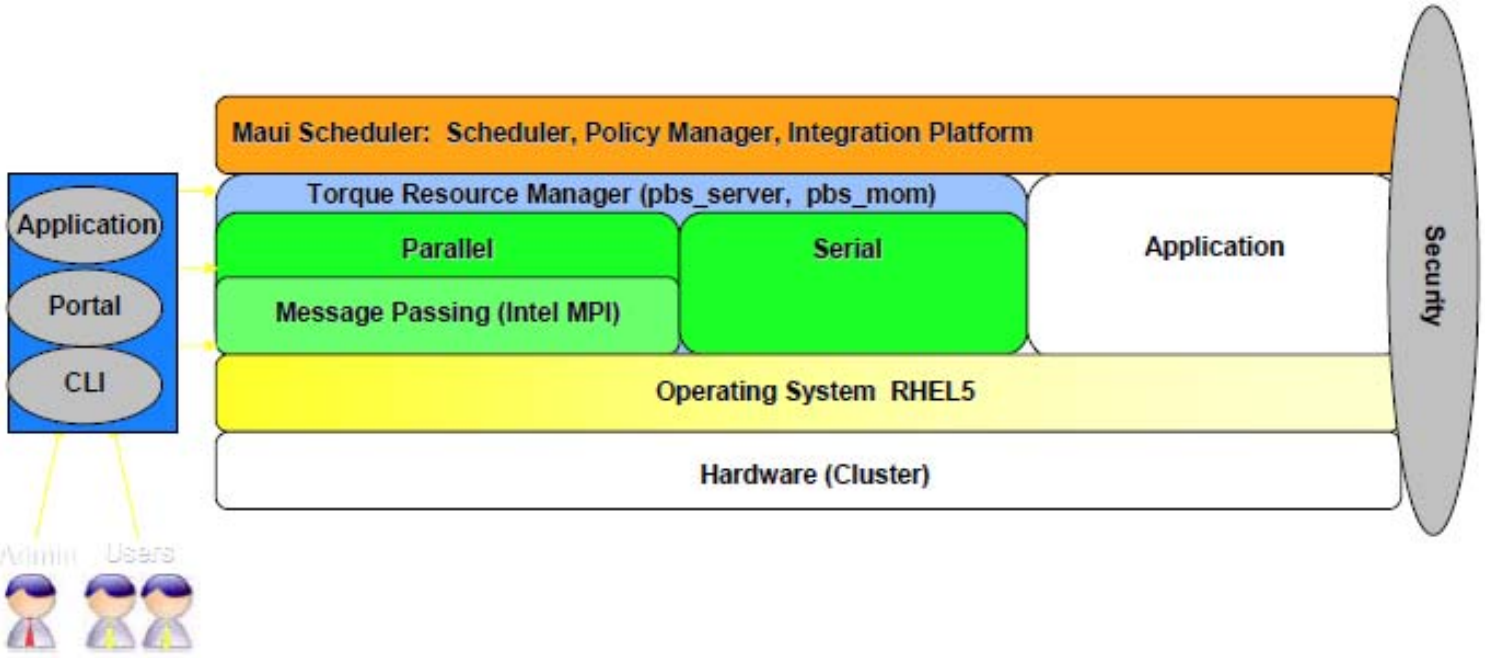
Advanced Technical Skills (ATS) North America

Job Scheduling with Maui and Torque

IBM High Performance Computing
January 2011

Y. Joanna Wong, Ph.D.
yjw@us.ibm.com

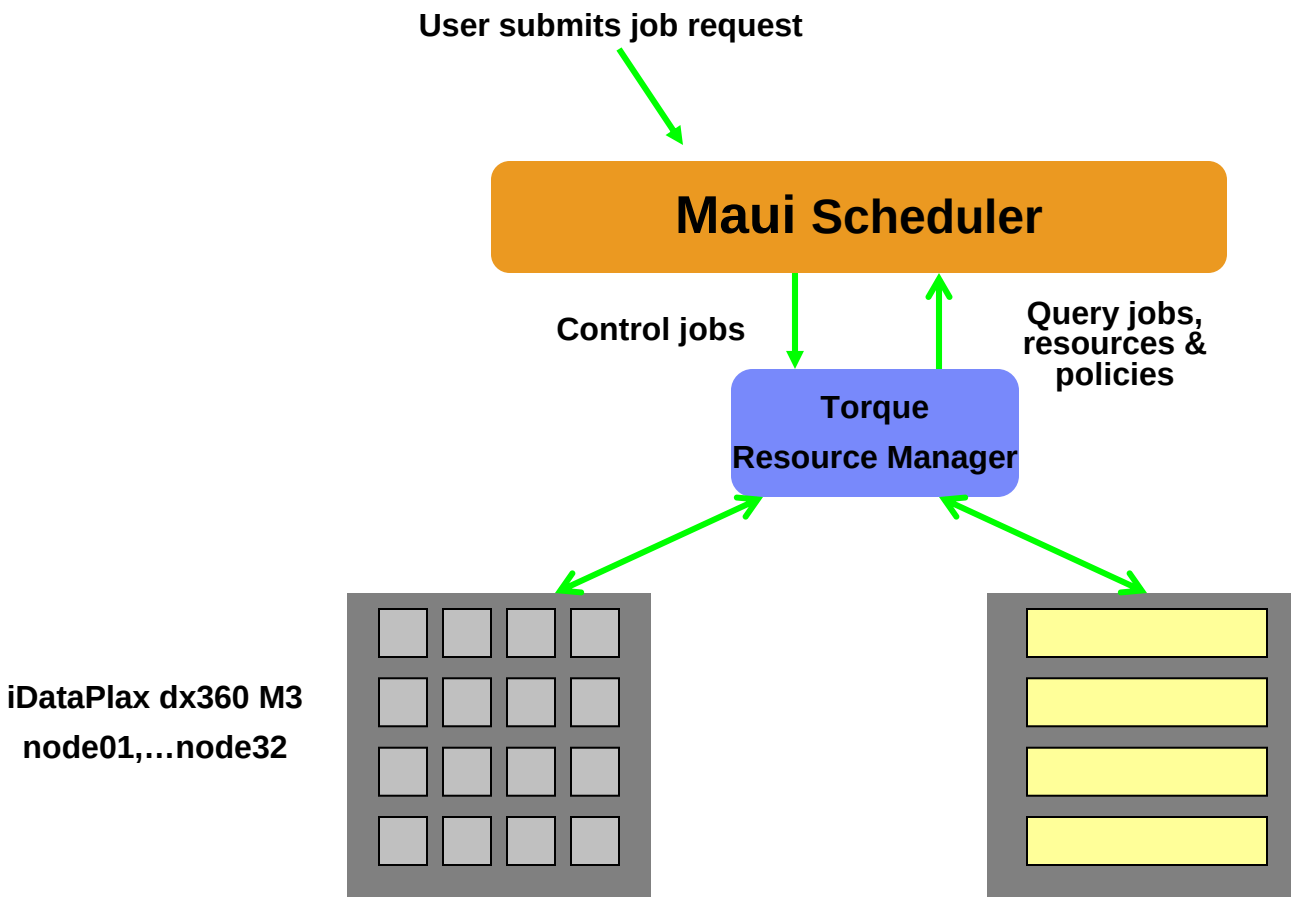
Torque / Maui Scheduler



Source: Adaptive Computing

Some terminology..

- **Resource Manager**
  - Manages a queue of jobs for a cluster of resources
  - Launches job to a simple FIFO job queue
  - The Resource Manager is Torque
- **Workload manager**
  - A scheduler that integrates with one or more Resource Managers to schedule jobs across domains of resources (servers, storage, applications)
  - Prioritizes jobs
  - Provides status of running and queued jobs
  - Implements fair-share mechanism and achieving efficient utilization of resources
  - Enforces established policy
  - Collects and reports resource usage statistics
  - The workload manager is Maui



Maui / Torque scheduler

- Maui Cluster Scheduler
  - Open source job scheduler maintained and supported by Adaptive Computing (formerly Cluster Resources, Inc) in mid-90s for use on clusters and supercomputers with contributions from several academic institutions and national research labs.
  - The Maui Cluster Scheduler is NOT a resource manager.
    - The scheduler tells the resource manger what to do, when and where to run jobs
    - Can be integrated with several resource managers, including Torque
  - Capable of supporting multiple scheduling policies, dynamic priorities, extensive reservations and fair-share capabilities
  - Users typically submit jobs and query state of resources and jobs through the resource manager.
    - Users will submit the job script for the resource manager.

