Using an Array Job to run a set of experiments

A motivating example

Open a terminal on Mercer and follow the example described here

In `/share/apps/examples/arrayjob/` we have a simple wordcount program (from [http://www.yasyf.com/coding/simple-python-word-frequency-count/](http://www.yasyf.com/coding/simple-python-word-frequency-count/)) and some classic texts from Project Gutenberg:
The program counts the frequency of each word in a text file. The `myjob.q` script runs `wordcount.py` against a single sample text.

For a small set of jobs such as this, you could simply edit `myjob.q` 5 times, and submit each individually. This becomes impractical when, say, scanning a whole canon of texts. But with only minor modifications to the script we can submit the whole set simultaneously:

```
$ cat myjob.q
#!/bin/bash
#PBS -l walltime=5:00
#PBS -l nodes=1:ppn=1
#PBS -l mem=1gb
TUTDIR=${SCRATCH}/tut3/
rsync -av /share/apps/examples/arrayjob/ $TUTDIR/
RUNDIR=${TUTDIR}/run-1
mkdir -p $RUNDIR
cd $RUNDIR
python $TUTDIR/wordcount.py
$TUTDIR/sample-1.txt > wordcounts.out.1
```

```
$ cat myarrayjob.q
#!/bin/bash
#PBS -l walltime=5:00
#PBS -l nodes=1:ppn=1
#PBS -l mem=1gb
#PBS -t 1-5
TUTDIR=${SCRATCH}/tut3/
rsync -av /share/apps/examples/arrayjob/ $TUTDIR/
RUNDIR=${TUTDIR}/run-$PBS_ARRAYID
mkdir -p $RUNDIR
cd $RUNDIR
python $TUTDIR/wordcount.py
$TUTDIR/sample-$PBS_ARRAYID.txt > wordcounts.out.$PBS_ARRAYID
```

Important things to notice:

- The `arrayjob` script describes resources needed for a **single task** in the array (same as the non-arrayjob script)
- Torque sets the environment variable `$PBS_ARRAYID` to the index within the array of the current task. You can use this in your script to uniquely identify each task
  - Most programming languages provide a `getenv()` function to get the value of an environment variable - this is one way to use the PBS_ARRAYID to control tasks
- We run each task in its own directory. This is very important since tasks will run simultaneously, and you do not want them to overwrite or interfere with each other
- The `qsub` option "-t" sets the job up as an array job, in this case with 5 tasks: 1, 2, 3, 4, 5
Exercise
Copy myarrayjob.q to your home directory and submit it.

Now use qstat -u $USER to see your jobs in the queue. What do you notice about the array job?

You should notice:

- The job ID has [ ] in it
- The job status (column "S") is "Q" - even after some of the jobs start running. The status is the status of the whole array, not just an individual job, so it will only be "R" when all of the tasks are running - and will become "C" once all of the tasks are completed.

```bash
$ qsub myarrayjob.q
3217255[
$ qstat -u $USER
soho.es.its.nyu.edu:

<table>
<thead>
<tr>
<th>Req'd</th>
<th>Req'd</th>
<th>Elap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job ID</td>
<td>Username</td>
<td>Queue</td>
</tr>
<tr>
<td>SessID</td>
<td>NDS</td>
<td>TSK</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>3217255[</td>
<td>ab123</td>
<td>s48</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1024m</td>
</tr>
</tbody>
</table>

To see the status of individual tasks, again use "-t".

```bash
$ qstat -tu $USER
soho.es.its.nyu.edu:

<table>
<thead>
<tr>
<th>Req'd</th>
<th>Req'd</th>
<th>Elap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job ID</td>
<td>Username</td>
<td>Queue</td>
</tr>
<tr>
<td>SessID</td>
<td>NDS</td>
<td>TSK</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>3217255[1]</td>
<td>ab123</td>
<td>s48</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1024m</td>
</tr>
<tr>
<td>3217255[2]</td>
<td>ab123</td>
<td>s48</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1024m</td>
</tr>
<tr>
<td>3217255[3]</td>
<td>ab123</td>
<td>s48</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1024m</td>
</tr>
<tr>
<td>3217255[4]</td>
<td>ab123</td>
<td>s48</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1024m</td>
</tr>
<tr>
<td>3217255[5]</td>
<td>ab123</td>
<td>s48</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1024m</td>
</tr>
</tbody>
</table>
```

The wordcount example runs very quickly. To make it last long enough to see individual tasks in a "Running" state, you can add "sleep 60" to the end of the job script

Quirks and Gotchas

- The [] is part of the job ID - if you omit it, the job ID will not be recognised:
There is a bug in `qstat`, so if you try to see the array by job ID, `qstat` will print every job in the system:

```
$ qstat 3217255
qstat: Unknown Job Id Error 3217255.soho.es.its.nyu.edu
```

To work around this, use the `-t` option for `qstat`:

```
$ qstat -t 3217255[]
```

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Name</th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>3217255[1]</td>
<td>myarrayjob.q-1</td>
<td>ab123</td>
</tr>
<tr>
<td>00:00:00 C s48</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3217255[2]</td>
<td>myarrayjob.q-2</td>
<td>ab123</td>
</tr>
<tr>
<td>00:00:00 C s48</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3217255[3]</td>
<td>myarrayjob.q-3</td>
<td>ab123</td>
</tr>
<tr>
<td>00:00:00 C s48</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3217255[4]</td>
<td>myarrayjob.q-4</td>
<td>ab123</td>
</tr>
<tr>
<td>00:00:00 C s48</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3217255[5]</td>
<td>myarrayjob.q-5</td>
<td>ab123</td>
</tr>
<tr>
<td>00:00:00 C s48</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

... or pipe `qstat` through `grep`:

```
$ qstat | grep 3217255
3217255[]             myarrayjob.q     ab123
0 C s48
```

**Arrayjob job IDs have [ and ] in them**
These characters are interpreted by bash as pathname expansion (eg "ls [ab]" will show files starting with a or b).
If commands using arrayjob IDs are behaving strangely, try escaping the [ ], eg

```
$ qstat -t 3217255\\[
```

Next: