Batch Scripts for RA & Mio

Timothy H. Kaiser, Ph.D.
tkaiser@mines.edu

Jobs are Run via a Batch System

Ra and Mio are shared resources

- Purpose:
  - Give fair access to all users
  - Have control over where jobs are run
  - Set limits on times and processor counts
  - Allow exclusive access to nodes while you own them

Batch Systems

- Create a script that describes:
  - What you want to do
  - What resources (# nodes) you want
  - How long you want the resources
- Submit you script to the queuing system
  - Wait for some time for your job to start
    - Resources
    - Time required
  - Your job runs and you get your output

Typical Batch System

- You might hear of Torque, Moab, or PBS. For our purposes they are the same
- Marketing pitch from the web page:

> **Our Workload Manager** is a policy-based job scheduler and event engine that enables utility-based computing for clusters.
>
> Moab Workload Manager combines intelligent scheduling of resources with advanced reservations to process jobs on the right resources at the right time. It also provides flexible policy and event engines that process workloads faster and in line with set business requirements and priorities.

RA - Mio Differences

- msub is only on RA (historical reasons)
  - On Mio the command is qsub but we recommend that you alias msub=qsub
  - qsub is on RA but you should not normally use it
- On RA your scripts must contain an account

A Simple Script for a MPI job

```
!/bin/bash
#PBS -l nodes=1:ppn=8
#PBS -l walltime=02:00:00
#PBS -N testIO
#PBS -A account
#PBS -o stdout
#PBS -e stderr
#PBS -V
#PBS -m abe
#PBS -M tkaiser@mines.edu
#-----------------------
cd $PBS_O_WORKDIR
mpiexec -n 8 c_ex00
```

Scripts contain normal shell commands and comments designated with a # that are interpreted by PBS

On RA you must specify an account

Running and monitoring jobs

- Use msub (qsub) to submit the job
- qstat to see its status (Q,R,C)
- qdel to kill the job if needed

```
[tkaiser@ra ~/mydir]$ msub mpmd

8452
[tkaiser@ra ~/mydir]$ qstat 8452
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
8452.ra                   testIO           tkaiser                0 R SHORT
[tkaiser@ra ~/mydir]$
[tkaiser@ra ~/mydir]$
[tkaiser@ra ~/mydir]$ qstat -u tkaiser
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
8452.ra                   testIO           tkaiser                0 R SHORT
[tkaiser@ra ~/mydir]$
[tkaiser@ra ~/mydir]$
[tkaiser@ra ~/mydir]$ qdel 8452
```

A Simple Script for a MPI job

Command	Comment
`#!/bin/bash`	We will run this job using the "Bash" shell
`#PBS -l nodes=1:ppn=8`	We want 1 node with 8 processors for a total of 8 processors
`#PBS -l walltime=02:00:00`	We will run for up to 2 hours
`#PBS -N testIO`	The name of our job is testIO
`#PBS -o stdout`	Standard output from our program will go to a file stdout
`#PBS -e stderr`	Error output from our program will go to a file stderr
`#PBS -A account`	An account is required on RA - from your PI
`#PBS -V`	Very important! Exports all environment variables from the submitting shell into the batch shell.
`#PBS -m abe`	Send mail on abort, begin, end use n to suppress
`#PBS -M tkaiser@mines.edu`	Where to send email to (cell phone?)
`#----------------------`	Not important. Just a separator line.
`cd $PBS_O_WORKDIR`	Very important! Go to directory $PBS_O_WORKDIR which is the directory which is where our script resides
`mpiexec -n 8 c_ex00`	Run the MPI program c_ex00 on 8 computing cores.

Another example

```
#!/bin/tcsh
#PBS -l nodes=2:ppn=8
#PBS -l walltime=00:20:00
#PBS -N testIO
#PBS -o out.$PBS_JOBID
#PBS -e err.$PBS_JOBID
#PBS -A support
#PBS -m n
#PBS -M tkaiser@mines.edu
#PBS -V
cat $PBS_NODEFILE > fulllist
sort -u $PBS_NODEFILE > shortlist
cd $PBS_O_WORKDIR
mpiexec -np 16 $PBS_O_WORKDIR/c_ex01
```

```
[tkaiser@ra ~/mpmd]$ cat fulllist
compute-9-9.local
compute-9-9.local
compute-9-9.local
compute-9-9.local
compute-9-9.local
compute-9-9.local
compute-9-9.local
compute-9-9.local
compute-9-8.local
compute-9-8.local
compute-9-8.local
compute-9-8.local
compute-9-8.local
compute-9-8.local
compute-9-8.local
compute-9-8.local
compute-9-8.local
[tkaiser@ra ~/mpmd]$ cat shortlist
compute-9-8.local
compute-9-9.local
[tkaiser@ra ~/mpmd]$ ls err* out*
err.8453.ra.mines.edu  out.8453.ra.mines.edu
```

