Job scheduler details

Advanced Computing Center for Research & Education
Outline

1. Batch queue system overview
2. Torque and Moab
3. Submitting jobs
Goals of a batch management system

- **Management goals**
  - Define cluster mission objectives and performance criteria
  - Evaluate current and historical cluster performance

- **Administrative goals**
  - Maximize utilization and cluster responsiveness
  - Tune fairness policies and workload distribution
  - Automate time-consuming tasks
  - Trouble-shoot job and resource failures
  - Integrate new hardware and cluster services into the batch system

- **End user goals**
  - Manage current workload
  - Identify available resources
  - Minimize workload response time
  - Track historical usage
  - Identify effectiveness of prior submissions
Components of a batch system

- **Master Node**
  - the node that `pbs_server` is running. Depending on the needs of the systems, a master node may be dedicated to this task or may fulfill the roles of other components as well.

- **Submit/Interactive Nodes**
  - provide an entry point to the system for users to be able to manage their workload.
  - users are able to submit and track their jobs, do testing and troubleshooting environment problems.
  - these nodes will have client commands (e.g., `qsub`, `qhold`, etc) available.
Components of a batch system

- **Compute Nodes**
  - work horses of the system.
  - execute submitted jobs.
  - each compute node, **pbs.mom** will be running to start, kill and manage submitted jobs.
  - communicates with **pbs.server** on the master node. Depending on the needs of the systems, a compute node may double as the master node (or more).

- **Resources**
  - can include high-speed networks, storage systems, license managers, etc. Availability of these resources is limited and need to be managed intelligently to promote fairness and increased utilization.
Basic job flow

- **Life cycle of a job:**
  - **Creation**
    - Submit script is written to hold all of the parameters of a job. (e.g. how long a job should run (i.e., walltime), what resources are necessary to run and what to execute)
  - **Submission (with qsub)**
    - Once submitted, the policies set by the administration and technical staff of the site will dictate the priority of the job and therefore, when it will start executing.
  - **Execution (query with qstat)**
  - **Finalization**
The job’s state indicates its current status and eligibility for execution

Pre-execution states

<table>
<thead>
<tr>
<th>State</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>idle</td>
<td>job is currently queued and eligible to run but is not executing (also, notqueued)</td>
</tr>
<tr>
<td>hold</td>
<td>the job is idle and is not eligible to run due to a user, admin, or batch system hold (also, batchhold, systemhold, userhold)</td>
</tr>
<tr>
<td>staged</td>
<td>job has been migrated to another scheduler but has not yet started executing</td>
</tr>
</tbody>
</table>
# Job states

## Execution states

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Starting</strong></td>
<td>the batch system has attempted to start the job and the job is currently performing pre-start tasks which may including provisioning resources, staging data, executing system pre-launch scripts, etc.</td>
</tr>
<tr>
<td><strong>Running</strong></td>
<td>job is currently executing the user application</td>
</tr>
<tr>
<td><strong>Suspended</strong></td>
<td>job was running but has been suspended by the scheduler or an admin. The user application is still in place on the allocated compute resources but it is not executing</td>
</tr>
<tr>
<td><strong>Canceling</strong></td>
<td>job has been canceled and is in process of cleaning up</td>
</tr>
</tbody>
</table>

## Post-Execution states

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Completed</strong></td>
<td>the job has completed running without failure</td>
</tr>
<tr>
<td><strong>Removed</strong></td>
<td>the job has run to its requested walltime successfully but has been canceled by the scheduler or resource manager due to exceeding its walltime or violating another policy. This includes jobs which have been canceled by users or admins either before or after a job has started.</td>
</tr>
<tr>
<td><strong>Vacated</strong></td>
<td>the job was canceled after partial execution due to a system failure</td>
</tr>
</tbody>
</table>
Batch Queuing System

- Policy-based intelligence engine that integrates scheduling, managing, monitoring, and reporting of cluster workloads.
- Goal: guarantee to meet the service levels while maximizing job throughput
- Software suite:
  - TORQUE/PBS resource manager
  - Moab/Maui 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.