- Torque Resource Manager
  - An open source resource manager providing control over batch jobs and distributed compute nodes
  - Community effort based on the original PBS project with enhancement in scalability, fault tolerance and feature extensions over standard OpenPBS
  - Fault Tolerance Additional failure conditions checked/handled Node health check script support
  - Aggressive development with new capabilities – advanced diagnostics, job arrays, high-throughput support
  - Scalability
    - Significantly improved server to MOM communication model
    - Ability to handle larger clusters (over 20,000 cores)
    - Ability to handle tens of thousands of jobs
    - Ability to support larger server messages

- Documentation on Maui and Torque resource manager at Adaptive Computing
  - Links from the url:
    http://www.adaptivecomputing.com/resources/docs/maui/index.php

    http://www.adaptivecomputing.com/resources/docs/torque/index.php

- MOAB workload manager is the commercially licensed policy-based job scheduler from Adaptive Computing that was initially based on the Maui cluster scheduler.

Maui commands

- Majority of Maui commands are for use by the scheduler administrators. For command details, access links from:

  http://www.adaptivecomputing.com/resources/docs/maui/a.gcommandoverview.php

Maui end user commands	Description
checkjob	Provide detailed status report for specified job
mjobctl e.g. mjobctl –c *JOBid*	Control and modify job cancels a job with ID *JOBid*
showbf	Show resource available for jobs with specific resource requirements
showq	Display all jobs in active, idle and non-queued states. The flags to display extended details can only be used by level 1, 2, or 3 scheduler administrators
showstart	Show estimates of when job can/will start
showres	Show existing reservations

Submitting jobs

- If Maui is configured to run as root, users can submit jobs to Maui directly using msub

  msub [-a datetime] [-A account] [-c interval] [-C directive_prefix] [-d path]
  [-e path] [-h] [-I] [-j join] [-k keep] [-K] [-l resource_list] [-m mail_options]
  [-M user_list] [-N name] [-o path] [-p priority] [-q destination] [-r]
  [-S path_list] [-u user_list] [-v variable_list] [-V] [-W additional_attributes]
  [-z][script]

- Jobs submitted by msub can run on any of the resources of the resources managers managed by Maui
- Jobs submitted to a resource manager (e.g. qsub for Torque) can only run resources managed by the resource manager**.**

Building Torque job script

- Users build job scripts and submit the job using the qsub command for scheduling

  qsub [-a date_time] [-A account_string] [-b secs] [-c checkpoint_options] [-C directive_prefix] [-d path] [-D path] [-e path] [-h] [-I] [-j join] [-k keep] [-l resource_list] [-m mail_options] [-M user_list] [-N name] [-o path] [-p priority] [-q destination] [-r c] [-S path_list] [-t array_request] [-u user_list] [-v variable_list] [-V] [-W additional_attributes] [-X] [-z] [script]

  http://www.clusterresources.com/torquedocs21/commands/qsub.shtml

- The job script is a plain text file
  - Includes shell scripting, comment lines
  - Command, directives specific to the batch system
    - The directive is an alternative to command line option to specify job attributes
  - All directive lines must precede shell script command
- Shell scripting is parsed at runtime
- The job script may be specified as qsub command line argument [script] or may be entered via STDIN or piped to qsub.

  cat job.script | qsub

Torque job script..