A Useful Utility

```
[ra]$ nsort --help

    A utility for returning a uniquely sorted
    list of nodes for a parallel application.

TYPICAL USAGE WITHIN A PBS SCRIPT
     nsort $PBS_NODEFILE > shortlist


AUTHOR
    Timothy H. Kaiser
    tkaiser@mines.edu
    Colorado School of Mines
    July 18, 2008


LOCATION
    /lustre/home/apps/utility/nsort
```

Replaces using the Unix sort utility

Using MVAPICH2

Issues

- Likes full paths to applications
- Need to manually start an MPI daemon
- Need to shut it down

Your script becomes

```
#!/bin/tcsh
#PBS -l nodes=2:ppn=8
#PBS -l walltime=00:20:00
#PBS -N testIO
#PBS -o out.$PBS_JOBID
#PBS -e err.$PBS_JOBID
#PBS -m n
#PBS -M tkaiser@mines.edu
#PBS -V

cd $PBS_O_WORKDIR

nsort $PBS_NODEFILE > shortlist
mpdboot --totalnum=2 --file=shortlist

mpiexec -np 16 $PBS_O_WORKDIR/c_ex01

mpdallexit
```

These numbers should match

mpdboot starts MPI
mpdallexit stops it

From the Documentation

```
sort $PBS_NODEFILE | uniq -c | awk '{printf "%s:%s\n",$1,$2}' > mpd.nodes
mpdboot -f mpd.hosts -n [NUM NODES REQUESTED]
...
mpdallexit
```

More Script notes

- #PBS -l
  - Used to specify resource requests
  - Can have multiple per "-l" lines per script
- #PBS -l nodes=2:ppn=8:fat
  - Ra has 16 Gbyte and 32 Gbyte nodes
  - Specify the 32 Gbyte nodes
- #PBS -l naccesspolicy=singlejob
  - Grant exclusive access to a node
  - Not normally needed

PBS Variables

Variable	Meaning
PBS_JOBID	unique PBS job ID
PBS_O_WORKDIR	jobs submission directory
PBS_NNODES	number of nodes requested
PBS_NODEFILE	file with list of allocated nodes
PBS_O_HOME	home dir of submitting user
PBS_JOBNAME	user specified job name
PBS_QUEUE	job queue
PBS_MOMPORT	active port for mom daemon
PBS_O_HOST	host of currently running job
PBS_O_LANG	language variable for job
PBS_O_LOGNAME	name of submitting user
PBS_O_PATH	path to executables used in job script
PBS_O_SHELL	script shell
PBS_JOBCOOKIE	job cookie
PBS_TASKNUM	number of tasks requested (see pbsdsh)
PBS_NODENUM	node offset number (see pbsdsh)

Where's my Output?

- Standard Error and Standard Out don't show up until after your job is completed
- Output is temporarily stored in /tmp on your nodes
  - You can ssh to your nodes and see your output
  - Looking into changing this behavior, or at least make it optional
- You can pipe output to a file:
  - mpiexec myjob > myjob.$PBS_JOBID

Useful Torque Commands

Command	Description
canceljob	**cancel job**
checkjob	provide detailed status report for specified job
mdiag	provide diagnostic reports for resources, workload, and scheduling
mjobctl	control and modify job
mrsvctl	create, control and modify reservations
mshow	displays various diagnostic messages about the system and job queues
msub	**submit a job (Don't use qsub)**
releasehold	release job defers and holds
releaseres	release reservations
sethold	set job holds
showq	**show queued jobs**
showres	show existing reservations
showstart	show estimates of when job can/will start
showstate	show current state of resources