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.
What is Torque’s job as the resource manager.
- Accepting and starting jobs across a batch farm.
- Cancelling jobs.
- Monitoring the state of jobs.
- Collecting returned outputs.

What is Moab’s job?
- Moab makes all the decisions.
- Should a job be started asking questions like:
  - Is there enough resource to start the job?
  - Given all the jobs I could start which one should I start?

Moab runs a scheduling iteration:
- When a job is submitted.
- When a job ends.
- At regular configurable intervals.
Torque and Moab

Job scheduler details

12 / 25
Detailed job flow

- Script submitted to Torque (using qsub) specifying required resources
- Moab periodically retrieves from Torque list of potential jobs, available node resources, etc.
- Moab prioritizes jobs in idle queue
- When resources become available, Moab tells Torque to execute certain jobs on particular nodes
- Torque dispatches jobs to the PBS MOMs (machine oriented miniserver) running on the compute nodes - pbs_mom is the process that starts the job script
- Job status changes reported back to Moab, information updated
- Moab sends further instructions to Torque
- Moab updates occur roughly every 2 minutes (configurable)
The queue

- Queue divided into 3 subqueues:
  - **active** - running
  - **eligible** - idle, but waiting to run
  - **blocked** - idle, held, deferred

- A job can be blocked for several reasons, e.g.,
  - requested resources not available
  - reserved nodes offline
  - cluster policy: e.g. user has maximum of 10 jobs in eligible queue
  - user places intentional hold
  - Moab supports four distinct types of holds, **user** holds, **system** holds, **batch** holds, and **defer** holds
Hold queue

- User hold (*qhold*)
- System hold: hold by system admin
- Batch hold
  - No resources
  - System limits
  - Resource manager failure
  - Policy violation
  - ......
- Job deferred: first deferred for some amount of time, then put into hold
  - At this time, it will be allowed back into the idle queue and again considered for scheduling. (several opportunities)
Job priority

- Each submitted job has a priority
- Job with the highest priority gets executed first
- The priority of a job is calculated based on a number of factors:

<table>
<thead>
<tr>
<th>Factor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Credential</td>
<td>relative importance of certain groups or accounts</td>
</tr>
<tr>
<td>Fairshare</td>
<td>favor jobs based on short term historical usage</td>
</tr>
<tr>
<td>Resource</td>
<td>weighting jobs by the amount of resources requested</td>
</tr>
<tr>
<td>Service</td>
<td>specifies which service metrics are of greatest value to the site</td>
</tr>
<tr>
<td>Target service</td>
<td>takes into account job scheduling performance targets</td>
</tr>
<tr>
<td>Usage</td>
<td>applies to active jobs only</td>
</tr>
<tr>
<td>Job attribute</td>
<td>allow the incorporation of job attributes into a job’s priority</td>
</tr>
</tbody>
</table>

Priority=CREDEIGHT*CREEDComp+FSWEIGHT*FSComp+...

- check job priority: `mdiaq -p`
Fairshare scheduling helps steer a system toward usage targets by adjusting job priorities based on short term historical usage.

Moab’s fairshare can target usage percentages, ceilings, floors or caps for users, groups, accounts etc.

mdiag -f
# Fairshare example

**Vanderbilt Logo**

```
FSINTERVAL 12:00:00
FSDEPTH 4
FSDECAY 0.5
FSPOLICY DEDICATEDPS

# all users should have a fs target of 10%
USERCFG[DEFAULT] FSTARGET=10.0

# user john should get 20% CPU usage
USERCFG[john] FSTARGET=20.0

# reduce staff priority if group usage exceed 15%
GROUPCFG[staff] FSTARGET=15.0-

# give group orion additional priority if usage drops below 25.7%
GROUPCFG[orion] FSTARGET=25.7+

FSUSERWEIGHT 10
FSGROUPWEIGHT 100
```
Fairshare example