- The job script will be executed from the user's home directory
- For parallel jobs, the job script will be staged to and executed on the first allocated compute node.
- The job script will use the default user environment variables (set in the shell startup script e.g. .bashrc) unless the '-V' or '-v' flags are specified to include all current environment variables (-V), or selected environment variables (-v)
- qsub will pass the value of the environment variables HOME, LANG, LOGNAME, PATH, MAIL, SHELL and TZ to the job script and be assigned to a new name prefixed with PBS_O
- qsub will process a line as a directive is the string of characters starting with the first non white space character on the line and of the same length as the directive prefix matches the directive prefix.
- The directive prefix is determined in order of preference:
  - value of command line option "-C"
  - Value of environment variable PBS_DPREFIX if defined
  - The string "#PBS"

- Resources are requested at job submission with:
  - with command line option –l for qsub. For example: -l walltime=1:00:00 –l nodes=4:ppn=4
  - Directives in the job script. For example,

    #PBS –l walltime=1:00.00

    #PBS –l nodes=4:ppn=4
- A few frequently requested resources:
- -l mem=<size> is maximum amount of physical memory used by the job (Ignored on Linux is number of nodes is not 1)

  where <size> is defined in form of number of bytes (suffix b) or words (suffix w)

  The multiplier is k=1024, m=1,048,576, g=1,073,741,824, t=1,099,511,627,776

  e.g. –l mem=1gb
- -l vmem=<size> is maximum amount of virtual memory used by all concurrent processes in the job
- -l walltime=<seconds> or [[HH:]MM:]SS is the maximum amount of real time during which the job is in run state
- -l cput=<seconds> or [[HH:]MM:]SS is the maximum amount of CPU time used by all processes in the job

- -l nodes={<node_count> <hostname>}[:ppn=<ppn>][:<property>][:<property>][+]

  is the number and/or type of nodes to be reserved for use by the job. The value is one or more node_specs joined with the '+' character, "node_spec[+node_spec...]". Each node_spec is an number of nodes required of the type declared in the node_spec and a name or one or more property or properties desired for the nodes. The number, the name, and each property in the node_spec are separated by a colon ':'. If no number is specified, one (1) is assumed. The name of a node is its hostname. The properties of nodes are:
  ppn=# - specifying the number of processors per node requested. Defaults to 1.
  property - a string assigned by the system administrator specify a node's features

  For example:

  -l nodes=2:ppn=4+4:ppn=2 : requesting 2 nodes with 4 cores per node and 4 nodes with 2 cores per node, a total of 6 nodes with 16 cores

  -l nodes=node001+node003+node005 : requesting 3 specific nodes by hostname

- -N name : Declares a name for the job. If –N is not specified, the job name is the base name of the job script.
- Running interactive jobs:
  - -I option specified on the command line
  - script include the –I directive
  - Job attributed interactive declared to be true: -W interactive=true
  - During execution of the interactive job, input to and output from the job is passed through the qsub.
  - Useful for debug while building and testing applications

Job Script Environment Variables

- Exported batch environment variables that can be used in job script :

Variable	Description
PBS_JOBNAME	user specified jobname
PBS_JOBID	Job identifier assigned by the batch system
PBS_ARRAYID	value of job array index for this job
PBS_NODEFILE	Name of file containing the list of node(s) assigned
PBS_QUEUE	Name of queue from which the job will be executed
PBS_O_HOST	Name of the host upon which the qsub command is running
PBS_O_QUEUE	Name of original queue to which job was submitted
PBS_O_WORKDIR	Absolute path of current working directory of qsub
PBS_O_LOGNAME	name of submitting user
PBS_O_HOME	Home directory of submitting user
PBS_O_PATH	Path variable user to locate executables within job script

- # specify RunName
- RunName=$PBS_JOBNAME

- sort -u $PBS_NODEFILE | awk '{print $1 " ifhn="$1"-ib0"}' > $MPDHOSTS
- ### sort -u $PBS_NODEFILE > $MPDHOSTS

- # make a hostfile in the working directory
- cat $PBS_NODEFILE | awk '{print $1 " ifhn="$1"-ib0"}' > hf.${NPROCS}
- ## cat $PBS_NODEFILE > hf.${NPROCS}
- fi

- echo "myhost is : $myhost"
- mpdboot --totalnum=$nodes --rsh=/usr/bin/ssh --file=${MPDHOSTS} --ifhn=${myhost}
- ## mpdboot --totalnum=$nodes --rsh=/usr/bin/ssh --file=${MPDHOSTS}
- mpdtrace -l