http://www.clusterresources.com/products/mwm/docs/a.gcommandoverview.shtml

Other Commands

canceljob	cancel job
changeparam	change in memory parameter settings
checkjob	provide detailed status report for specified job
checknode	provide detailed status report for specified node
diagnose	provide diagnostic report for various aspects of resources, workload, and scheduling
mcredctl	controls various aspects about the credential objects within Moab
mdiag	provide diagnostic reports for resources, workload, and scheduling
mjobctl	control and modify job
mnodectl	control and modify nodes
mrmctl	query and control resource managers
mrsvctl	create, control and modify reservations
mschedctl	modify scheduler state and behavior
mshow	displays various diagnostic messages about the system and job queues
mshow -a	query and show available system resources
msub	scheduler job submission
releasehold	release job defers and holds
releaseres	release reservations
resetstats	reset scheduler statistics
runjob	force a job to run immediately
sethold	set job holds
setqos	modify job QOS settings
setres	set an admin/user reservation
setspri	adjust job/system priority of job
showbf	show current resource availability
showconfig	show current scheduler configuration
showq	show queued jobs
showres	show existing reservations
showstart	show estimates of when job can/will start
showstate	show current state of resources
showstats	show usage statistics
showstats -f	show various tables of scheduling/system performance

Useful Torque commands

Command	Description
tracejob	trace job actions and states recorded in TORQUE logs
pbsnodes	view/modify batch status of compute nodes
qalter	modify queued batch jobs
qdel	delete/cancel batch jobs
qhold	hold batch jobs
qrls	release batch job holds
qrun	start a batch job
qsig	send a signal to a batch job
qstat	view queues and jobs
pbsdsh	launch tasks within a parallel job
qsub	submit a job (Don't use)

http://www.clusterresources.com/torquedocs21/a.acommands.shtml

Torque commands

momctl	manage/diagnose MOM (node execution) daemon
pbsdsh	launch tasks within a parallel job
pbsnodes	view/modify batch status of compute nodes
qalter	modify queued batch jobs
qchkpt	checkpoint batch jobs
qdel	delete/cancel batch jobs
qhold	hold batch jobs
qmgr	manage policies and other batch configuration
qrerun	rerun a batch job
qrls	release batch job holds
qrun	start a batch job
qsig	send a signal to a batch job
qstat	view queues and jobs
qsub	submit jobs (Don't use except for interactive runs)
qterm	shutdown pbs server daemon
tracejob	trace job actions and states recorded in TORQUE logs

Topics to be Covered

- Running less than N tasks per node
  - OpenMP
  - Large memory per task
- Multi Program Multi Data
- Bag of task using PBS
- Lots of tricks
- Staging data

Running Interactive

Interactive Runs

- It is possible to grab a collection of nodes and run parallel interactively
- Nodes are yours until you quit
- Might be useful during development
- Running a number of small parallel jobs
- Required (almost) for running parallel debuggers

Multistep Process

- Ask for the nodes using qsub
  - `qsub -q INTERACTIVE -A support -I -V -l nodes=1`
- You are automatically Logged on to the nodes you are given
- Ignore: cp: cannot stat `/opt/torque/mom_priv/jobs/73156.ra.local.SC': No such file or directory
- Go to your working directory
- Run your job(s)
- Logout to release your nodes

Don’t abuse it, lest you be shot.

Example

```
[tkaiser@ra ~/guide]$qsub -q INTERACTIVE -I -V -l nodes=1 -l naccesspolicy=singlejob
qsub: waiting for job 1280.ra.mines.edu to start
qsub: job 1280.ra.mines.edu ready

[tkaiser@compute-9-8 ~]$cd guide
[tkaiser@compute-9-8 ~/guide]$mpirun -n 4 c_ex00
Hello from 0 of 4 on compute-9-8.local
Hello from 1 of 4 on compute-9-8.local
Hello from 2 of 4 on compute-9-8.local
Hello from 3 of 4 on compute-9-8.local
[tkaiser@compute-9-8 ~/guide]$
[tkaiser@compute-9-8 ~/guide]$mpirun -n 16 c_ex00
Hello from 1 of 16 on compute-9-8.local
Hello from 0 of 16 on compute-9-8.local
Hello from 3 of 16 on compute-9-8.local
Hello from 4 of 16 on compute-9-8.local
...
...
Hello from 15 of 16 on compute-9-8.local
Hello from 2 of 16 on compute-9-8.local
Hello from 6 of 16 on compute-9-8.local
[tkaiser@compute-9-8 ~/guide]$
[tkaiser@compute-9-8 ~/guide]$exit
logout

qsub: job 1280.ra.mines.edu completed
[tkaiser@ra ~/guide]$
```

Ensures exclusive access

Challenge

- Take the example that we ran yesterday
  - Have process 0 sent an integer, 12345, to process 1
  - Both processes print out their ID and the integer
  - All other processes just print ID
- Compile using mpicc or mpif90

Challenge

- Modify your run script
  - Put output into a “stamped” file
  - Send you email when it is done
- Run on 8 and 16 cores
- Run interactively on 2, 4, and 7 cores