Interval 12:00:00
FSDEPTH 4
FSDECAY 0.5

<table>
<thead>
<tr>
<th>Interval</th>
<th>Total</th>
<th>User john Usage</th>
<th>Total Cluster Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>60</td>
<td></td>
<td>110</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td>125</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td></td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td></td>
<td>150</td>
</tr>
</tbody>
</table>

Usage = \[
\frac{60 + 0.5^1 \times 0 + 0.5^2 \times 10 + 0.5^3 \times 50}{110 + 0.5^1 \times 125 + 0.5^2 \times 100 + 0.5^3 \times 150}
\]
Submitting jobs

- You create a job submission script (a shell script containing the set of commands you want run on some set of cluster compute nodes)

- Submit to job scheduler:
  
  qsub[options] job_submission_script

  - It returns a job id upon successful submission.
  - You will need the id to monitor, deleting the job.

- Jobs are queued up till the system is ready to run it (requested resources become available)

- The scheduler selects which jobs to run, when, and where, according to a predetermined site policy meant to balance competing user needs and to maximize efficient use of the cluster resources
Writing job script

#!/bin/bash #first line defines shell
#PBS -M my.address@vanderbilt.edu #send status/progress emails
#PBS -m bae #email at beginning, abort, & end
#PBS -l nodes=1:ppn=1 #resources (-l) required for job
#PBS -l walltime=00:05:00 #REQUIRED! Estimated wall clock
   #format: (hh:mm:ss)
#PBS -l mem=1000mb
#PBS -o myjob.output # send stdout to myjob.output
#PBS -j oe # join stdout/err to myjob.output
myexecutable input1 input2
## PBS Environmental variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PBS_O_HOME</td>
<td>the HOME environmental variable of the submitter</td>
</tr>
<tr>
<td>PBS_O_HOST</td>
<td>the name of the host upon which the <code>qsub</code> command is running</td>
</tr>
<tr>
<td>PBS_O_QUEUE</td>
<td>the name of the original queue to which the job was submitted</td>
</tr>
<tr>
<td>PBS_O_WORKDIR</td>
<td>the absolute path of the current working directory of the <code>qsub</code> command</td>
</tr>
<tr>
<td>PBS_JOBID</td>
<td>the job identifier assigned to the job by the batch system</td>
</tr>
<tr>
<td>PBS_NODEFILE</td>
<td>the name of the file contain the list of nodes assigned to the job (for parallel jobs)</td>
</tr>
</tbody>
</table>
### Useful commands

<table>
<thead>
<tr>
<th>command</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>qsub [options] &lt;pbs_script&gt;</td>
<td>submit job for execution</td>
</tr>
<tr>
<td>qdel, qhold, qrls &lt;jobID&gt;</td>
<td>remove, hold, release hold</td>
</tr>
<tr>
<td>qstat</td>
<td>view job(s) status</td>
</tr>
<tr>
<td>q</td>
<td>view queue status with nice format</td>
</tr>
<tr>
<td>showq</td>
<td>view queue status</td>
</tr>
<tr>
<td>pbsnodes -l -a</td>
<td>view nodes &amp; attributes</td>
</tr>
<tr>
<td>checkjob -v &lt;jobID(s)&gt;</td>
<td>view job(s) status</td>
</tr>
<tr>
<td>mdiag -f</td>
<td>check fairshare</td>
</tr>
<tr>
<td>mdiag -v -p</td>
<td>check job priority</td>
</tr>
<tr>
<td>mdiag -v -j &lt;jobID&gt;</td>
<td>resource summary</td>
</tr>
<tr>
<td>tracejob -n &lt;#days&gt; &lt;jobID&gt;</td>
<td>trace job history</td>
</tr>
</tbody>
</table>
Checking cluster utilization

- Overview charts:
  - Number and percentage of active compute CPUs
  - Number and percentage of active compute nodes
  - Number of active, eligible, and blocked jobs

http://www.accre.vanderbilt.edu/?page_id=767
Scheduler Etiquette

- Scheduler policies:
  http://www.accre.vanderbilt.edu/?page_id=89
- Limits on:
  - Number of jobs in queue
  - Maximum and minimum job lengths
  - Memory usage
- We place special restrictions when necessary.