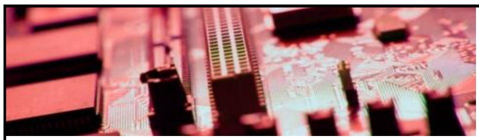


Moab / Torque

Scheduling and Batch Systems

LCI Workshop, August 2014

Brian Haymore
Center for High Performance Computing
University of Utah

Goals & Overview

Our goal is to cover the various aspects of managing the workload of a cluster using Moab and Torque. We will cover the various software components, deployment, and configuration. A more advanced overview along with operational examples will then be covered to detail scheduling policies including fairshare, advanced reservations, quality of service, allocation management, backfill, preemption, credentials, etc. We will also look at how to diagnose and troubleshoot the job queue behavior as well as ways to troubleshoot the scheduling/batch system components.

Definitions

Torque: The *Terascale Open-source Resource and QUEue Manager* is a distributed resource manager providing control over batch jobs and distributed compute nodes.

Moab: The Moab Cluster Suite is a cluster workload management package, available from Adaptive Computing, Inc, that integrates scheduling, managing, monitoring, and reporting of cluster workloads.

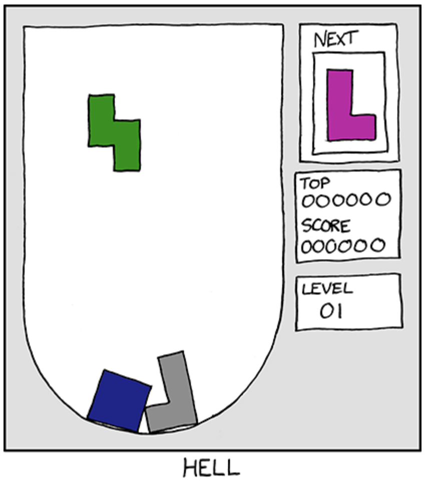
History

- Moab
  - Started in mid 90's as the Maui Cluster Scheduler.
  - Developed initially at MHPCC by David Jackson.
  - Maui was re-released under an open-source license in 2000.
  - In 2001 Moab was released as a next generation follow up to Maui by CRI.
  - David Jackson founded Cluster Resources Inc, now Adaptive Computing.
  - Moab is a closed source commercial product.
- Torque
  - PBS (Portable Batch System) was initially developed for NASA in 1991.
  - In 1998 an open-source version was released as OpenPBS.
  - OpenPBS was not actively developed in favor of the commercial PBS Pro version.
  - Terascale Open-source Resource and QUEue Manager (Torque) was released in 2003.
  - The Torque project is hosted by Adaptive Computing.

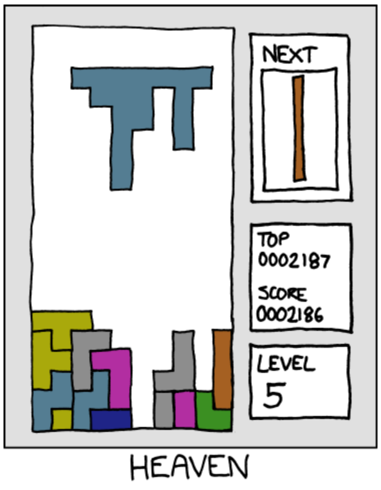
Agenda

- **Torque**
  - Topology Overview
  - Service Overview
  - Job Life Cycle Overview
  - Deployment
  - Configuration & Setup
  - Command Overview
  - Logs and Accounting
- **Moab**
  - Topology Overview
  - Service Overview
  - Deployment
  - Configuration
  - Command Overview
  - Logs and Stats
- **Policies, tuning, & advanced config.**
  - Priorities
  - Fairshare
  - Backfill
  - Quality of Service
  - Preemption
  - Advanced Reservations
- **Diagnostics and troubleshooting**
  - Jobs & Queue
  - Resource Manager & Scheduler
  - Execution Host
  - Example Issues & Resolutions
- **External Resources**
  - Web Links
  - Mailing List
  - Contact Info

Scheduling...I Laughed, I Cried.

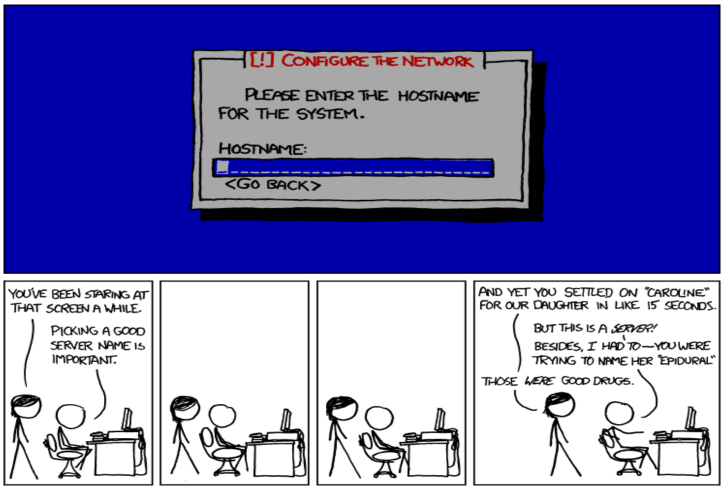


http://xkcd.com/724/



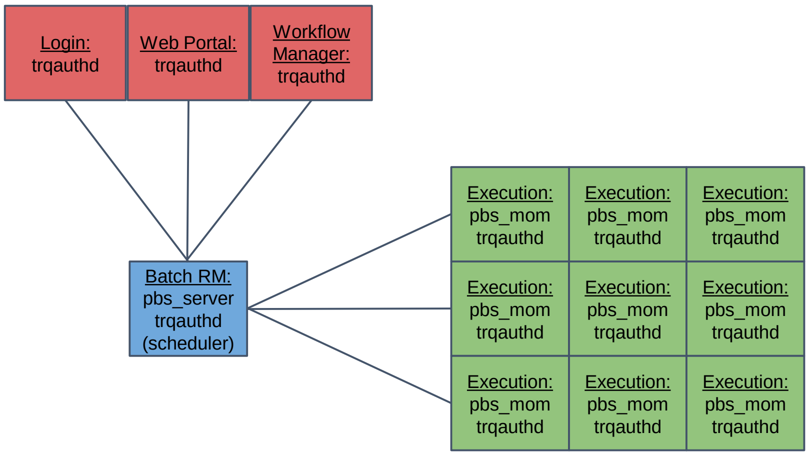
http://xkcd.com/888/

What is in a name?


[!] CONFIGURE THE NETWORK
PLEASE ENTER THE HOSTNAME FOR THE SYSTEM.
HOSTNAME:
<GO BACK>
YOU'VE BEEN STARING AT THAT SCREEN A WHILE.
PICKING A GOOD SERVER NAME IS IMPORTANT.
AND YET YOU SETTLED ON "CAROLINE" FOR OUR DAUGHTER IN LIKE 15 SECONDS.
BUT THIS IS A SERVER!
BESIDES, I HAD TO—YOU WERE TRYING TO NAME HER "EPIDURAL."
THOSE WERE GOOD DRUGS.

http://xkcd.com/910/

Torque Topology Diagram


Login: trqauthd
Web Portal: trqauthd
Workflow Manager: trqauthd
Batch RM: pbs_server trqauthd (scheduler)
Execution: pbs_mom trqauthd
Execution: pbs_mom trqauthd
Execution: pbs_mom trqauthd
Execution: pbs_mom trqauthd
Execution: pbs_mom trqauthd
Execution: pbs_mom trqauthd
Execution: pbs_mom trqauthd
Execution: pbs_mom trqauthd
Execution: pbs_mom trqauthd

Torque Service Components

- **pbs_server:** Torque (PBS) Batch Server
  - Run on the Head Node or Resource Manager Node of a cluster.
  - pbs_server provides:
    - Queue, Policy, and Resource configuration/definition.
    - Job scripts and job state data are stored by pbs_server.
    - pbs_server provides accounting logs that track usage and job timelines.
    - Diagnostic messages are sent to log files, but can also be configured to use the syslog facility.

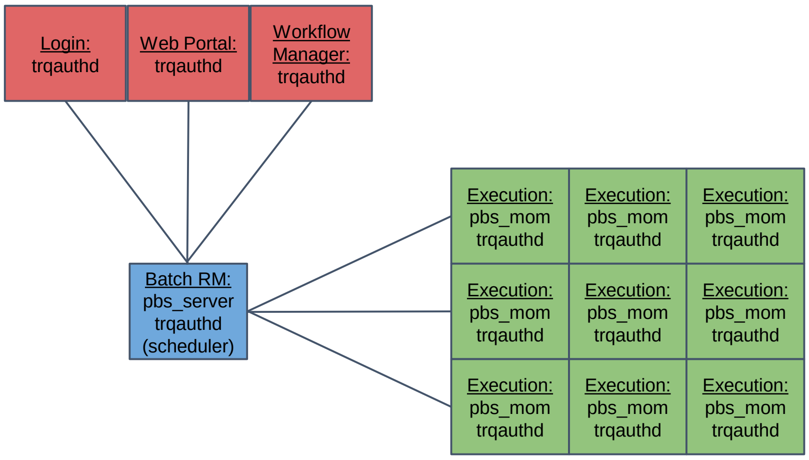
Torque Service Components

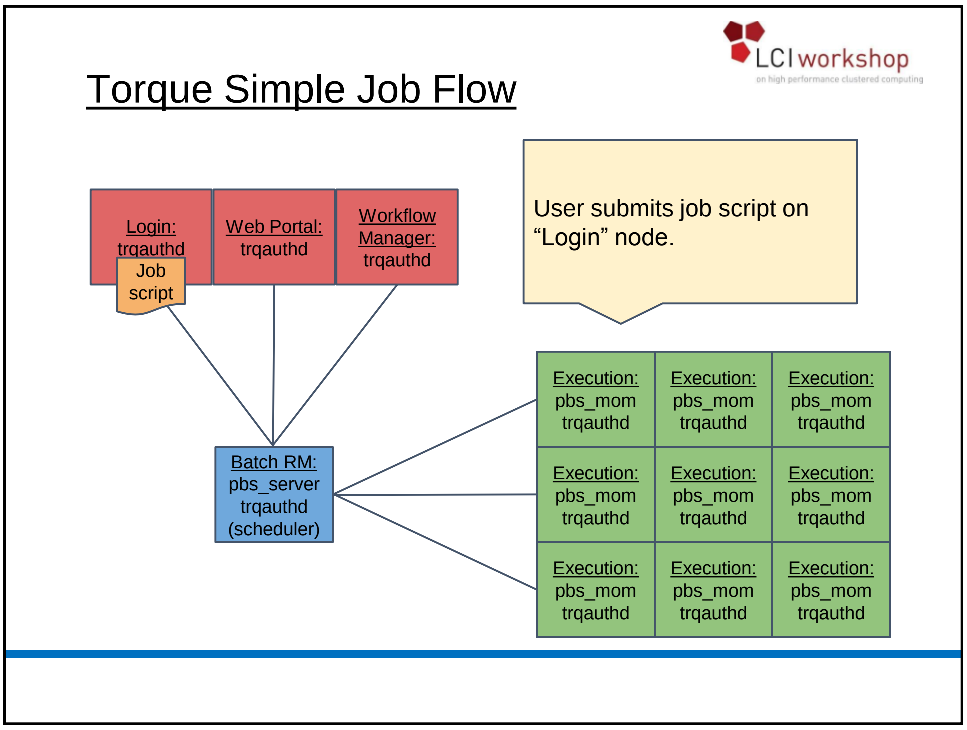
- **pbs_mom**: Torque (PBS) Batch Execution mini-Server
  - The MOM name comes from: PBS Machine Oriented Mini-server.
  - Runs on each of the cluster execution nodes (*usually*).
  - The “head” or first node of a job is referred to as Mother Superior. All other nodes in a job are referred to as Sister Moms.
  - pbs_mom provides:
    - Facilitates job execution per the direction of pbs_server.
    - Monitor and limit job’s resource usage.
    - Notify pbs_server when the job exits.
    - Interact with other pbs_mom’s to create a “sisterhood” to enable multi execution node jobs.
    - Spool stdout and stderr from the job script execution.

Torque Service Components

- **trqauthd**: Torque Authorization Daemon
  - Added in Torque 4.0.0 to replace pbs_iff as the authorization layer.
  - Run on each node in a cluster where client commands will be run. This usually means all execution and login/head nodes.

Torque Simple Job Flow

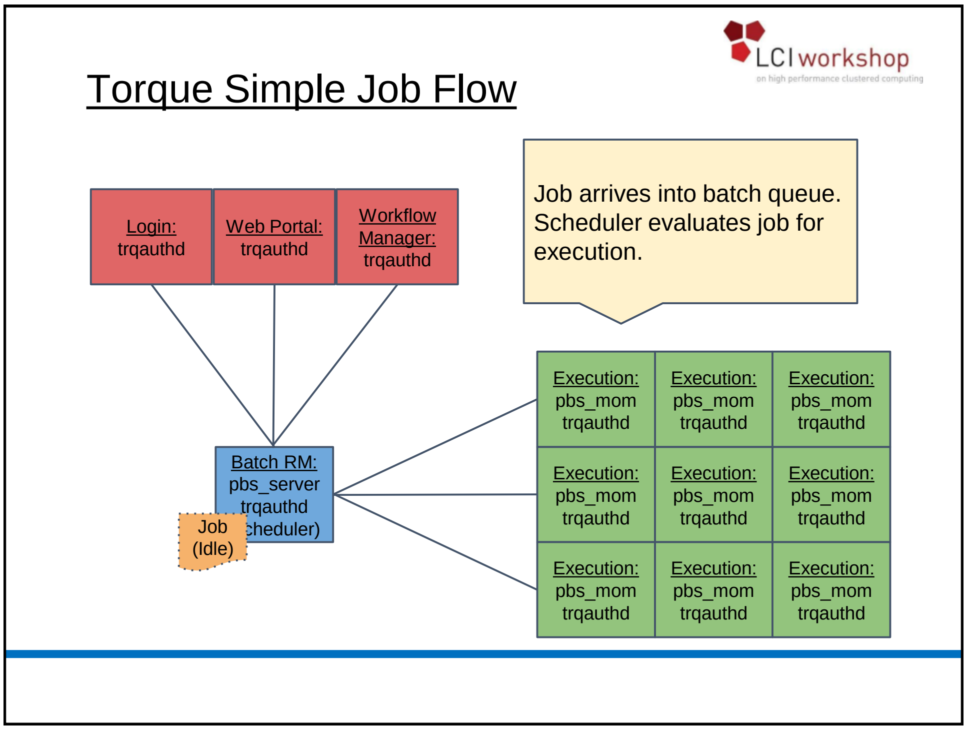

Login: trqauthd
Web Portal: trqauthd
Workflow Manager: trqauthd
Batch RM: pbs_server trqauthd (scheduler)
Execution: pbs_mom trqauthd
Execution: pbs_mom trqauthd
Execution: pbs_mom trqauthd
Execution: pbs_mom trqauthd
Execution: pbs_mom trqauthd
Execution: pbs_mom trqauthd
Execution: pbs_mom trqauthd
Execution: pbs_mom trqauthd
Execution: pbs_mom trqauthd



Torque Simple Job Flow

User submits job script on “Login” node.

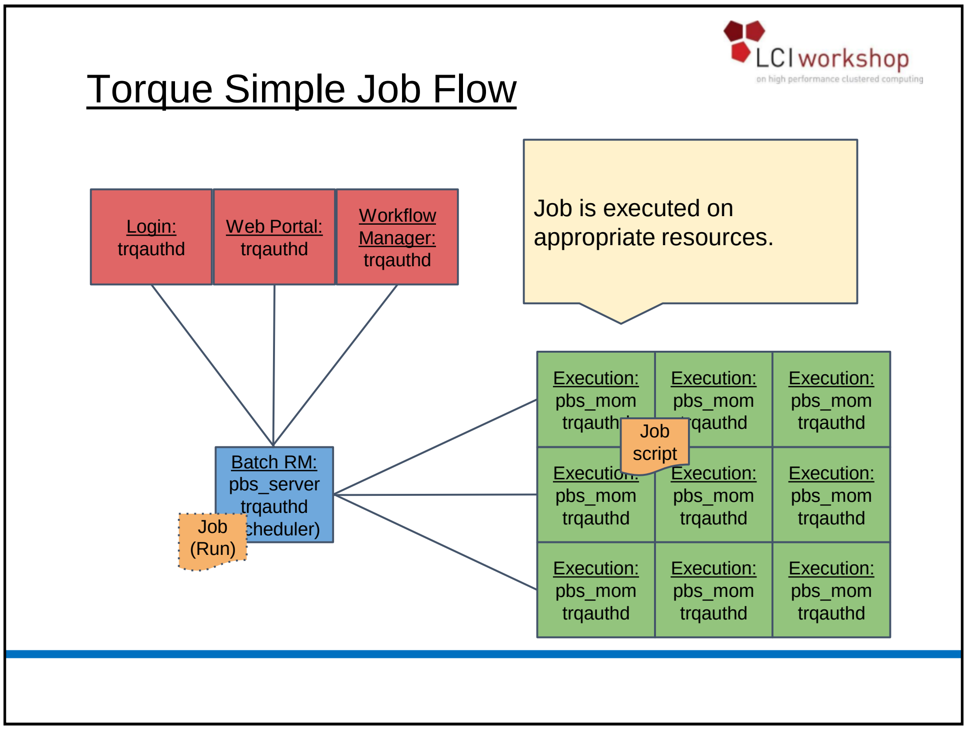
Login: trqauthd
Job script
Web Portal: trqauthd
Workflow Manager: trqauthd
Batch RM: pbs_server trqauthd (scheduler)
Execution: pbs_mom trqauthd



Torque Simple Job Flow

Job arrives into batch queue. Scheduler evaluates job for execution.

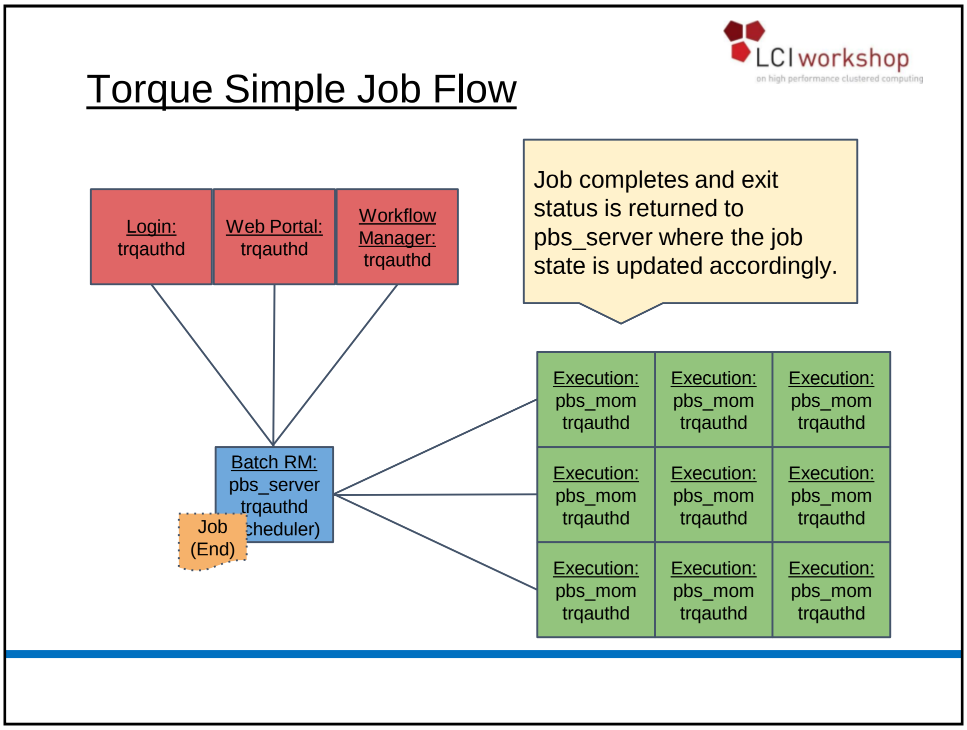
Login: trqauthd
Web Portal: trqauthd
Workflow Manager: trqauthd
Batch RM: pbs_server trqauthd (scheduler)
Job (Idle)
Execution: pbs_mom trqauthd



Torque Simple Job Flow

Job is executed on appropriate resources.

LCI workshop
on high performance clustered computing
Login: trqauthd
Web Portal: trqauthd
Workflow Manager: trqauthd
Batch RM: pbs_server trqauthd (scheduler)
Job (Run)
Execution: pbs_mom trqauthd
Job script

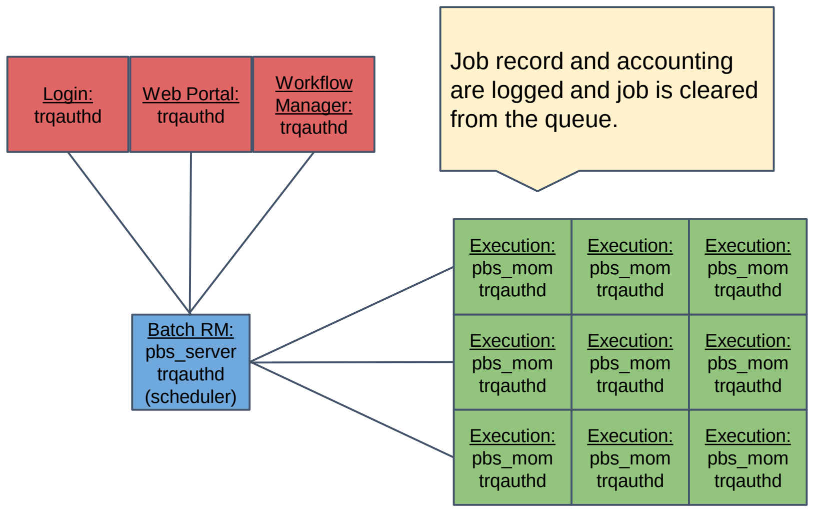


Torque Simple Job Flow

Job completes and exit status is returned to pbs_server where the job state is updated accordingly.

LCI workshop
on high performance clustered computing
Login: trqauthd
Web Portal: trqauthd
Workflow Manager: trqauthd
Batch RM: pbs_server trqauthd (scheduler)
Job (End)
Execution: pbs_mom trqauthd

Torque Simple Job Flow


Job record and accounting are logged and job is cleared from the queue.
Login: trqauthd
Web Portal: trqauthd
Workflow Manager: trqauthd
Batch RM: pbs_server trqauthd (scheduler)
Execution: pbs_mom trqauthd

Torque Deployment

- Download Torque from http://www.adaptivecomputing.com and follow the INSTALL document contained in the torque package.
- Customizations you might consider when you run ‘configure’:
  - --prefix=(Insert install path or defaults to /usr/local)
  - --with-server-home=(Insert server home path or, defaults to /var/spool/torque)
  - --with-sched=no (where using Moab so no need to build this)
  - --with-maildomain=(insert your main domain here)
  - --disable-spool (spool stdout/err to ~/.pbs_spool or ~/)
  - --enable-acct-x (logs x attributes to accounting logs)
  - --enable-syslog (leverage central syslog facility)
  - --enable-munge-auth (use munge instead of ruserok for user authorization, requires munge to be installed and setup).
- Select appropriate configure options then ‘make’ & ‘make install’

Torque Deployment

- **‘server home’** directory (defaults to /var/spool/torque).
  - Each host in a cluster that torque is used on will have a unique ‘server home’ directory.
  - This is where all logs, configuration, etc are stored according to the role of the host we are on (Exec, RM, Login).
- Starting up pbs_server for the first time (ONLY do this once).
  - Execute ‘pbs_server -t create’ and answer ‘y’ when asked if you really wanted to create the server database.
  - Execute ‘trqauthd’.
  - Execute ‘qmgr’ followed by ‘print server’ to verify things work.
- The INSTALL guide details out 3 ways to take our torque build and deploy it onto the rest of our cluster.
  - ‘make {install_clients, install_mom, install_server, …}’
  - ‘make rpm’
  - ‘make packages’ (option to include custom configuration files)
- Proceed with install method of choice to all hosts and start daemons.

Torque Initial Config & Setup

- ‘**qmgr’** is used to define the core setup in torque.
  - Some things we will look at via qmgr are:
    - ACLs.
    - Admin & operator role definition.
    - Queue definitions.
    - Resource definitions.
    - Min/default resources.
    - Timeout, delay, logging, and other tunable values.
  - ‘qmgr’ values can be adjusted via the ‘qmgr’ interactive shell or taking a file of preset values and piping that file through ‘qmgr’.
- Resources are defined in $SERVER_HOME/server_priv/nodes.
  - Can be populated via ‘hand’ or via ‘qmgr’ as noted above.
  - Defines name, proc count, gpu count, and node features.
- pbs_mom config is stored in $SERVER_HOME/mom_priv/config.
  - ACLs, timeout and logging tunable values.
  - Data management configuration to direct data staging method.
  - Define the path that torque will size and monitor.

Torque Initial Config & Setup

- Prologue/Epilogue facilities allow for script commands to be run before a job starts or after it ends. These help offer means to insure nodes are ready for a job or that we cleanup after a job.
  - $SERVER_HOME/mom_priv/prologue[.parallel]
    - A script framework that is executed prior to the start of the users job script (or interactive session).
  - $SERVER_HOME/mom_priv/epilogue[.parallel]
    - A script framework that is executed after a users job script (or interactive session) has exited.
    - If a job is being killed prior to exiting on it’s own a delay is inserted between the kill and then execution of the epilogue script.
  - For either Prologue or Epilogue, we need to be careful not to have them do things that could hang, exceed the timeout threshold or add significant waste to job turnaround.

Qmgr Sample Setup

```
#
# Create queues and set their attributes.
#
#
# Create and define queue kingspeak
#
create queue kingspeak
set queue kingspeak queue_type = Execution
set queue kingspeak resources_default.nodect = 1
set queue kingspeak resources_default.walltime = 00:15:00
set queue kingspeak kill_delay = 10
set queue kingspeak enabled = True
set queue kingspeak started = True
#
# Set server attributes.
#
set server scheduling = True
set server acl_hosts = kprm.ipoib.kingspeak.peaks
set server acl_hosts += *.peaks
```

```
set server managers = root@*.peaks
set server managers += sysadmin1@*.peaks
set server managers += sysadmin2@*.peaks
set server operators = helpdesk1@*.peaks
set server operators += helpdesk2@*.peaks
set server default_queue = kingspeak
set server log_events = 511
set server mail_from = kprm@chpc.utah.edu
set server query_other_jobs = True
set server resources_default.nodect = 1
set server resources_default.walltime = 00:15:00
set server scheduler_iteration = 600
set server node_check_rate = 150
set server tcp_timeout = 300
set server job_stat_rate = 45
set server poll_jobs = True
set server mom_job_sync = True
set server keep_completed = 30
set server log_keep_days = 30
set server job_force_cancel_time = 600
set server moab_array_compatible = True
```

Sample ‘nodes’ File

```
kp001.ipoib np=16 s2600 m65536 chpc
kp002.ipoib np=16 s2600 m65536 chpc
kp003.ipoib np=16 s2600 m65536 chpc
kp004.ipoib np=16 s2600 m65536 chpc
kp005.ipoib np=16 s2600 m65536 chpc
kp006.ipoib np=16 s2600 m65536 chpc
kp007.ipoib np=16 s2600 m65536 chpc
kp008.ipoib np=16 s2600 m65536 chpc
kp009.ipoib np=16 gpus=2 s2600 m65536 meteo
kp010.ipoib np=16 gpus=2 s2600 m65536 meteo
kp011.ipoib np=20 s2500 m131072 meteo
kp012.ipoib np=20 s2500 m131072 meteo
kp013.ipoib np=16 s2600 m65536 bmi
kp014.ipoib np=16 s2600 m65536 bmi
kp015.ipoib np=16 s2600 m65536 bmi
kp016.ipoib np=16 s2600 m65536 bmi
kp017.ipoib np=16 s2600 m65536 bmi
kp018.ipoib np=16 gpus=1 s2600 m32768 astro
kp019.ipoib np=16 gpus=1 s2600 m32768 astro
kp020.ipoib np=16 gpus=1 s2600 m32768 astro
kp021.ipoib np=16 gpus=1 s2600 m32768 astro
```

- First value is the name of the execution host.
  - If your execution hosts are multi-homed we find it best to use the hostname that you want jobs to use.
- Second value (‘np=’) is the number of processors torque is configured to manage for a given execution host.
- Third value (‘gpu=’, for those with GPUs) indicates the number of GPUs in the host.
- The rest are ‘node features’ that are just text strings associated with nodes that help with job submissions and documentation.

Sample pbs_mom ‘config’ File

```
$clienthost
kprm.ipoib.wasatch.peaks
$restricted          *.peaks
$remote_reconfig yes
$loglevel             3
$log_keep_days   30
$timeout              300
$job_output_file_umask  022
$usecp                *:/uufs/      /uufs/
$usecp                *:/scratch/    /scratch/
size[fs=/scratch/local]
```

- Define the RM host where pbs_server is run.
- Set ALCs to restrict which hosts pbs_mom will interact with.
- Allow for remote reconfig via momctl.
- Set log level and log retention window.
- Set the umask for stdout/stderr files. This is useful when your user support staff does not have administrative rights (root or otherwise).
- ‘usecp’ definitions to instruct pbs_mom to use ‘cp’ to move data to and from these spaces vs expecting to need to do a remote host to host transfer.
- Define the local path that pbs_mom will size and then report back to pbs_server to be used by jobs asking for specific local storage needs.

Sample pbs_mom ‘prologue’ File

```
#!/bin/bash
# Torque prologue & prologue.parallel Script

# Disable dynamic control of 'cpuspeed' which
# will push the Mhz of the CPUs.  Dynamic will
# be restored after the job exits.

( /bin/kill -s SIGUSR1 `/bin/ps -C cpuspeed -o
pid= |/usr/bin/xargs` ) >/dev/null 2>&1

exit 0
```

- Prologue & prologue.parallel are identical. Prologue is run on ‘mother superior’ and prologue.parallel are run on each ‘sister mom’ that is part of the job.
- We run ‘cpuspeed’ on our system to allow for processor frequency scaling. The goal is that while a node is idle that it will slow down the frequency to save some $ on power and cooling. Then when a job is run on a host, we tell ‘cpuspeed’ to max out the processor frequency so that they get top performance and consistent timings.

Sample pbs_mom ‘epilogue’ File

```
#!/bin/bash
# Torque epilogue & epilogue.parallel Script

# Clean all processes owned by user $2 of
# job $1 from the allocated nodes.

( /bin/kill -s SIGKILL `/bin/ps -u $2 -o pid=
|/usr/bin/xargs` ) >/dev/null 2>&1

# Restore dynamic control to 'cpuspeed'.

( /bin/kill -s SIGHUP `/bin/ps -C cpuspeed -o
pid= |/usr/bin/xargs` ) >/dev/null 2>&1

# Clean up any stale files in /dev/shm/.

( rm -f /dev/shm/* ) >/dev/null 2>&1

# Clean up /tmp/* to make sure there is space
# for the next job. (/tmp is a ram disk)

( find /tmp -user $2 -exec rm -rf  "{}" \; )
>/dev/null 2>&1

exit 0
```

- Epilogue & epilogue.parallel are identical. Epilogue is run on ‘mother superior’ and epilogue.parallel is run on each ‘sister mom’ in the job.
- First step we take in epilogue is to SIGKILL signal to all processes still on the host owned by the user. We do not allow node sharing so we are safe to simply find all processes by this user and send them the signal.
- We then send the SIGHUP signal to the ‘cpuspeed’ daemons to restore dynamic processor frequency scaling now that the job has ended.
- We clean up leftover files in /dev/shm and /tmp from the job to help give the next job a clean slate.

Torque Config & Setup Finalized

- Now that we have:
  - Configured pbs_server via ‘qmgr’ to match our needs.
  - Populated our pbs_server ‘nodes’ file to contain all of our resources.
  - Created and distributed our:
    - pbs_mom config file.
    - prologue & prologue.parallel files.
    - epilogue & epilogue.parallel files.
- We can now insure our changes are live and restart the services on our cluster hosts (trqauthd, pbs_mom, pbs_server) to bring our configurations live.
- We can also do a few basic tests to see that things look good:
  - Run ‘qmgr’ and issue a ‘print server’ to validate our changes.
  - Run ‘qstat -q’ to validate our queue definitions look right.
  - Run ‘pbsnodes -a’ to see a list of our resources and their states.

Torque User Commands

- **qstat**: command used to query the status of jobs, queues, or a batch server. Common usage:
  - ‘qstat -f $JOBID’: displays full status information for $JOBID.
  - ‘qstat -a’: displays all jobs.
- **pbsnodes**: command used to query the state of nodes. It can also be used to mark a node offline, or online as well as add a status note. Common usage:
  - ‘pbsnodes -a’: detailed list of all nodes, their state, job assignments, and status details.
  - ‘pbsnodes -o $HOST -N “Some note.”’: offlines $HOST and adds your note to it.
  - ‘pbsnodes -r $HOST -N “”’: Clears the offline state as well as any note on $HOST.
  - ‘pbsnodes -ln’: Lists all nodes that are down or offline including the note.

Torque User Commands

- **qsub**: command used to submit a job (interactive or script based). Common usage:
  - ‘qsub -I -l nodes=2:ppn=8,walltime=1:00:00 -A MyAccount -N TestJob’: Interactive job asking for 2 nodes, with 8 procs per node, for 1 hour walltime limit. MyAccount is the allocation account requested and the job name is TestJob.
  - ‘qsub JobScript.pbs’: batch job where all job requirements are defined in the job script, JobScript.pbs, as are the commands to run the desired workload.
- **qdel**: command used to delete one or more jobs. Jobs are only able to be deleted by the owning user, a batch operator, or batch administrator. Common usage:
  - ‘qdel $JOBID’ or ‘qdel $JOBLIST’: Delete job(s) from the queue regardless of their current state.

Sample Torque Job Script

```
#PBS -S /bin/bash
#PBS -l nodes=2:ppn=8,walltime=01:00:00
#PBS -A MyAccount
#PBS -M User1@MailDomain.com
#PBS -m abe
#PBS -N TestJob

# Set environment variables.
ScratchDir=”/scratch/global/$PBS_JOBID”
InputDir=”$HOME/DataSet/”
ResultsDir=”$HOME/Results/”

# Create job specific directory on network scratch space.
mkdir $ScratchDir

# Copy data files to job specific dir on network scratch space.
cp $InputDir/data.file $ScratchDir

# Execute application against data set and send stdout to log file.
mpirun -np 16 -machinefile $PBS_NODEFILE ~/bin/a.out
$ScratchDir/data.file >$ScratchDir/output.$PBS_JOBID.log

# Copy important data if any to home
cp $ScratchDir/output.$PBS_JOBID.log $ResultsDir/

# Clean up used scratch space
rm -rf $ScratchDir
```

- Torque job specifications are defined on lines that start with ‘#PBS’.
- The body of the script contains commands needed to execute our workload.
- Environment variables are set to make script writing easier. Examples are $PBS_JOBID & $PBS_NODEFILE.
- std{out,err} are spooled to ~/.pbs_spool during execution and moved to the submit dir upon completion.
- A script should have steps to clean up after itself.

Torque Management Commands

- **qrun:** command used to manually start a job (requires batch operator or batch administrator privilege to run). Common usage:
  - ‘qrun $JOBID’: manually instructs the batch system to attempt to start job $JOBID. Resource selection will simply meet the minimum requirements of the job. If resources are not available the command will fail.
- **qhold:** command used to place a hold on a job. There are 3 types of holds, User, Other(Operator), and System. Sufficient rights are needed to set Other and System holds. Common usage:
  - ‘qhold -h {u,o,s} $JOBID’: Places a hold of type User, Other, or System on $JOBID.
- **qrls:** command used to release a hold on a job. There are 3 types of holds, User, Other(Operator), and System. Sufficient rights are needed to remove Other and System holds. Common usage:
  - ‘qrls -h {u,o,s} $JOBID’: Releases a hold of type User, Other, or System on $JOBID.

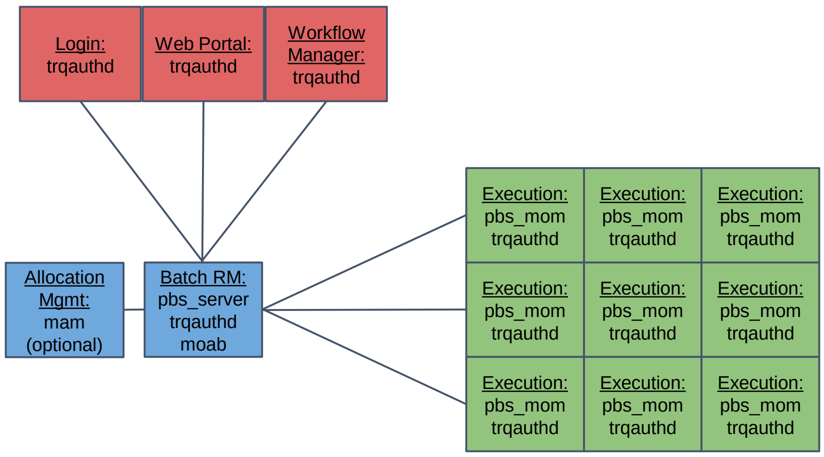
Torque Management Commands

- **qalter**: command used to alter the attributes of a job. Allows for job attributes and requirements to be adjusted. For running jobs a User is only able to lower the limits of resource requests. Common usage:
  - ‘qalter -l walltime=72:00:00 $JOBID’: This command could be used, with Admin or Operator access, to increase the walltime limit of a running job.
- **momctl**: command that allows local or remote shutdown, reconfiguration, diagnostics or querying of the pbs_mom daemon. Common usage:
  - ‘momctl -h $HOST -d 3’: prints level 3 diagnostics from pbs_mom on $HOST.
  - ‘momctl -h $HOST -r $SEVER-HOME/mom_priv/config’: reconfigure pbs_mom on $HOST after changes are made to the ‘config’ file.

Torque Logs & Accounting

- The 'pbs_server' daemon keeps a daily log of all activity in the $SERVER-HOME/server_logs directory.
- The 'pbs_mom' daemon keeps a daily log of all activity in the $SERVER-HOME/mom_logs/ directory.
- These logs contain information on communication between pbs_server and pbs_mom daemons as well as information on jobs as they enter the queue and are dispatched, run, and terminated.
- The 'trqauthd' daemon keeps daily logs in the $SERVER-HOME/client_logs/ directory. The trqauthd logs contain all successful and failed command authorizations and errors that occur.
- The 'pbs_server' daemon also keeps daily accounting logs in the $SERVER-HOME/server_priv/accounting/ directory. These records include events, timestamps, and information on resources requested and used.

Moab & Torque Diagram


Login: trqauthd
Web Portal: trqauthd
Workflow Manager: trqauthd
Allocation Mgmt: mam (optional)
Batch RM: pbs_server trqauthd moab
Execution: pbs_mom trqauthd
Execution: pbs_mom trqauthd
Execution: pbs_mom trqauthd
Execution: pbs_mom trqauthd
Execution: pbs_mom trqauthd
Execution: pbs_mom trqauthd
Execution: pbs_mom trqauthd
Execution: pbs_mom trqauthd
Execution: pbs_mom trqauthd

Moab Deployment

- Download Moab Workload Manager, part of Moab HPC Suite, from http://www.adaptivecomputing.com and follow the INSTALL document contained in the package. Moab is a commercially licensed product, but they do offer evaluations. The install guide for Moab is part of their online documentation on their web site.
- Customizations you might consider when you run 'configure':
  - --prefix=(your install path, or default /opt/moab/)
  - --with-homedir=(your MOABHOMEDIR, or default /opt/moab/)
  - --with-serverhost=$HOST (Specify the Moab server hostname)
  - --with-am=mam (Optional to use Moab Accounting Manager)
  - --with-torque=/Path/To/Torque/Install/ (If you used a non standard 'prefix' for the torque install)
  - --with-machine=$ClusterName (Names your cluster in moab.cfg)
  - --with-profile (installs /etc/profile.d/moab.[c]sh env files)
  - --with-init (use this on $serverhost to install the init script)

Moab Deployment

- After you select the appropriate configure options for your needs it's time for 'make' & 'make install'.
  - Note that the Moab download is a binary package. The configure, make, make install steps are simply used to perform the installation and offer install customizations.
  - Also note that if you do a local install then you will need to also install the Moab package to any system where you wish to use client commands.
- **'MOABHOMEDIR'** directory (defaults to /opt/moab/).
  - This directory holds the moab config, license, log, and stats files (with some organization to keep things sane).
  - If your install does not use a central location then only the $serverhost running Moab will contain the logs and stats.
- If other hosts need Moab installed locally now is the time to do that and then we are ready to start Moab on the $serverhost.

Moab Deployment Options

- **MAM**: Moab Accounting Manager is an accounting management system that allows for usage tracking and charging for resources. It acts much like a bank in which credits are deposited into funds. As resources are reserved and utilized, funds are encumbered, charged and usage is recorded.
  - For the scope of this presentation I will not go into the setup or configuration of MAM, but I will show how to tie Moab into a MAM service.
- You can also optionally download the following, though we will NOT cover these in this presentation:
  - **Moab Cluster Manager**: Java based GUI client for Moab.
  - **Moab Web Services**: RESTful interface for including Moab interaction into other custom services.
  - **Viewpoint**: Web based client portal for Moab.

Moab Components

- **Moab**: Moab Workload Manager (Scheduler).
  - Run on the Head Node or Resource Manager Node of a cluster.
  - Moab provides:
    - Scheduling and policy enforcement.
    - Allocation enforcement & accounting via MAM integration.
    - Detailed logging of all of its activities, including verbose server logging and event logging. Moab is also able to use system logging facilities.
    - Events & FairShare usage are store in the 'stats' dir in daily files.
    - Simulation mode that allows you to evaluate many scenarios that would otherwise be difficult to explore.
    - Extensive diagnostic tools.
- **$MOABHOMEDIR/:** Location of moab config, license, logs, stats, etc.

Sample ‘moab.cfg’

```
SCHEDCFG[kingspeak]             SERVER=kprm.wasatch.peaks:42559

ADMINCFG[1]                     USERS=root,Admin1,Admin2,Admin3 NAME=SystemAdmin
ADMINCFG[2]                     USERS=HelpDesk1,HelpDesk2,Helpdesk3 NAME=HelpDesk
ADMINCFG[3]                     USERS=ALL NAME=Users

RMCFG[kingspeak]                TYPE=PBS HOST=kprm.ipoib.wasatch.peaks
RMCFG[kingspeak]                SUBMITCMD=/usr/local/bin/qsub
RMCFG[kingspeak]                JOBEXTENDDURATION=00:15:00,48:00:00 TIMEOUT=60
RMCFG[kingspeak]                SYNCJOBID=TRUE FLAGS=PROXYJOBSUBMISSION
JOBIDFORMAT=INTEGER

AMCFG[chpc]                               TYPE=MAM HOST=alloc.helmsdeep PORT=7113
TIMEOUT=00:01:00
AMCFG[chpc]                               FLUSHINTERVAL=NONE FLAGS=STRICTQUOTE
FALLBACKQOS=FREECYCLE
AMCFG[chpc]                               STARTFAILUREACTION=RETRY,RETRY
CHARGEPOLICY=DEBITALLBLOCKED

USEDATABASE                     INTERNAL

DEFAULTDOMAIN          ipoib.kingspeak.peaks
DEFAULTCLASSLIST                kingspeak

LOGFILE                moab.log
LOGFILEMAXSIZE         500000000
LOGFILEROLLDEPTH                50
LOGLEVEL               7
```

Sample ‘moab.cfg’

```
# Prioritization Weights
SERVICEWEIGHT          1
QUEUETIMEWEIGHT                 5
XFACTORWEIGHT          1000
RESWEIGHT              1
PROCWEIGHT             10
TARGETWEIGHT           1
TARGETXFACTORWEIGHT             2
TARGETQUEUETIMEWEIGHT           2
FSWEIGHT               1500
FSACCOUNTWEIGHT                 1
FSQOSWEIGHT            100
FSUSERWEIGHT           10
FSGROUPWEIGHT                   1
CREDWEIGHT             1
QOSWEIGHT              1

# FairShare (2 week window, 7 intervals of 48 hours)
FSPOLICY               DEDICATEDPS
FSDEPTH                7
FSINTERVAL             2:00:00:00
FSDECAY                0.80
```

Sample 'moab.cfg'

```
# Backfill & General Policies
BACKFILLPOLICY              FIRSTFIT
BFCHUNKSIZE                 64
BFCHUNKDURATION                      00:30:00
MISSINGDEPENDENCYACTION            RUN
NODEAVAILABILITYPOLICY               DEDICATED
NODEACCESSPOLICY                     SINGLEJOB
JOBNODEMATCHPOLICY                   EXACTPROC

#Reservation Policies
RESERVATIONPOLICY                    CURRENTHIGHEST
RESERVATIONDEPTH[DEFAULT]       10
RESERVATIONQOSLIST[GENERAL]     GENERAL
RESERVATIONDEPTH[GENERAL]       10
RESERVATIONQOSLIST[FREECYCLE]   FREECYCLE
RESERVATIONDEPTH[FREECYCLE]     0
RESERVATIONQOSLIST[meteo]       meteo
RESERVATIONDEPTH[meteo]         20
RESERVATIONQOSLIST[bmi]              bmi
RESERVATIONDEPTH[bmi]                20
RESERVATIONQOSLIST[astro]       astro
RESERVATIONDEPTH[astro]         20
RESERVATIONQOSLIST[oguest]      oguest
RESERVATIONDEPTH[oguest]        20
```

Sample 'moab.cfg'

```
# Standing Reservations

# General (CHPC) 36 Node Reservation
SRCFG[General_36]                    QOSLIST=&GENERAL+,&long+,&FREECYCLE+
SRCFG[General_36]        HOSTLIST=kp0[0-2][0-9].ipoib,kp03[0-6].ipoib
SRCFG[General_36]        PERIOD=INFINITY

# Meteo 15 Node Reservation
SRCFG[Meteo_15]          ACCOUNTLIST=&meteo-kp+,owner-guest QOSLIST=&meteo+,oguest
SRCFG[Meteo_15]          HOSTLIST=kp03[7-9].ipoib,kp04[0-9].ipoib,kp05[0-1].ipoib
SRCFG[Meteo_15]          PERIOD=INFINITY

# BMI 6 Node Reservation
SRCFG[BMI_6]             ACCOUNTLIST=&bmi-kp+,owner-guest QOSLIST=&bmi+,oguest
SRCFG[BMI_6]             HOSTLIST=kp05[2-7].ipoib
SRCFG[BMI_6]             PERIOD=INFINITY

# Astro 28 Nodes Reservation
SRCFG[ASTRO_28]          ACCOUNTLIST=&astro-kp+,owner-guest QOSLIST=&astro+,&astro-fast+,oguest
SRCFG[ASTRO_28]          HOSTLIST=kp05[8-9].ipoib,kp0[6-7][0-9].ipoib,kp08[0-5].ipoib
SRCFG[ASTRO_28]          PERIOD=INFINITY
```

Sample 'moab.cfg'

```
# Preemption Settings
PREEMPTPOLICY                  CANCEL
PREEMPTSEARCHDEPTH             1024
JOBMAXPREEMPTPERITERATION      1024
PREEMPTJOBCOUNT                1024
IGNOREPREEMPTEEPRIORITY        TRUE
GUARANTEEDPREEMPTION                          TRUE
JOBPREEMPTMINACTIVETIME        00:01:00
DISABLESAMECREDPREEMPTION      QOS,USER

# Global Job Limits
SYSTEMMAXJOBWALLTIME=259200

# NODECFG policies for nodes
NODECFG[DEFAULT] MAXLOAD=1.0 ENABLEPROFILING=TRUE

# ACCOUNT
ACCOUNTCFG[DEFAULT] QDEF=GENERAL QLIST=GENERAL ENABLEPROFILING=TRUE
ACCOUNTCFG[owner-guest] QDEF=oguest QLIST=^oguest
ACCOUNTCFG[meteo-kp] QDEF=meteo QLIST=^meteo
ACCOUNTCFG[bmi-kp] QDEF=bmi QLIST=^bmi
ACCOUNTCFG[astro-kp] QDEF=astro QLIST=^astro
```

Sample 'moab.cfg'

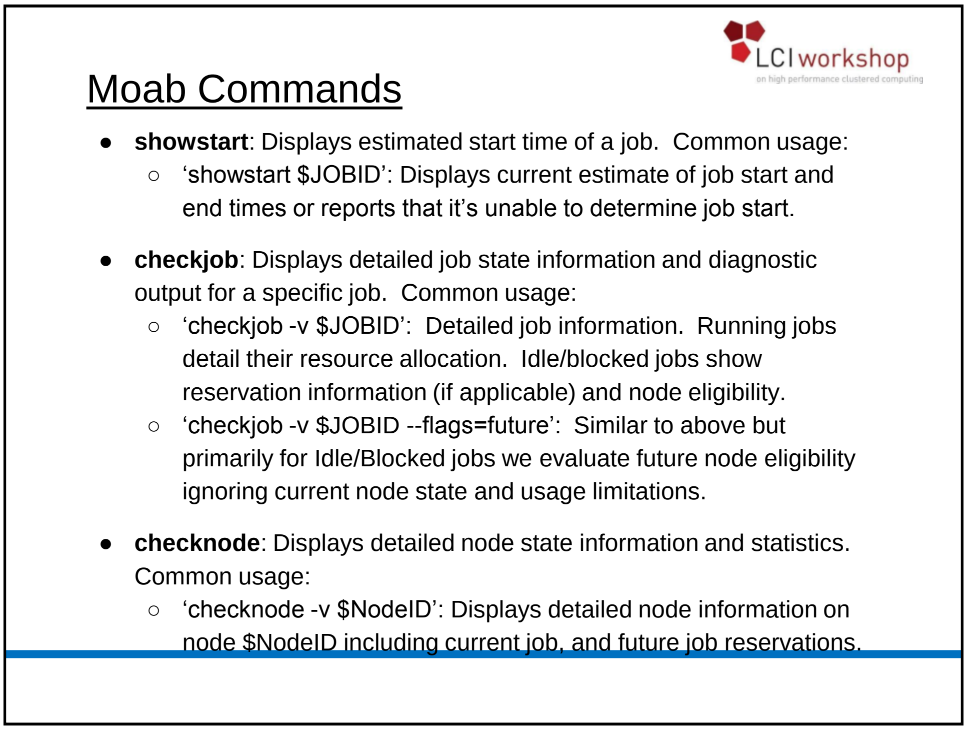
```
# QOS Definitions
QOSCFG[DEFAULT] PRIORITY=250000 ENABLEPROFILING=TRUE
# Freecycle jobs are limited 25% of general nodes per user under competition.
QOSCFG[FREECYCLE] PRIORITY=1 MAXNODE[USER]=8,32 QFLAGS=PREEMPTEE
QFLAGS=USERESERVED:General_36
# Long jobs are limited to 2 nodes total at any given time.
QOSCFG[long] PRIORITY=250000 MAXPROC=32 OMAXJWC=1209600 QFLAGS=PREEMPTOR
QFLAGS=USERESERVED:General_36
# og (Owner Guest QOS)
QOSCFG[oguest] PRIORITY=1 QFLAGS=PREEMPTEE
QOSCFG[GENERAL] PRIORITY=250000 QFLAGS=PREEMPTOR QFLAGS=USERESERVED:General_36
QOSCFG[meteo] PRIORITY=500000 QFLAGS=USERESERVED:Meteo_15 OMAXJWC=1209600
QFLAGS=PREEMPTOR
QOSCFG[bmi] PRIORITY=500000 QFLAGS=USERESERVED:BMI_6 OMAXJWC=2592000
QFLAGS=PREEMPTOR
QOSCFG[astro] PRIORITY=500000 QFLAGS=USERESERVED:ASTRO_28 MAXNODE=27
OMAXJWC=1209600 QFLAGS=PREEMPTOR
QOSCFG[astro-fast] PRIORITY=500000 QFLAGS=USERESERVED:ASTRO_28 MAXNODE=1
OMAXJWC=1209600 QFLAGS=PREEMPTOR MEMBERULIST=User4,User8,User73

# CLASS
CLASSCFG[DEFAULT] ENABLEPROFILING=TRUE
# GROUP
GROUPCFG[DEFAULT] ENABLEPROFILING=TRUE
# USER defaults
USERCFG[DEFAULT] FSTARGET=5.00+ MAXIJOB=5 ENABLEPROFILING=TRUE
```

Moab Config & Setup Finalized

- Now that we have:
  - Installed Moab, environment values, and init script.
  - Configured 'moab.cfg' to suite our our needs.
- We are now ready to start Moab to have our changes take effect.
- We can also do a few basic tests to see that things look good:
  - Run 'mdiag -C $MOABHOMEDIR/etc/moab.cfg' to check for any obvious spelling and or syntax errors.
  - Run 'mdiag -R' to see Moab's connection status with Torque and MAM.
  - Run 'showstats -v' to display general statistics of the cluster.

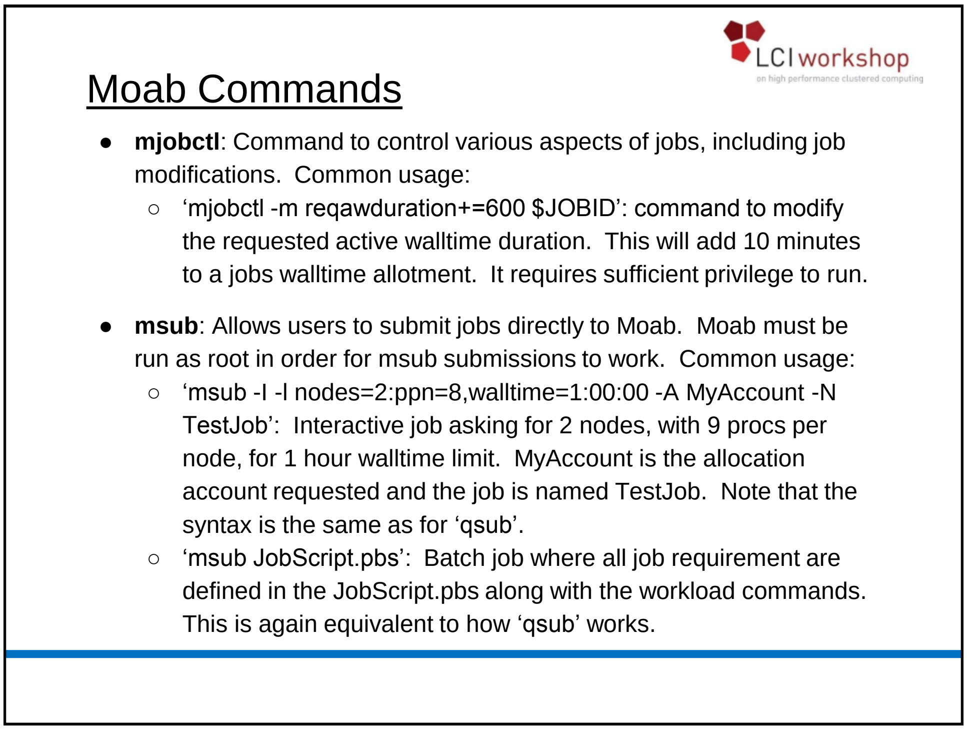
Moab Commands

- **showq**: Displays info about running, eligible, and blocked jobs. Common usage:
  - 'showq': Simple list of all jobs segmented in running, eligible, and blocked groups. Also reports on the current cluster usage.
  - 'showq -{r,i,b}': Lists only 'Running', 'Idle', or 'Blocked' jobs.
  - 'showq -w {user,group,acct,class,qos}=<VAL>': Displays only jobs associated with the the specified constraint.
- **mshow -a**: Command for querying available system resources. This is useful for users to test different resource requirements or job attributes and see how they affect job start time.   Common usage:
  - 'mshow -a -w <ATTR>=<VAL>[,<ATTR>=<VAL>]... --flags=future': Displays resource availability, immediate & future, based on the current queue state. Where ATTR can be account, group, qos, user, duration, minnodes, minprocs, nodefeature, etc…



Moab Commands

- **showstart**: Displays estimated start time of a job. Common usage:
  - 'showstart $JOBID': Displays current estimate of job start and end times or reports that it's unable to determine job start.
- **checkjob**: Displays detailed job state information and diagnostic output for a specific job. Common usage:
  - 'checkjob -v $JOBID': Detailed job information. Running jobs detail their resource allocation. Idle/blocked jobs show reservation information (if applicable) and node eligibility.
  - 'checkjob -v $JOBID --flags=future': Similar to above but primarily for Idle/Blocked jobs we evaluate future node eligibility ignoring current node state and usage limitations.
- **checknode**: Displays detailed node state information and statistics. Common usage:
  - 'checknode -v $NodeID': Displays detailed node information on node $NodeID including current job, and future job reservations.



Moab Commands

- **mjobctl**: Command to control various aspects of jobs, including job modifications. Common usage:
  - 'mjobctl -m reqawduration+=600 $JOBID': command to modify the requested active walltime duration. This will add 10 minutes to a jobs walltime allotment. It requires sufficient privilege to run.
- **msub**: Allows users to submit jobs directly to Moab. Moab must be run as root in order for msub submissions to work. Common usage:
  - 'msub -I -l nodes=2:ppn=8,walltime=1:00:00 -A MyAccount -N TestJob': Interactive job asking for 2 nodes, with 9 procs per node, for 1 hour walltime limit. MyAccount is the allocation account requested and the job is named TestJob. Note that the syntax is the same as for 'qsub'.
  - 'msub JobScript.pbs': Batch job where all job requirement are defined in the JobScript.pbs along with the workload commands. This is again equivalent to how 'qsub' works.

Moab Commands

- **showstats**: Command shows various accounting and resource usage statistics for the cluster. Accepts '-t <timespec> flag. Common usage:
  - 'showstats -v': Displays general cluster statistics.
  - 'showstats -{a,g,q,u} [ID]': Displays account, group, qos, or user active and historical statistics. If an 'ID' is specified then it will show only for that ID.
  - 'showstats -f <statisticstype>': Displays a table of selected stat such as JOBCOUNT, AVGQTIME, BFCOUNT, WCACCURACY, etc.
- **mschedctl**: Command that controls or queries various aspects of scheduler behavior. Common usage:
  - 'mschedctl -l config': Displays current active configuration.
  - 'mschedctl -m config LOGLEVEL 7': Change a specific parameter. Note this change is not persistent between restarts.
  - 'mschedctl -{s,p,k,r,R}': stop,pause,kill,resume,Restart Moab.

Moab Commands

- **showres**: Lists reservations currently in place. Common usage:
  - 'showres': Display all system, job, and standing reservations.
  - 'showres -n $ResID': Display list of nodes reserved by $ResID.
- **mrsvctl**: Command to create, modify, query and release reservations. Common usage:
  - 'mrsvctl -c -h ALL -s { <ABSTIME> | <RELTIME> }': Create a 'system' reservation to drain the cluster for the purpose of a downtime or other such outage.
  - 'mrsvctl -c -a user==User1 -f chpc -t 4 -d 8:00:00 -s +24:00:00': Create a reservation accessible only by User1 that has 4 'chpc' nodes, an 8 hour duration and will start in 24 hours.
  - 'mrsvctl -r $ResID': Release reservation $ResID.
- **showstate**: Displays summary state of the cluster. Common usage:
  - 'showstate --flags=noauto': Displays state of cluster including "warning messages" at the end that are useful in finding issues.

Moab Commands

- **showhist.moab.pl**: A script that displays historical job information. Included in this information are queue time, start time, end time, resources allocated, and job credentials. Common usage:
  - ‘showhist.moab.pl -{a,g,q,u,j} {ID}’: Displays completed job info contrained to the $ID chosen. The flag indicates account, group, qos, user, or job id.
- **mdiag**: Command used to display information about various aspects of the cluster and the results of internal diagnostic tests. There are a lot of possible flags this command can take, more than I can try to represent here. “Some” common usage:
  - ‘mdiag -f’: Displays fairshare usage.
  - ‘mdiag -p’: Displays job priority details for current eligible jobs.
  - ‘mdiag -r $ResID’: Displays reservation details.
  - ‘mdiag -{S,R}’: Displays Sched or Resource Mgr info & status.
  - ‘mdiag -b’: Displays diagnostic information about blocked jobs.

Moab Policies, Tuning & Adv Config

- **Job Prioritization**: Priority is calculated from multiple groups of job related factors such as Credentials, FairShare, Service Level, Target Service Level, and Requested Resources.
- ‘mdiag -p’: Running this command will show us how our current settings are affecting priority scores of the current eligible idle jobs.

```
diagnosing job priority information (partition: ALL)

Job              PRIORITY*  Cred(  QOS)   FS( User:Group:Accnt:  QOS)  Serv(QTime:XFctr)  Targ(QTime:XFctr)  Res( Proc)
       Weights   --------     1(    1)  1500(  10:    1:    1:  100)     1(    5: 1000)     1(    2:    2)     1(   10)

92278              572177  87.4(50000)   7.6(  2.9:  0.0:  0.0:  0.0)   4.8(5101.:  2.2)   0.0(  0.0:  0.0)   0.2( 96.0)
97350              514129  97.3(50000)   2.1(  0.7:  0.0:  0.0:  0.0)   0.6(369.0:  1.1)   0.0(  0.0:  0.0)   0.1( 64.0)
89586              508531  98.3(50000)   0.0(  0.0:  0.0:  0.0:  0.0)   1.6(1382.:  1.5)   0.0(  0.0:  0.0)   0.0( 16.0)
96537              330249  75.7(25000)  22.7(  5.0:  0.0:  0.0:  0.0)   1.2(520.0:  1.4)   0.0(  0.0:  0.0)   0.4(128.0)
99147              306082  81.7(25000)  17.6(  3.6:  0.0:  0.0:  0.0)   0.5(118.0:  1.1)   0.0(  0.0:  0.0)   0.2( 64.0)
96288              259990  96.2(25000)   0.0(  0.0:  0.0:  0.0:  0.0)   2.9(1229.:  1.3)   0.0(  0.0:  0.0)   1.0(256.0)
96431               78159   0.0(  1.0)  92.5(  4.8:  0.0:  0.0:  0.0)   6.8(680.0:  1.9)   0.0(  0.0:  0.0)   0.6( 48.0)
98915               77718   0.0(  1.0)  95.3(  4.9:  0.0:  0.0:  0.0)   4.5(274.0:  2.1)   0.0(  0.0:  0.0)   0.2( 16.0)
```

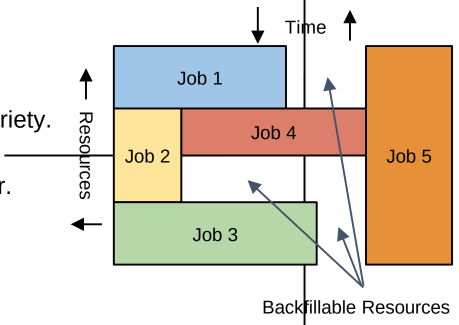
Moab Policies, Tuning & Adv Config

- **Fairshare**: Allows historical resource utilization to be incorporated into job feasibility and priority decisions. This allows admins to set utilization targets for users, groups, accounts, classes, and QoS levels. FS targets can be set as a ‘ceiling’(-), ‘floor’(+) or ‘target’().
- ‘mdiag -f’: Command displays current FS settings & utilization.

```
Depth: 7 intervals   Interval Length: 2:00:00:00   Decay Rate: 0.80

FS Policy: DEDICATEDPS
System FS Settings:  Target Usage: 0.00

FSInterval      %    Target     0       1       2       3       4       5       6
FSWeight     ------- -------  1.0000  0.8000  0.6400  0.5120  0.4096  0.3277  0.2621
TotalUsage    100.00 ------- 62467.4 91256.7 101434.1 100828.2 68688.2 75968.3 61744.9

USER
-------------
User1*                2.23   5.00+   4.76    5.05    0.78 ------- ------- ------- -------
User2*                1.73   5.00+   2.32    1.65 -------   2.42    0.65    5.07    1.33
User3*                0.12   5.00+ ------- -------   0.00    0.67    0.11    0.01    0.01
User4         9.33   5.00+   6.20   11.25   12.93    7.25    4.29    8.10   15.63
User5*                1.99   5.00+ ------- ------- -------   5.32   12.08    1.00 -------
User6                 9.12   5.00+   7.46    6.04    6.90   10.88   17.67   13.49   11.04
User7*                4.55   5.00+   8.41    0.83    4.03    8.12    3.35    1.77    3.45
User8                20.59   5.00+  19.92   37.98   14.59   20.55   16.14    5.36 -------
```

Moab Policies, Tuning & Adv Config

- **Backfill**: Backfill is the ability for the scheduler to evaluate the current running jobs, those in the queue that will be started soon and also be aware of the resources left unused (wasted) due to the layout of the jobs on the resources. It then evaluates the idle jobs in the queue to find a job that will fit into the gap. A job can’t be backfilled if it would disrupt a reservation for a higher priority job. Other considerations:
  - Wallclock accuracy.
  - Lack of wallclock variety.
  - Serial jobs take over.
  - Backfill Chunking.



Moab Policies, Tuning & Adv Config

- **Quality of Service**: Allows for special treatment to classes of jobs, users, groups, etc. A QoS can be thought of as a container of special privileges ranging from fairness policy exemptions, to special job prioritization, to special functionality. One also defines users, groups, and accounts that can access these QoS privileges. For example:

```
# QOS Definitions
QOSCFG[FREECYCLE] PRIORITY=1 MAXNODE[USER]=8,32 QFLAGS=PREEMPTEE
QFLAGS=USERESERVED:General_36
QOSCFG[long] PRIORITY=250000 MAXPROC=32 OMAXJWC=1209600 QFLAGS=PREEMPTOR
QFLAGS=USERESERVED:General_36
QOSCFG[oguest] PRIORITY=1 QFLAGS=PREEMPTEE
QOSCFG[GENERAL] PRIORITY=250000 QFLAGS=PREEMPTOR QFLAGS=USERESERVED:General_36
QOSCFG[meteo] PRIORITY=500000 QFLAGS=USERESERVED:Meteo_15 OMAXJWC=1209600
QFLAGS=PREEMPTOR
QOSCFG[bmi] PRIORITY=500000 QFLAGS=USERESERVED:BMI_6 OMAXJWC=2592000
QFLAGS=PREEMPTOR
QOSCFG[astro] PRIORITY=500000 QFLAGS=USERESERVED:ASTRO_28 MAXNODE=27
OMAXJWC=1209600 QFLAGS=PREEMPTOR
QOSCFG[astro-fast] PRIORITY=500000 QFLAGS=USERESERVED:ASTRO_28 MAXNODE=1
OMAXJWC=1209600 QFLAGS=PREEMPTOR MEMBERULIST=User4,User8,User73
QOSCFG[genome1] PRIORITY=500000 QFLAGS=USERESERVED:Genome1-2_4 OMAXJWC=1209600
QFLAGS=PREEMPTOR
QOSCFG[genome2] PRIORITY=500000 QFLAGS=USERESERVED:Genome1-2_4 OMAXJWC=1209600
QFLAGS=PREEMPTOR
```

Moab Policies, Tuning & AdvConfig

- **Preemption**: Sites possess workloads of varying importance, and users may want to run jobs with higher priorities before jobs with lower priorities. This can be done by using preemption. Preemption is simply the process by which a higher-priority job can take the place of a lower-priority job. We have 2 usage cases for preemption:
  - 'General' jobs with allocation can preempt 'general' jobs that are out of allocation. This provides opportunity for allocations to be used, while not needing to block users who have run out.
  - 'Owner' jobs can always preempt 'guest' jobs that are using the owner's resources. Helps prevent waste of idle resources.

```
PREEMPTPOLICY                CANCEL
IGNOREPREEMPTEEPRIORITY      TRUE
GUARANTEEDPREEMPTION         TRUE
JOBPREEMPTMINACTIVETIME      00:01:00
DISABLESAMECREDPREEMPTION    QOS,USER

QOSCFG[FREECYCLE] PRIORITY=1 QFLAGS=PREEMPTEE QFLAGS=USERESERVED:General_36
QOSCFG[GENERAL] PRIORITY=250000 QFLAGS=PREEMPTOR QFLAGS=USERESERVED:General_36
```

Moab Policies, Tuning & Adv Config

- **Advanced Reservations**: A mechanism by which Moab guarantees the availability of a set of resources at a particular time. Each reservation consists of three major components: (1) a set of resources, (2) a time frame, and (3) an access control list. Here are 3 examples of Advanced Reservations:
  - **One-TIme Reservations**: A reservation to meet the need of a single instance. Such as a request for a set of nodes to be set aside for a specific user for a window of time. For example:
    - Create a reservation for User1 on 4 nodes for 1 day to start on 12/25/14 at 8:00am:
      - ‘mrsvctl -c -a user==User1 -h ‘kp00[1-4].ipoib’ -d 24:00:00 -s 8:00:00_12/25/14 -n User1_DAT’
    - Submit a job targeting the ‘User1_DAT’ reservation:
      - ‘msub -l flags=ADVRES:User1_DAT.1 JobScript.pbs’

Moab Policies, Tuning & Adv Config

- **Standing Reservations**: Provide a superset of the capabilities typically found in a batch queuing system's class or queue architecture. Standing reservations allow for greater flexibility and efficiency than is typically found in a normal queue management system. For example our ‘moab.cfg’ could have:

```
# Meteo 15 Node Reservation
SRCFG[Meteo_15]          ACCOUNTLIST=&meteo-kp+,owner-guest QOSLIST=&meteo+,oguest
SRCFG[Meteo_15]          HOSTLIST=kp03[7-9].ipoib,kp04[0-9].ipoib,kp05[0-1].ipoib
SRCFG[Meteo_15]          PERIOD=INFINITY

#  Account Defintions
ACCOUNTCFG[owner-guest] QDEF=oguest QLIST=^oguest
ACCOUNTCFG[meteo-kp] QDEF=meteo QLIST=^meteo

# QOS Definitions
QOSCFG[oguest] PRIORITY=1 QFLAGS=PREEMPTEE
QOSCFG[meteo] PRIORITY=500000 QFLAGS=USERESERVED:Meteo_15 OMAXJWC=1209600
QFLAGS=PREEMPTOR
```

  - The ‘PERIOD’ can be DAY, WEEK, or INFINITY. Also ‘DAYS’, ‘STARTTIME’, and ‘ENDTIME’ can be set.

Moab Policies, Tuning & Adv Config

- **Administrative Reservations**: These are similar in nature to One-Time reservations but for the purpose of administrative tasks. Useful to drain a cluster for a scheduled downtime or to clear a set of nodes for some task that we need the resources clear of any running jobs starting at a specific time. For example:
  - Set a reservation for a scheduled downtime and verify.
    - 'mrsvctl -c -D 'Fall quarterly downtime reservation 10/14/14' -n FallDowntime -h ALL -s 7:00:00_10/14/14'

```
NOTE:    reservation FallDowntime.63 created
```

- 'showres FallDowntime.63'

```
ReservationID      Type S      Start        End    Duration    N/P    StartTime

FallDowntime.63     User - 97:17:20:37    INFINITY    INFINITY  157/2648 Tue Oct 14 07:00:00

1 reservation located
```

Diagnostics & Troubleshooting

Keep things simple when you find yourselves trying to understand when something doesn't appear to be working right. Familiarize yourselves with your systems so that you can navigate the commands, information & logs.

- Jobs & Queue:
  - 'showq -[r,i,b]' displays the state of the job in the queue.
    - Key points: State, priority, and reservation indication.
  - 'checkjob -v $JOBID' details job state information from Moab.
    - Key points: Credentials, job requirements, reservation details, node availability, system messages, and state.
  - 'qstat -f $JOBID' details job state information from Torque.
    - Key points: Credentials, job requirements, env and state.
  - Look at the users job script (if non-interactive).
  - Look at the stdout & stderr files that are spooled for running jobs.

Diagnostics & Troubleshooting

- Resource Manager & Scheduler:
  - pbs_server logs detail the interactions between the pbs_server, Moab, user commands, and pbs_mom.
  - pbs_server accounting files contain job stage events, resource allocations, and utilization records.
  - ‘tracejob -q $JOBID’ parses pbs_server torque logs/accounting to show job event information.
  - moab logs detail out interactions between moab and torque. They also provide detailed records of the scheduling iteration.
  - A copy of the job script (non-interactive) is stored on the server while the job is in queue.
  - ‘showconfig’, ‘changeparam’, and ‘mdiag -C’ help you evaluate your ‘moab.cfg’ syntax, settings and make in flight changes.

Diagnostics & Troubleshooting

- Execution Nodes:
  - ‘showstate’ also displays nodes in unexpected states.
  - ‘checknode -v $NODE’ displays detailed Moab node info.
  - ‘pbsnodes -a $NODE’ displays detailed Torque node info.
  - ‘momctl -d3 -h $NODE’ a pbs_mom diagnostic command that prints detailed information on pbs_mom config and status.
  - pbs_mom log files detail the interactions between the pbs_mom(s), Moab, and pbs_server.
  - ‘tracejob -q $JOBID’ parses pbs_mom torque logs to show job & pbs_mom event information.
  - A copy of the job script (non-interactive) is stored on ‘mother superior’ while the job is in queue.

Troubleshooting Example #1

Why wont job 110263 start? Job 110263 has been in the queue longer than the other jobs, my job should have already started!

- First lets look at the 'idle' queue status: 'showq -i'

```
eligible jobs----------------------
JOBID          PRIORITY  XFACTOR  Q  USERNAME    GROUP  PROCS     WCLIMIT     CLASS      SYSTEMQUEUETIME

115768*          337456    11.7 GE  u0637796   steele     20    1:00:00 kingspeak   Fri Jul 25 13:32:01
115769*          337452    11.7 GE  u0637796   steele     20    1:00:00 kingspeak   Fri Jul 25 13:32:01
115767*          337447    11.7 GE  u0637796   steele     20    1:00:00 kingspeak   Fri Jul 25 13:32:01
115625*          337444    11.7 GE  u0637796   steele     20    1:00:00 kingspeak   Fri Jul 25 13:31:53
115626*          337438    11.7 GE  u0637796   steele     20    1:00:00 kingspeak   Fri Jul 25 13:31:53
116302*          335087    11.7 GE  u0637796   steele     20    1:00:00 kingspeak   Fri Jul 25 13:33:08
116308*          334359    11.7 GE  u0637796   steele     20    1:00:00 kingspeak   Fri Jul 25 13:33:08
116306*          331380    11.7 GE  u0637796   steele     20    1:00:00 kingspeak   Fri Jul 25 13:33:08
116311*          331342    11.7 GE  u0637796   steele     20    1:00:00 kingspeak   Fri Jul 25 13:33:08
116305*          328634    11.7 GE  u0637796   steele     20    1:00:00 kingspeak   Fri Jul 25 13:33:08
111794           302109     2.4 GE  u0678934   detar     96    10:00:00 kingspeak   Fri Jul 25 10:37:22
111749           301392     1.6 GE  u0794535     zpu    112  1:00:00:00 kingspeak   Fri Jul 25 09:47:43
111760           296910     2.4 GE  u0678934   detar     96    10:00:00 kingspeak   Fri Jul 25 10:36:14110263            281417
2.2 GE  u0050476   bedrov    256  3:00:00:00 kingspeak   Tue Jul 22 07:27:12
111305           271252     1.9 GE  u0050476   bedrov     64  3:00:00:00 kingspeak   Wed Jul 23 09:43:01
111327           271106     1.9 GE  u0050476   bedrov     64  3:00:00:00 kingspeak   Wed Jul 23 10:11:11
```

Troubleshooting Example #1

- Lets look at the job state information: 'checkjob 110263'

```
AName: mdex
State: Idle
Creds:  user:u0050476  group:bedrov  account:bedrov  class:kingspeak  qos:GENERAL
WallTime:   00:00:00 of 3:00:00:00
BecameEligible: Tue Jul 22 07:27:12
SubmitTime: Tue Jul 22 07:27:12
  (Time Queued  Total: 3:17:10:32  Eligible: 3:17:10:32)

TemplateSets:  DEFAULT
NodeMatchPolicy: EXACTPROC
Total Requested Tasks: 256

Req[0]  TaskCount: 256  Partition: ALL
Procs ==16  Memory >= 0  Disk >= 0  Swap >= 0

SystemID:   kingspeak
SystemJID:  110263
Notification Events: JobStart,JobEnd,JobFail  Notification Address: user@domain

BypassCount:    7
Flags:          ADVRES:General_44,RESTARTABLE,PREEMPTOR
Attr:           checkpoint
StartPriority:  281543kp007.ipoib           available: 16 tasks supportedNOTE:  job req cannot run in partition kingspeak (available procs do
not meet requirements : 16 of 256 procs found)
idle procs: 1224  feasible procs:  16

Node Rejection Summary: [CPU: 34][State: 129][Reserved: 33]
```

Troubleshooting Example #1

- Now lets understand the current job prioritization: ‘mdiag -p’

```
diagnosing job priority information (partition: ALL)

Job              PRIORITY*   Cred( QOS)    FS( User:Group:Accnt:  QOS)  Serv(QTime:XFctr)  Targ(QTime:XFctr)   Res( Proc)
         Weights  --------     1(    1)  1500(  10:    1:    1:  100)     1(    5: 1000)     1(    2:    2)     1(   10)

115768             337514   74.1(25000)  22.1(  5.0:  0.0:  0.0:  0.0)   3.7(537.0: 10.0)   0.0(  0.0:  0.0)   0.1( 20.0)
115769             337510   74.1(25000)  22.1(  5.0:  0.0:  0.0:  0.0)   3.7(537.0: 10.0)   0.0(  0.0:  0.0)   0.1( 20.0)
115767             337501   74.1(25000)  22.1(  5.0:  0.0:  0.0:  0.0)   3.7(536.0:  9.9)   0.0(  0.0:  0.0)   0.1( 20.0)
115625             337498   74.1(25000)  22.1(  5.0:  0.0:  0.0:  0.0)   3.7(536.0:  9.9)   0.0(  0.0:  0.0)   0.1( 20.0)
115626             337497   74.1(25000)  22.1(  5.0:  0.0:  0.0:  0.0)   3.7(536.0:  9.9)   0.0(  0.0:  0.0)   0.1( 20.0)
116302             335154   74.6(25000)  22.3(  5.0:  0.0:  0.0:  0.0)   3.1(428.0:  8.1)   0.0(  0.0:  0.0)   0.1( 20.0)
116308             334427   74.8(25000)  22.3(  5.0:  0.0:  0.0:  0.0)   2.9(394.0:  7.6)   0.0(  0.0:  0.0)   0.1( 20.0)
116306             331448   75.4(25000)  22.5(  5.0:  0.0:  0.0:  0.0)   2.0(257.0:  5.3)   0.0(  0.0:  0.0)   0.1( 20.0)
116311             331410   75.4(25000)  22.5(  5.0:  0.0:  0.0:  0.0)   2.0(255.0:  5.3)   0.0(  0.0:  0.0)   0.1( 20.0)
116305             328702   76.1(25000)  22.7(  5.0:  0.0:  0.0:  0.0)   1.2(130.0:  3.2)   0.0(  0.0:  0.0)   0.1( 20.0)
111794             302138   82.7(25000)  14.8(  3.0:  0.0:  0.0:  0.0)   2.1(817.0:  2.4)   0.0(  0.0:  0.0)   0.3( 96.0)
111749             301390   82.9(25000)  14.7(  3.0:  0.0:  0.0:  0.0)   2.0(866.0:  1.6)   0.0(  0.0:  0.0)   0.4(112.0)
111760             296938   84.2(25000)  15.1(  3.0:  0.0:  0.0:  0.0)   0.4( 37.0:  1.1)   0.0(  0.0:  0.0)   0.3( 96.0)110263             281428
88.8(25000)   0.0(  0.0:  0.0:  0.0:  0.0)  10.3(5327.:  2.2)   0.0(  0.0:  0.0)   0.9(256.0)
111305             271264   92.2(25000)   0.0(  0.0:  0.0:  0.0:  0.0)   7.6(3751.:  1.9)   0.0(  0.0:  0.0)   0.2( 64.0)
111327             271117   92.2(25000)   0.0(  0.0:  0.0:  0.0:  0.0)   7.6(3723.:  1.9)   0.0(  0.0:  0.0)   0.2( 64.0)

Percent Contribution  --------   78.8( 78.8)  17.4( 17.4:  0.0:  0.0:  0.0)   3.6(  1.8:  1.8)   0.0(  0.0:  0.0)   0.2(  0.2)
```

Troubleshooting Example #1

- Looking at the current fairshare user usage: ‘mdiag -f -o user’

```
FairShare Information

Depth: 7 intervals   Interval Length: 2:00:00:00   Decay Rate: 0.80

FS Policy: DEDICATEDPS
System FS Settings:  Target Usage: 0.00

FSInterval        %     Target       0        1        2        3        4        5        6
FSWeight       ------- -------  1.0000  0.8000  0.6400  0.5120  0.4096  0.3277  0.2621
TotalUsage      100.00 ------- 57513.2 86554.8 78945.2 82403.6 89055.1 99194.6 86908.1

USER
-------------
u0794535*         2.05    5.00+     7.37     1.21 -------     2.09     1.15 ------- -------
u0678934*         2.01    5.00+     0.14     1.99     2.08     5.54     1.12     0.75     3.37u0050476         23.04    5.00+
26.11   14.82   30.75   24.86   18.87   19.96   30.83u0637796*         0.02    5.00+ -------     0.08     0.02     0.00 -------
------- -------
```

Troubleshooting Example #1

- Why isn’t the job 110263 running? Answer:
  - First the job’s current priority score places it near the end of the queue, regardless of when the job was submitted.
  - The priority information reveals that the job is not gaining priority from FairShare. That suggests that the user has achieved or exceeded their FairShare target.
  - We also note that the job does not have a reservation due to the number of jobs ahead of it in the queue. That means the job waits for a reservation or for a backfill opportunity.
  - Conclusion: Nothing seems wrong or out of place with the behavior of the batch system. This is simply a case where the job has to wait for it’s turn to run. This is a tough sell at times to users though... ;)

Troubleshooting Example #2

User reports their job did not run to completion as it should have. The user reports that the system canceled or killed their job.

- First lets examine the jobs life cycle with ‘tracejob’.

```
Job: 202030.emrm.ipoib.privatearch.arches

07/26/2014 20:50:15  S    enqueuing into ember, state 1 hop 1
07/26/2014 20:50:15  A    queue=ember
07/26/2014 20:50:19  S    Job Run at request of root@emrm.ipoib.privatearch.arches
07/26/2014 20:50:20  A    user=u0656316 group=thomson account=owner-guest jobname=md_p_00_144100215 queue=ember
ctime=1406429415 qtime=1406429415 etime=1406429415 start=1406429420 … Resource_List.neednodes=1:ppn=12
Resource_List.nodect=1 Resource_List.nodes=1:ppn=12 Resource_List.walltime=05:00:00
07/26/2014 23:45:33  S    Job deleted at request of root@emrm.ipoib.privatearch.arches
07/26/2014 23:45:33  S    Job sent signal SIGTERM on delete
07/26/2014 23:45:33  A    requestor=root@emrm.ipoib.privatearch.arches
07/26/2014 23:49:01  S    Job sent signal SIGKILL on delete
07/26/2014 23:49:06  S    Exit_status=271 resources_used.cput=02:46:06 resources_used.mem=257728kb
resources_used.vmem=3995064kb resources_used.walltime=02:58:47
07/26/2014 23:49:06  S    on_job_exit valid pjob: 202030.emrm.ipoib.privatearch.arches (substate=50)
07/26/2014 23:49:06  A    user=u0656316 group=thomson account=owner-guest jobname=md_p_00_144100215 queue=ember
ctime=1406429415 qtime=1406429415 etime=1406429415 start=1406429420 ... Resource_List.neednodes=1:ppn=12
Resource_List.nodect=1 Resource_List.nodes=1:ppn=12 Resource_List.walltime=05:00:00 session=20791 end=1406440146
Exit_status=271 resources_used.cput=02:46:06 resources_used.mem=257728kb resources_used.vmem=3995064kb
resources_used.walltime=02:58:47
07/26/2014 23:49:45  S    dequeuing from ember, state COMPLETE
```

Troubleshooting Example #2

- Now lets examine the job events recorded in the Moab ‘events’ log files. We can ‘grep $JOBID $MOABHOMEDIR/stats/events.$DATE’.

```
20:50:20 1406429418:355712 job      202030      JOBSTART      0  12 u0656316   thomson   18000      Idle       ember 1406429415
1406429418 1406429418 1406429418      -      - >=    0M >=      0M          - 1406429415  12   12 -:oguest RESTARTABLE owner-
guest             -    -  0    0.00      ALL     1      0M      0M      0M        0 2140000000
em036.ipoib,em036.ipoib,em036.ipoib,em036.ipoib,em036.ipoib,em036.ipoib,em036.ipoib,em036.ipoib,em036.ipoib,em036.ipoib,em03
6.ipoib,em036.ipoib ember - - - - - 0.00 - - - 0,NAccessPolicy=SINGLEJOB - - - 3
23:45:33 1406439932:355782 job      202030      JOBCANCEL      0  12 u0656316   thomson   18000   Running       ember 1406429415
1406429418 1406429418 1406439932      -      - >=    0M >=      0M          - 1406429415  12   12 -:oguest RESTARTABLE owner-
guest             -    -  0 19062.31     ember     1      0M      0M      0M        0 2140000000
em036.ipoib,em036.ipoib,em036.ipoib,em036.ipoib,em036.ipoib,em036.ipoib,em036.ipoib,em036.ipoib,em036.ipoib,em036.ipoib,em03
6.ipoib,em036.ipoib ember - - - -
\START0\20\20\20\20\20\20\20\20\20\2000:00:01\201:00:00:00\20\20\20\20\20\20N/A\20\20\20\200\20\20\201\20job\20cancelled\20-
\20job\20preempted\20(202048)\0a 0.00 - - - 0,SID=20791,NAccessPolicy=SINGLEJOB - - - 3 MSG='job preempted (202048)'
```

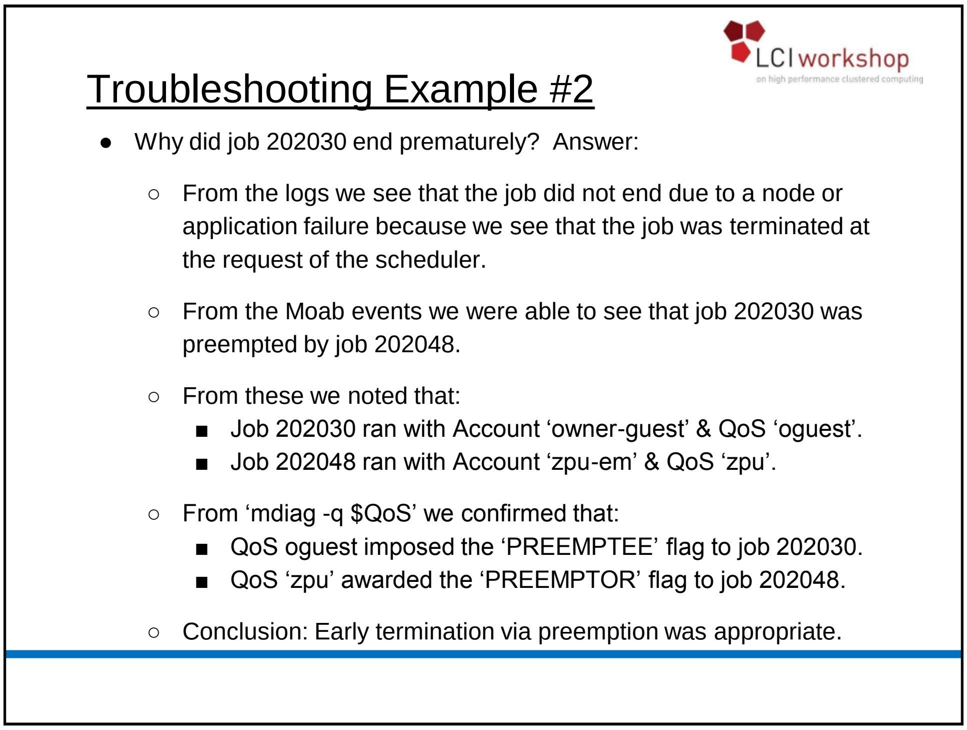
- Lets check what job 202048 looks like too.

```
23:49:13 1406440152:355785 job      202048      JOBSTART      0  48 u0439327      zpu    5400      Idle       ember 1406439932
1406440152 1406440152 1406440152      -      - >=    0M >=      0M          - 1406439932  48   12 -:zpu
ADVRES,RESTARTABLE,HASPREEMPTED   zpu-em            -    -  0    0.00     ember     1      0M      0M      0M        0 2140000000
em033.ipoib,em033.ipoib,em033.ipoib,em033.ipoib, … ,em036.ipoib,em036.ipoib,em036.ipoib,em036.ipoib ember - Zpu_4 - -
\START0\20\20\20\20\20\20\20\20\20-
00:00:01\20\20\2023:59:58\20\20\20\20\20\20N/A\20\20\20\200\20\2077\20cannot\20start\20job\20on\20reserved\20resources\20-
\20\0a 0.00 - - - 0,NAccessPolicy=SINGLEJOB - - - 220
```

