An Introduction to the Portable Batch System (PBS)

Michael L. Seltzer
(with a huge thank you to Rita Singh)
CMU Robust Speech Recognition Group
January 24, 2002
Why do we need a new queue?

• As machines get increasingly faster and cheaper, our current queue machines are becoming increasingly outdated and ready to become doorstops, compost, and fodder for the mechanical engineering catapult class projects (did anyone see that?)

• We can replace those machines with newer ones, but we are still paying a lot of money in LSF licenses.

• In the spirit of open source, we have decided to try move to a free queue system. The money we are now using on licenses can be used to buy more machines.

• The old LSF queue is still up and running with 15 DEC Alphas and will continue to do so until the new queue is stable.
Our New Queue Machines

- Over the last 6 months, we have purchased 19 new Linux machines for batch processing.
  - 17 Dual Processor P3 1 GHz, 1 GB RAM
  - 1 Dual Processor P3 1 GHz, 4GB RAM
  - 1 Single Processor P4 1.7 GHz, 1GB RAM
- This queue currently has
  - 37.7 GHz of processing power
  - >1000 GB of disk space
The Cast of Characters
(*.speech.cs.cmu.edu)

mickey minnie dumbo goofy fred wilma
kermit piggy fozzie gonzo bunsen beaker
bigbird ernie bert utonium bubbler blossom buttercup
Who’s Who?

- **minnie**
  - PBS Server
  - Dual P3 1GHz
  - 1GB RAM

- **mickey**
  - Single P4 1.7GHz
  - 1GB RAM

- **bigbird**
  - Dual P3 1GHz
  - 4GB RAM

---

Carnegie Mellon University

Robust Speech Group
Disk Space & Partitions *(more on next slide)*

<table>
<thead>
<tr>
<th>Hostname</th>
<th>Disk Size</th>
<th>Partitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>mickey</td>
<td>40 GB</td>
<td>1x4GB, 2x8GB, 1x13.5GB</td>
</tr>
<tr>
<td>minnie</td>
<td>40 GB</td>
<td>1x2GB, 1x3GB, 3x10GB</td>
</tr>
<tr>
<td>bigbird</td>
<td>36 GB</td>
<td>1x2GB, 1x4GB, 3x8GB</td>
</tr>
<tr>
<td>fred</td>
<td>40 GB</td>
<td>1x2GB, 1x3GB, 3x10GB</td>
</tr>
<tr>
<td>wilma</td>
<td>40 GB</td>
<td>1x2GB, 1x3GB, 3x10GB</td>
</tr>
<tr>
<td>bubbler</td>
<td>40 GB</td>
<td>1x40GB</td>
</tr>
<tr>
<td>blossom</td>
<td>40 GB</td>
<td>1x40GB</td>
</tr>
<tr>
<td>buttercup</td>
<td>40 GB</td>
<td>1x40GB</td>
</tr>
<tr>
<td>utonium</td>
<td>40 GB</td>
<td>1x40GB</td>
</tr>
</tbody>
</table>
## Disk Space & Partitions (2)

<table>
<thead>
<tr>
<th>Hostname</th>
<th>Disk Size</th>
<th>Partitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>bert</td>
<td>60 GB</td>
<td>2x4GB, 3x10GB, 1x14GB</td>
</tr>
<tr>
<td>ernie</td>
<td>60 GB</td>
<td>2x4GB, 3x10GB, 1x14GB</td>
</tr>
<tr>
<td>kermit</td>
<td>60 GB</td>
<td>2x4GB, 3x10GB, 1x14GB</td>
</tr>
<tr>
<td>piggy</td>
<td>60 GB</td>
<td>2x4GB, 3x10GB, 1x14GB</td>
</tr>
<tr>
<td>gonzo</td>
<td>60 GB</td>
<td>2x4GB, 3x10GB, 1x14GB</td>
</tr>
<tr>
<td>dumbo</td>
<td>100 GB</td>
<td>1x10GB, 4x20GB</td>
</tr>
<tr>
<td>goofy</td>
<td>100 GB</td>
<td>1x10GB, 4x20GB</td>
</tr>
<tr>
<td>fozzie</td>
<td>100 GB</td>
<td>1x10GB, 4x20GB</td>
</tr>
<tr>
<td>bunsen</td>
<td>100 GB</td>
<td>9x10GB</td>
</tr>
<tr>
<td>beaker</td>
<td>100 GB</td>
<td>9x10GB</td>
</tr>
</tbody>
</table>
What is the Portable Batch System (PBS)?

