PBS `qsub`

When you are new to PBS, the place to start is the `qsub` command, which **submits** jobs to your HPC systems. The only jobs that the `qsub` accepts are scripts, so you'll need to package your tasks appropriately. Here is a simple example script (`myjob.pbs`):

```
#!/bin/bash

#PBS -N demo // job name
#PBS -o demo.txt // output file
#PBS -e demo.txt // error output file
#PBS -q workq // queue name
#PBS -l mem=100mb // requested memory size

mpiexec -machinefile /etc/myhosts -np 4 /home/user/area
```

The first line specified the shell to use in interpreting the script, while the next few lines starting with `#PBS` are directives that are passed to PBS. The first names the job, the next two specify where output and error output go, the next to last identifies the queue that is used, and the last lists a resource that will be needed, in this case `100 MB` of memory. The blank line signals the end of PBS directives. Lines that follow the blank line indicate the actual job (details on the `mpi` commands are discussed later). Once you have created the batch script for your job, the `qsub` command is used to submit the job:

```
qsub job.pbs
```

Some of the more commonly used qsub options are:

- `-q queue`: Select the queue to run the job in. The queues you can use are listed by running qstat.
- `-l walltime=??:??:??`: The wall clock time limit for the job. Time is expressed in seconds as an integer, or in the form:

`[[hours:]minutes:]seconds[.milliseconds]`

- `-l vmem=???MB`: The total (virtual) memory limit for the job - can be specified with units of “MB” or “GB” but only integer values can be given. There is a small default value. Your job will only run if there is sufficient free memory so making a sensible memory request will allow your jobs to run sooner. A little trial and error may be required to find how much memory your jobs are using.
- `-l ncpus=?`: The number of cpus required for the job to run on.
    - `-l ncpus=N`: If the number of cpus requested, N, is small the job will run within a single shared memory node. If the number of cpus specified is greater, the job will be distributed over multiple nodes.
    - `-l ncpus=N:M` This form requests a total of N cpus with (a multiple of) M cpus per node.

Look at the qsub and pbs_resources man page for complete details of all options. Note that `-l` options maybe combined as a comma separated list with no spaces, e.g., `-l vmem=512mb,walltime=10:00:00`.

However, the newer PBS (**PBS Pro**), comes with the concept of resource chunking using a `select` parameter, let’s see another job submission script example:

```
#!/bin/bash
#PBS -P <project code>
#PBS -q workq
#PBS -l select=2:ncpus=16:mpiprocs=16
#PBS -l place=scatter:excl
#PBS -l walltime=<hh:mm:ss>
#PBS -o <output-file>
#PBS -e <error-file>

module load intel
module load mpi/intel.4.2.0
mpirun <exec> <inputfiles>
```

Where the line `-l select=2:ncpus=16:mpiprocs=16` is the number of processors required for the **MPI** job. select specifies the number of nodes (or chunks of resource) required; `ncpus` indicates the number of CPUs per chunk required; and `mpiprocs` represents the number of MPI processes to run out of the CPUs selected (normally `ncpus` would equal `mpiprocs`).

selected (normally `ncpus` would equal `mpiprocs`).

As this is not the most intuitive command, the following table is provided as guidance as to how this command works:

select	ncpus	mpiprocs	description
`2`	`16`	`16`	`32` Processor job, using `2` nodes and `16` processors per node
`4`	`8`	`8`	`32` Processor job, using `4` nodes and `8` processors per node
`16`	`1`	`1`	`16` Processor job, using `16` nodes running `1` `mpi` process per processor and utilising `1` processor per node
`8`	`16`	`16`	`128` Processor job, using `8` nodes and `16` processors per node (each running an `mpi` process)

The flag `ompthreads` can be used to specify the number of **OpenMP** threads used in the job script. For example: `#PBS -l select=2:ncpus=16:mpiprocs=8:ompthreads=2` This would automatically set the environment variable `OMP_NUM_THREADS=2` and it can easily be seen when using OpenMP with MPI that ompthreads multiplied by mpiprocs should equal ncpus (if you want all MPI tasks to each have the same number of threads).

If your job exceeds this time the scheduler will terminate the job. It is recommended to find a usual runtime for the job and add some more (say 20%) to it. For example, if a job took approximately 10 hours, the walltime limit could be set to 12 hours, e.g. `"-l walltime=12:00:00"`. By setting the walltime the scheduler can perform job scheduling more efficiently and also reduces occasions where errors can leave the job stalled but still taking up resource for the default much longer walltime limit.

'qsub' man page (PBS Pro)