Troubleshooting Example #2

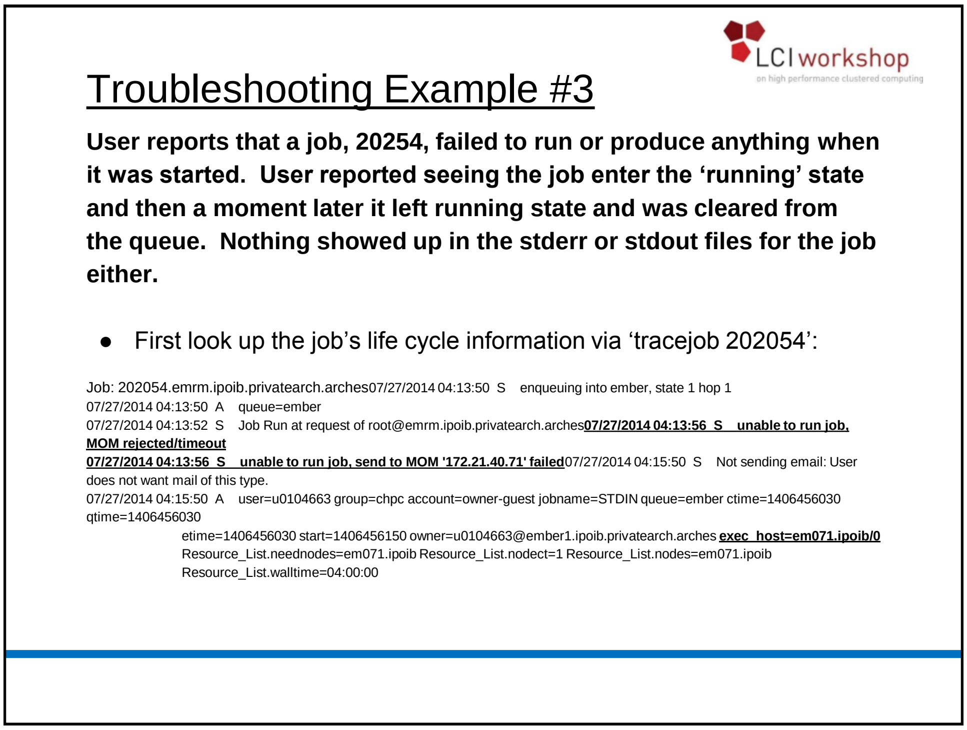
- Lets verify that job 202048 preempting job 202030 is appropriate. Lets look at the FLAGS associated with the QoS both job 202030 and 202048 ran under by running ‘mdiag -q oguest’ & ‘mdiag -q zpu’:

```
‘oguest’: QOS Status
System QOS Settings:  QList: --- (Def: DEFAULT)  Flags: -

Name              * Priority QTWeight QTTarget XFWeight XFTarget     QFlags     JobFlags Limitsoguest
1        0        0        0     0.00   PREEMPTEE            - -
  Accounts: owner-guest^


‘zpu’: QOS Status
System QOS Settings:  QList: --- (Def: DEFAULT)  Flags: -

Name              * Priority QTWeight QTTarget XFWeight XFTarget     QFlags     JobFlags Limitszpu
500000       0        0        0     0.00 PREEMPTOR,USERESERVED:Zpu_4       ADVRES  OMAXJWC=1209600
  NOTE:  qos zpu is tied to rsv 'Zpu_4'
  Accounts: zpu-em^
```



Troubleshooting Example #2

- Why did job 202030 end prematurely? Answer:
  - From the logs we see that the job did not end due to a node or application failure because we see that the job was terminated at the request of the scheduler.
  - From the Moab events we were able to see that job 202030 was preempted by job 202048.
  - From these we noted that:
    - Job 202030 ran with Account ‘owner-guest’ & QoS ‘oguest’.
    - Job 202048 ran with Account ‘zpu-em’ & QoS ‘zpu’.
  - From ‘mdiag -q $QoS’ we confirmed that:
    - QoS oguest imposed the ‘PREEMPTEE’ flag to job 202030.
    - QoS ‘zpu’ awarded the ‘PREEMPTOR’ flag to job 202048.
  - Conclusion: Early termination via preemption was appropriate.



Troubleshooting Example #3

User reports that a job, 20254, failed to run or produce anything when it was started. User reported seeing the job enter the ‘running’ state and then a moment later it left running state and was cleared from the queue. Nothing showed up in the stderr or stdout files for the job either.

- First look up the job’s life cycle information via ‘tracejob 202054’:

```
Job: 202054.emrm.ipoib.privatearch.arches07/27/2014 04:13:50  S    enqueuing into ember, state 1 hop 1
07/27/2014 04:13:50  A    queue=ember
07/27/2014 04:13:52  S    Job Run at request of root@emrm.ipoib.privatearch.arches07/27/2014 04:13:56  S    unable to run job,
MOM rejected/timeout
07/27/2014 04:13:56  S    unable to run job, send to MOM '172.21.40.71' failed07/27/2014 04:15:50  S    Not sending email: User
does not want mail of this type.
07/27/2014 04:15:50  A    user=u0104663 group=chpc account=owner-guest jobname=STDIN queue=ember ctime=1406456030
qtime=1406456030
                    etime=1406456030 start=1406456150 owner=u0104663@ember1.ipoib.privatearch.arches exec_host=em071.ipoib/0
                    Resource_List.neednodes=em071.ipoib Resource_List.nodect=1 Resource_List.nodes=em071.ipoib
                    Resource_List.walltime=04:00:00
```

Troubleshooting Example #3

- Now examine the pbs_mom logs on node ‘em071.ipoib’ to see if we can determine why job ‘202054’ failed to run.

```
07/27/2014 04:13:53;0001;   pbs_mom.3778;Job;job_nodes;job: 202054.emrm.ipoib.privatearch.arches numnodes=1
numvnod=107/27/2014 04:13:56;0008;   pbs_mom.3778;Job;check_pwd;no password entry for user u0104663
07/27/2014 04:13:56;0001;   pbs_mom.3778;Svr;pbs_mom;LOG_ERROR::start_exec, bad credentials: job id
202054.emrm.ipoib.privatearch.arches
07/27/2014 04:13:56;0001;   pbs_mom.3778;Job;start_exec;bad credentials: job id 202054.emrm.ipoib.privatearch.arches
07/27/2014 04:13:56;0001;   pbs_mom.3778;Job;exec_bail;bailing on job 202054.emrm.ipoib.privatearch.arches code -1
07/27/2014 04:13:56;0008;   pbs_mom.3778;Req;send_sisters;sending ABORT to sisters for job
202054.emrm.ipoib.privatearch.arches07/27/2014 04:13:56;0080;   pbs_mom.3778;Svr;scan_for_exiting;searching for exiting jobs
07/27/2014 04:13:56;0008;   pbs_mom.3778;Job;kill_job;scan_for_exiting: sending signal 9, "KILL" to job
202054.emrm.ipoib.privatearch.arches, reason: local task termination detected
07/27/2014 04:13:56;0080;   pbs_mom.3778;Svr;preobit_preparation;top
07/27/2014 04:13:56;0080;   pbs_mom.3778;Job;202054.emrm.ipoib.privatearch.arches;epilog subtask created with pid 9247 -
substate set to JOB_SUBSTATE_OBIT - registered post_epilogue
07/27/2014 04:13:56;0080;   pbs_mom.3778;Req;post_epilogue;preparing obit message for job 202054.emrm.ipoib.privatearch.arches
07/27/2014 04:13:56;0080;   pbs_mom.3778;Job;202054.emrm.ipoib.privatearch.arches;obit sent to server
07/27/2014 04:13:56;0001;   pbs_mom.3778;Svr;pbs_mom;LOG_ALERT::obit_reply, server rejected job obit - unexpected job state
07/27/2014 04:13:56;0001;   pbs_mom.3778;Job;202054.emrm.ipoib.privatearch.arches;server rejected job obit - unexpected job state
07/27/2014 04:13:56;0080;   pbs_mom.3778;Job;mom_deljob;deleting job 202054.emrm.ipoib.privatearch.arches in state OBIT
07/27/2014 04:13:56;0080;   pbs_mom.3778;Job;202054.emrm.ipoib.privatearch.arches;removed job script
```

Troubleshooting Example #3

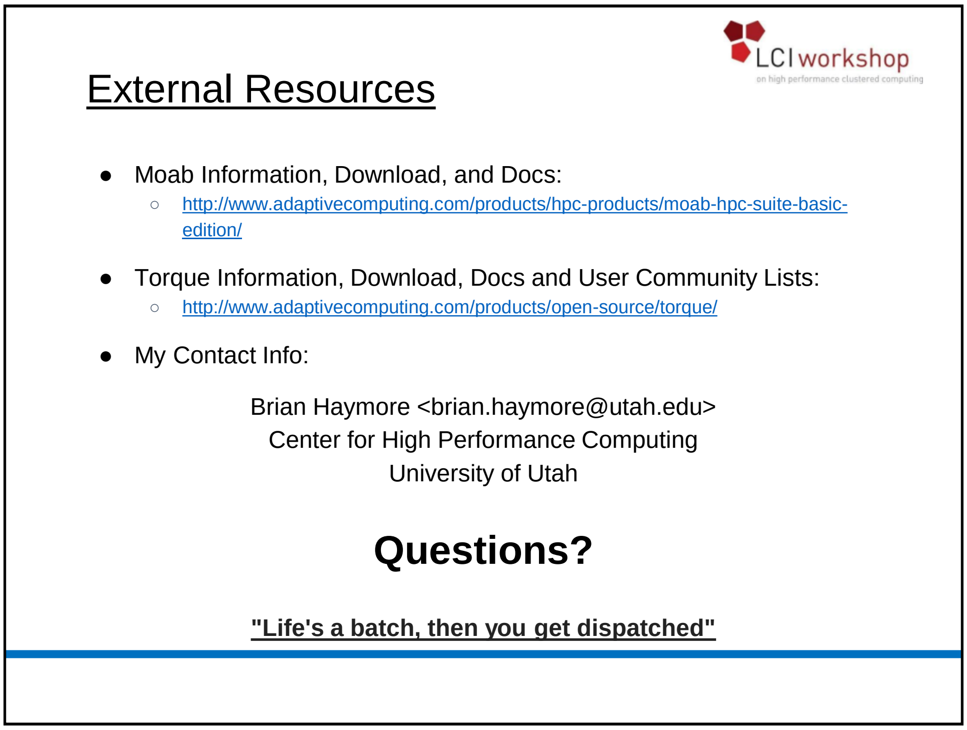
- Now that we see that node ‘em071.ipoib’ is having issues accessing user account information we need to take action to prevent that node from affecting other jobs/users until we correct the situation.
  - Execute ‘pbsnodes -o em071.ipoib -N “offlined due to issue with account lookup -bdh 7/27/14”’ to mark the node offline preventing other jobs from being impacted by the state of this node and also set a note to detail why we have offlined it.
  - ‘pbsnodes -ln’ displays all nodes that are down and or offline along with any ‘note’ associated with the nodes.
  - Once we have corrected the issue with node ‘em071.ipoib’ and we are ready to return it to service we can execute ‘pbsnodes -r em071.ipoib -N “”’. This will instruct torque to clear the offline state on the node and renegotiate a connection with the pbs_mom on the node before achieving online status.

Troubleshooting Example #3

- Why did job 202054 fail to run or produce any output?
  - 'tracejob' showed us that the job was rejected by the execution host, em071.ipoib, assigned to job 202054.
  - From the pbs_mom logs on host em071.ipoib we were able to determine that pbs_mom rejected the job because it was not able to look up the users credentials.
  - We then offlined the affected execution host to prevent it from impacting others jobs.
  - Once the issue on the execution host was corrected we can then clear the offline state and allow the node to return to service.
  - Conclusion: the failure to run was not due to anything the user did wrong, but that the execution host was in a bad state of health.

Troubleshooting Example #3

- Acknowledgement: The issue I used here in this example is one that we should not really need to worry about in practice. Sites should implement 'node health checks' in some form to address simple health state issues that can occur.
  - Torque and Moab both have mechanisms to support 'node health checks' in addition to the 'prologue' and 'epilogue' facilities.
  - Warewulf NHC is good place to start and is covered in the 'Node Health Check" presentation by Michael Jennings of LBNL.
    - http://warewulf.lbl.gov/trac/wiki/Node%20Health%20Check



External Resources

- Moab Information, Download, and Docs:
  - http://www.adaptivecomputing.com/products/hpc-products/moab-hpc-suite-basic-edition/
- Torque Information, Download, Docs and User Community Lists:
  - http://www.adaptivecomputing.com/products/open-source/torque/
- My Contact Info:

Brian Haymore <brian.haymore@utah.edu>
Center for High Performance Computing
University of Utah

Questions?

"Life's a batch, then you get dispatched"