- PBS is a mechanism for submitting batch job requests on or across multiple machines.
- It provides:
  - Scheduling of job requests among available queues and machines on a given system according to available system resources and requirements
  - Job submission on one system with routing to another system for execution
  - Job and queue monitoring
Getting set up to use PBS

- What you need:
  - User accounts on all queue machines
  - Disk space on a queue machine - all data, scripts that your job requires has to be local to one of the queue machines
  - /usr/local/PBS/bin added to your path list (in .cshrc)
  - /usr/local/PBS/man added to your manpath list
    - Alternatively, %> man –M /usr/local/PBS/man <command>
Submitting a job: **qsub**

- qsub – to submit a job to the queue

- Basic format:
  
  ```
  qsub -switch <arg> -switch <arg> .... -switch <arg> /path/script
  ```

- Specify complete path to script, not relative path.

- Unlike LSF, script cannot have any arguments. (more on this later)

- Writes jobid to stdout when job is submitted
## Submitting a job: `qsub <switches> <args>`

<table>
<thead>
<tr>
<th><strong>sw</strong></th>
<th><strong>description</strong></th>
<th><strong>arguments/examples</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-e</code></td>
<td>file to write standard error</td>
<td><code>/net/fred/usr1/mseltzer/expt1/test.err</code></td>
</tr>
<tr>
<td><code>-o</code></td>
<td>file to write standard output</td>
<td><code>/net/fred/usr1/mseltzer/expt1/test.out</code></td>
</tr>
</tbody>
</table>
| `-j`   | join stderr and stdout | `oe` – join stdout to stderr  
          `eo` – join stderr to stdout |
| `-N`   | job name  
         15 chars max, no white space,  
         first char has to be alphabetic | `myjob.1` |
| `-r`   | rerun a job if it fails | `y|n` |
| `-m`   | mail options | `a` – mail sent when job is aborted (default)  
          `b` – mail sent when job is begun  
          `e` – mail sent when job terminates |
| `-q`   | Specify which queue | default – the only one we have (default) |
Specifying resources: `qsub -l`

- The `–l` switch allows you to specify resources such as cpu time, actual running time, memory, or machines for your job.
- `qsub –l name1=value1,name2=value2`
- `qsub –l name1=value1 –l name2=value1`
- Examples:
  - `qsub –l mem=512mb myjob.sh`
  - `qsub –l walltime=1:10:00 myjob.sh`
  - `qsub –l cput=10000 myjob.sh`
  - `qsub –l nodes=mickey myjob.sh`

- See `pbs_resources_linux` man page for more resources
Specifying dependencies: `qsub -W`

- The –W switch allows you to specify other attributes for your job. For our purposes, this is most useful for establishing dependencies among jobs
- `qstat –W depend=type:value`
- Example:
  - Start after job 001 and 002 have ended w/ or w/o errors
    `qsub –W depend=afterany:001:002 myjob.sh`
  - Start after job 001 and 002 have ended without any errors
    `qsub –W depend=afterok:001:002 myjob.sh`

- See `qsub` man page for additional dependencies
An example: qsub

#!/bin/csh

set mydir = /net/wilma/usr1/mseltzer/pbs_test

set jobid1 = `qsub -N job.1 -l mem=256mb -e $mydir/job.1.err \\
               -o $mydir/job.1.out -r y $mydir/test_qsub_depend.1.csh`
echo $jobid1

set jobid2 = `qsub -N job.2 -l mem=256mb -e $mydir/job.2.err \\
               -o $mydir/job.2.out -r y $mydir/test_qsub_depend.2.csh`
echo $jobid2

set jobid3 = `qsub -N job.3 -e $mydir/job.3.err -o \\
               $mydir/job.3.out -r y -W depend=afterok:${jobid1}:${jobid2} \\
               $mydir/test_qsub_depend.3.csh`
echo $jobid3
Checking the status of your jobs: qstat

- Once you launch jobs, you can check on their status using `qstat`
- A job can be in one of several possible states:

<table>
<thead>
<tr>
<th>sw</th>
<th>Description/argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>-a</td>
<td>alternate format (a little more information)</td>
</tr>
<tr>
<td>-u</td>
<td>username</td>
</tr>
<tr>
<td>-n</td>
<td>Nodes allocated to job are shown</td>
</tr>
<tr>
<td>-f</td>
<td>Full listing (lots of information)</td>
</tr>
<tr>
<td>-Q</td>
<td>Information about the queue itself</td>
</tr>
</tbody>
</table>

| E   | Job is exiting after having run             |
| H   | Job is held                                 |
| Q   | job is queued, eligible to run              |
| R   | job is running                              |
An example: `qstat`

```
mseltzer@goofy.speech.cs.cmu.edu: /tmp

wilma:pbs_test> qsub_depend_test.csh
3421.minnie.speech.cs.cmu.edu
3422.minnie.speech.cs.cmu.edu
3423.minnie.speech.cs.cmu.edu
wilma:pbs_test> qstat -n -a

minnie.speech.cs.cmu.edu:

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Username</th>
<th>Queue</th>
<th>Jobname</th>
<th>SessID</th>
<th>NDS</th>
<th>TSK</th>
<th>Req'd Memory</th>
<th>Req'd Time</th>
<th>S</th>
<th>Elap Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>3421.minnie.spe mseltzer default</td>
<td>job.1</td>
<td>11063</td>
<td>--</td>
<td>--</td>
<td>256mb</td>
<td>--</td>
<td>-- R</td>
<td>--</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3422.minnie.spe mseltzer default</td>
<td>job.2</td>
<td>11070</td>
<td>--</td>
<td>--</td>
<td>256mb</td>
<td>--</td>
<td>-- R</td>
<td>--</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3423.minnie.spe mseltzer default</td>
<td>job.3</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>-- H</td>
<td>--</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

wilma:pbs_test> qstat -Q default

<table>
<thead>
<tr>
<th>Queue</th>
<th>Max</th>
<th>Tot</th>
<th>Ena</th>
<th>Str</th>
<th>Que</th>
<th>Run</th>
<th>Hld</th>
<th>Wat</th>
<th>Trn</th>
<th>Ext</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>0</td>
<td>3</td>
<td>yes</td>
<td>yes</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Execution</td>
</tr>
</tbody>
</table>

wilma:pbs_test> qstat

Job id | Name | User | Time Use | S | Queue |
------|------|------|----------|---|-------|
3423.minnie | job.3 | mseltzer | 0 R | default |

wilma:pbs_test> qstat
```
Deleting a job: `qdel`

- Jobs can be deleted from the queue using `qdel <jobid>`
If you are used to LSF…

- Things to worry about:
  - Make sure all your paths are complete and absolute. No relative or partial paths in your scripts.
  - The job script submitted to the batch server cannot have any arguments. However, that script can call another script which has arguments.
  - Job names are limited to 15 characters. The scheduler will not accept your job if the name is too long.

- A handy tool – bsub_pbs
  - /afs/cs/usr/mseltzer/bin/bsub_pbs
  - A handy Perl script (by way of Bhiksha) which will take an LSF style “bsub” command string, convert it to qsub and launch it. Not perfect, but very useful for using the PBS system quickly if you are used to LSF.
Some Final Things to Consider

- If you are going to use the queue, take disk space on only one machine.

- Do not launch extremely long jobs on the queue, unless they can be broken down into smaller jobs. Long jobs that only run on a single machine should not be run on the queue.

- Machines may move into and out of the queue as need/use dictates.
Wrap-up

- PBS provides a free replacement to LSF for batch processing. The queue is currently up and running, with 3 machines (goofy, blossom, bubbler) not on the queue. They will be back in the queue shortly (hopefully!).

- The queue software can accept up to 1024 processors. Imagine paying for that many LSF licenses. (Maybe all that money we will save can pay for a sys admin!)

- Currently, Rita and I have queue manager privileges, but we are definitely willing to train others!

- 10 more P4 machines (names TBD!) will be joining the queue soon.
More information

- Websites with information about PBS
  - http://www.openpbs.org
One last ditch effort to make a talk about batch servers funny…

"Now, Beakie, we'll just flip this switch and 60,000 refreshing volts of electricity will surge through your body. Ready?"
Thanks, you’ve been a great crowd...