideration
  - Maximum number of jobs a user can run on host

- **Account (LCRM bank) Limits**
  - Maximum number of nodes
  - Wall-clock time
  - Node*time product
  - Maximum number of jobs that can be on a host
LCRM Limits No Longer Enforced

- Maximum number of jobs on any given host (QTOTLIM)
- Maximum CPU time for a host (TOOLONG)
- Maximum process size (PTOOBIG)
- Memory limits (WMEML and WMEMT)
- Limits specific to a user in a bank
Grace Time

- The remaining time a job will be granted to run after it has been earmarked for removal.

- Removal can be by:
  - Job Cancelling to prepare for DAT (grace time is typically 10 to 15 minutes).

- Jobs that requested a signal will receive that signal when earmarked for removal:
  - In such cases, the system grace time takes precedence over the signal notification time the job requested.

Note: There is currently no grace time for removing standby jobs when normal jobs are submitted.
Resource Usage Accounting
Statistics on every job are collected on atlas

The usage parameter of interest is the processor*time product

All of a job’s CPU cycles are attributed to the job – process-based accounting is no longer done

This usage feeds the fair share calculation

Other resources (memory, disk space, bandwidth, etc.) is not tracked or charged

The lrmusage database is now being populated with usage data from Moab-scheduled machines
LCRM Wrappers and Translation Guide
## Terminology Translation Table

<table>
<thead>
<tr>
<th>Simplified Definition</th>
<th>SLURM</th>
<th>LCRM</th>
<th>Moab</th>
</tr>
</thead>
<tbody>
<tr>
<td>A subset of nodes on a host</td>
<td>partition</td>
<td>pool</td>
<td>class / queue</td>
</tr>
<tr>
<td>Host scheduling group</td>
<td></td>
<td>partition</td>
<td>partition</td>
</tr>
<tr>
<td>The project to charge usage</td>
<td></td>
<td>bank</td>
<td>account</td>
</tr>
<tr>
<td>User</td>
<td>user</td>
<td>user</td>
<td>user</td>
</tr>
<tr>
<td>Unix group</td>
<td>group</td>
<td></td>
<td>group</td>
</tr>
<tr>
<td>normal, standby, expedite</td>
<td></td>
<td>job class</td>
<td>QoS</td>
</tr>
<tr>
<td>A job submitted via srun</td>
<td>interactive</td>
<td>interactive</td>
<td>non-msub</td>
</tr>
<tr>
<td>An msub job without a command script</td>
<td></td>
<td></td>
<td>interactive</td>
</tr>
</tbody>
</table>
# Job State Comparision

<table>
<thead>
<tr>
<th>Job State</th>
<th>SLURM</th>
<th>LCRM</th>
<th>Moab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comm Problem</td>
<td>PENDING</td>
<td>ELIG, WCPU,</td>
<td>NotQueued</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WPRIO</td>
<td></td>
</tr>
<tr>
<td>Queued</td>
<td></td>
<td>Idle</td>
<td></td>
</tr>
<tr>
<td>System Hold</td>
<td>HELDs</td>
<td>SystemHold</td>
<td></td>
</tr>
<tr>
<td>User Hold</td>
<td>HELDu</td>
<td>UserHold</td>
<td></td>
</tr>
<tr>
<td>Staging</td>
<td>BAT_WAIT,</td>
<td>Staging</td>
<td></td>
</tr>
<tr>
<td></td>
<td>STAGING</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Starting</td>
<td>RUN</td>
<td>Starting</td>
<td></td>
</tr>
<tr>
<td>Running</td>
<td>RUNNING</td>
<td>Running</td>
<td></td>
</tr>
<tr>
<td>Suspected</td>
<td>SUSPENDED</td>
<td>Suspended</td>
<td></td>
</tr>
<tr>
<td>Deferred</td>
<td>DEFERRED</td>
<td>Deferred</td>
<td></td>
</tr>
<tr>
<td>Depends on job</td>
<td>DEPEND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Completed</td>
<td>COMPLETED</td>
<td>Complete</td>
<td>Completed</td>
</tr>
<tr>
<td>WC Limit Reached</td>
<td>TIMEOUT</td>
<td>REMOVED</td>
<td>Removed</td>
</tr>
<tr>
<td>Cancelled</td>
<td>CANCELLED</td>
<td>REMOVED</td>
<td>Removed</td>
</tr>
<tr>
<td>Job Failed</td>
<td>FAILED</td>
<td>COMPLETE</td>
<td>Vacated</td>
</tr>
</tbody>
</table>
Miscellaneous LCRM / Moab Comparisons

- Moab time expressions are in HH:MM:SS while LCRM is HH:MM.
- Moab job priorities are integers that range from -1,000,000,000 to 1,000,000,000. LCRM priority is a floating point number between 0.0 and 1.0.
- Moab system priorities go even higher.
- LCRM overloads job states (ELIG, RUN, COMPLETE, etc.) with reasons a job is not running (JRESLIM, QTOTLIM, etc.).
- Moab has pure job states and provides checkjob to show why a job is not running.
LCRM Wrappers

- Perl scripts that communicate with Moab while emulating LCRM behavior
- Located in /usr/bin on Linux machines
- Located in /opt/freeware/bin on AIX machines
- Six supported commands:
  - psub
  - palter
  - phold
  - prel
  - pstat
  - prm
psub Wrapper

- Legacy LCRM jobs command scripts must be submitted by psub
- Provide most of the options of the LCRM psub
- Some options are ignored and warnings are issued
- Archaic options are ignored and warnings can be seen when psub –v is used
- Generates the PSUB_ variables
- Deprecated variables identified in LCRM release notes are not supported
- Output / Error default file naming conventions are preserved
- Automatically translates LCRM’s HH:MM time into Moab’s HH:MM:SS format
## psub to msub Comparison

<table>
<thead>
<tr>
<th></th>
<th>psub option</th>
<th>msub option</th>
</tr>
</thead>
<tbody>
<tr>
<td>start after time</td>
<td>-A</td>
<td>-a</td>
</tr>
<tr>
<td>account (bank)</td>
<td>-b</td>
<td>-A</td>
</tr>
<tr>
<td>constraint</td>
<td>-c</td>
<td>-l feature=&lt;value&gt;</td>
</tr>
<tr>
<td>error output file</td>
<td>-e</td>
<td>-e</td>
</tr>
<tr>
<td>join error w/ output</td>
<td>-eo</td>
<td>-j</td>
</tr>
<tr>
<td>expedite</td>
<td>-expedite</td>
<td>-l qos=expedite</td>
</tr>
<tr>
<td>node count</td>
<td>-ln</td>
<td>=l nodes=&lt;count&gt;</td>
</tr>
<tr>
<td>mail at job end</td>
<td>-me</td>
<td>-m e</td>
</tr>
<tr>
<td>output file</td>
<td>-o</td>
<td>-o</td>
</tr>
<tr>
<td>node pool</td>
<td>-pool</td>
<td>-q</td>
</tr>
<tr>
<td>project name</td>
<td>-prj</td>
<td>-l project=&lt;name&gt;</td>
</tr>
<tr>
<td>standby</td>
<td>-standby</td>
<td>-l qos=standby</td>
</tr>
<tr>
<td>wall-clock limit</td>
<td>-tW</td>
<td>-l walltime=&lt;time&gt;</td>
</tr>
</tbody>
</table>
SLURM Environment Variables

- Users should wean their jobs from referencing the PSUB_ environment variable set and substitute the SLURM_ environment variables when running on LC machines:
  - SLURM_JOBID
  - SLURM_NNODES
  - SLURM_TASK_PID
  - Etc.

- For more info, invoke “man srun” and skip to the section entitled: “ENVIRONMENT VARIABLES”
Feature Constraints and Pools

- Once upon a time, the LCRM `psub -c` option overloaded features with pools.
- In newer LCRM versions, the `psub -pool` option was added to provide a more versatile job spec.
- The `psub` wrapper supports the `psub -pool` option and translates this to a Moab class (queue).
- Users are advised to submit LCRM legacy scripts that specify a node pool with the `-pool` option and not with the `-c` option. The `psub` wrapper will reject a class option specified with the `-c` option.
- The `psub` wrapper translates the `psub -c` option into a feature.
pstat Wrapper

- Interrogates Moab to retrieve job information
- Displays job information in LCRM pstat format
- Combined pstat options not allowed
  - E.g. `pstat -fT 53` must be written as `pstat -f -T 53`
- Translates Moab job states to LCRM job states
- Translates Moab’s HH:MM:SS format to LCRM’s HH:MM format
- Priority field displays actual Moab value
The Many Ways to View the Queue

- showq
- mdiag -j (-v)
- pstat -A
- squeue
- mjstat
- showq -c shows completed jobs
- pstat -T shows completed jobs
**prm Wrapper**

- Provides the same functionality as the mjobctl –c or canceljob commands.
- The email to user option is not supported.
- The –gt option will effectively modify the job’s remaining time and send a signal.
- Do not use scancel to remove jobs submitted to Moab using msub or the psub wrapper.
LLNL Utilities for Moab

- mjstat – emulates the output of spjstat
- lcrm2moab – converts an LCRM legacy job command script to conform to msub style
Determining a User’s Accounts

- LCRM provides bac, defbank, and newbank
- The user’s account membership is set by administrators (just like with LCRM)
- The user’s default account is also set by administrators
- The user can view their account membership and default accounts by invoking mdiag –u <user>
  - ALIST presents the accounts of which the user is a member
  - ADEF indicates the default account
- There is no Moab analog to setting a “newbank” and automatically charging all following jobs to the new account
Determining a Job’s Node and Time Limits

- LCRM provides the plim and brlim tools to display the limits that are imposed for a host and bank.
- CPU-based time limits are not currently imposed.
- Moab provides the mdiag –t –v command to display host limits.
- Moab provides the mdiag –c command to display class (SLURM partition) limits.
- Limits will also be advertised in news job.lim.<host>
Determining a User’s Fair Share and a Job’s Priority

- LCRM provides pshare to display a user and bank shares and usage
- LCRM does a poor job of indicating how a user’s shares and usage contribute to a job’s priority
- Moab’s mdiag command offers the –f option to display a user’s fair share calculation based on shares and usage
- Moab’s mdiag command offers the –p option to display how all of the factors combine to create a job’s priority
Expediting a Job

- LCRM provides pexp to allow administrators and privileged users the ability to expedite a normal job.

- In lieu of a pexp wrapper, as a privileged user to expedite your own job, invoke
  - `mjobctl -m qos=expedite <jobID>`

- In lieu of a pexp wrapper, as an admin to expedite a user’s job, invoke
  - `mjobctl -m flags+=ignpolicies <jobID>`
  - `mjobctl -p +1 <jobID>`
LCRM Library Emulation

- **The bank library calls are no longer supported**
  - `lrmbankallbanks()`
  - `lrmbankcurbank()`
  - `lrmbankdefbank()`
  - `lrmbanksetcurbank()`
  - `lrmbanksetdefbank()`

- **The calls to request a signal at a time remaining are emulated**
  - `lrmgetresource()`
  - `lrmgettime()`

- **The polling method for being notified when time limit is about to expire is emulated**
  - `lrmwarn()`
Native Remaining Time APIs

- When SLURM is the underlying resource manager, SLURM’s `slurm_get_rem_time()` is very accurate and fast.
- Moab offers its own call, `MCCJobGetRemainingTime()` that works no matter what the underlying resource manager is.
Yogrt Library

- An LLNL developed library to ease the transition to Moab
- Offers a single API to get a job’s remaining time that will work for Moab and LCRM
- Chooses the fastest and most accurate method for retrieving the data
Common Questions

- **When will my job run?**
  - showstart will display estimated start and completion times

- **Why is my job not running?**
  - checkjob <jobID>

- **When would a job of n processors run?**
  - showstart <n@duration>

- **What accounts (banks) can I use?**
  - mdiag –u <user>

- **Why did my job die?**
  - checkjob <jobID> output shows
    - Whether a job was cancelled or terminated on its own
    - The job’s exit (completion) code
Further Information
User Resources

- /usr/local/docs/Moab
- ICRM Group Website: https://computing.llnl.gov/jobs/moab/
  - Moab Quick Start Guide
  - Moab at LC User Guide
  - psub to msub translation table
  - Options for LCRM Library Users
  - Understanding Moab Job Priorities
- Cluster Resources, Inc.