HPC PBSPro script files

Important: Most software will only consume 1 CPU core - e.g., requesting 8 CPU cores for a PAUP job blocks other people using the unused 7 CPU cores. Example 1 below would likely be the most users should be basing their job scripts from. If in doubt, contact HPRC staff.

For more information about PBSPro please click to see guide. For a brief description of PBS directives provided in examples below, see the "**Brief Explanation of PBS directive used in examples above**" section immediately following the final example PBS script.

HPC staff should be able to assist researchers needing help with PBS scripts.

Singularity Example

The following PBS script requests 1 CPU core, 2GB of memory, and 24 hours of walltime

```
#!/bin/bash
#PBS -j oe
#PBS -m ae
#PBS -N JobName1
#PBS -M FIRSTNAME.LASTNAME@jcu.edu.au
#PBS -l walltime=24:00:00
#PBS -l select=1:ncpus=1:mem=2gb

cd $PBS_O_WORKDIR
shopt -s expand_aliases
source /etc/profile.d/modules.sh
echo "Job identifier is $PBS_JOBID"
echo "Working directory is $PBS_O_WORKDIR"

module load R/4.1.2
R ...    # Replace ... with your arguments & options.
```

Module Examples
Example 1:

The following PBS script requests 1 CPU core, 2GB of memory, and 24 hours of walltime for the running of "paup -n input.nex".

```
#!/bin/bash
#PBS -j oe
#PBS -m ae
#PBS -N JobName1
#PBS -M FIRSTNAME.LASTNAME@jcu.edu.au
#PBS -l walltime=24:00:00
#PBS -l select=1:ncpus=1:mem=2gb

cd $PBS_O_WORKDIR
shopt -s expand_aliases
source /etc/profile.d/modules.sh
echo "Job identifier is $PBS_JOBID"
echo "Working directory is $PBS_O_WORKDIR"

module load paup
paup -n input.nex
```

If the file containing the above content has a name of `JobName1.pbs`, you simply execute `qsub JobName1.pbs` to place it into the queueing system.

Example 3:

The following PBS script requests 20 CPU cores, 60GB of memory, and 10 days of walltime for running of an MPI job.

```
#!/bin/bash
#PBS -j oe
#PBS -m ae
#PBS -N JobName3
#PBS -M FIRSTNAME.LASTNAME@my.jcu.edu.au
#PBS -l walltime=240:00:00
#PBS -l select=1:ncpus=20:mem=60gb

cd $PBS_O_WORKDIR
shopt -s expand_aliases
source /etc/profile.d/modules.sh
echo "Job identifier is $PBS_JOBID"
echo "Working directory is $PBS_O_WORKDIR"

module load migrate
module load mpi/openmpi
mpirun -np 20 -machinefile $PBS_NODEFILE migrate-n-mpi
...
```

If the file containing the above content has a name of `JobName3.pbs`, you simply execute `qsub JobName3.pbs` to place it into the queueing system.

Example 2:

The following PBS script requests 8 CPU cores, 32GB of memory, and 3 hours of walltime for running of 8 MATLAB jobs in parallel.

```
#!/bin/bash
#PBS -j oe
#PBS -m ae
#PBS -N JobName2
#PBS -M FIRSTNAME.LASTNAME@my.jcu.edu.au
#PBS -l walltime=3:00:00
#PBS -l select=1:ncpus=8:mem=32gb

cd $PBS_O_WORKDIR
shopt -s expand_aliases
source /etc/profile.d/modules.sh
echo "Job identifier is $PBS_JOBID"
echo "Working directory is $PBS_O_WORKDIR"

module load matlab
matlab -r myjob1 &
matlab -r myjob2 &
matlab -r myjob3 &
matlab -r myjob4 &
matlab -r myjob5 &
matlab -r myjob6 &
matlab -r myjob7 &
matlab -r myjob8 &
wait    # Wait for background jobs to finish.
```

If the file containing the above content has a name of `JobName2.pbs`, you simply execute `qsub JobName2.pbs` to place it into the queueing system.

Example 4:

The following PBS script request uses job arrays. If you aren't proficient with bash scripting, using job arrays could be painful. The example below has each sub-job requesting 1 CPU core, 1 GB of memory, and 80 minutes of walltime.

```
#!/bin/bash
#PBS -j oe
#PBS -m ae
#PBS -N ArrayJob
#PBS -M FIRSTNAME.LASTNAME@jcu.edu.au
#PBS -l walltime=1:20:00
#PBS -l select=1:ncpus=1:mem=1gb

cd $PBS_O_WORKDIR
shopt -s expand_aliases
source /etc/profile.d/modules.sh

module load matlab
matlab -r myjob$PBS_ARRAYID
```

If the file containing the above content has a name of `ArrayJob.pbs` and you will be running 32 sub-jobs, you simply use `qsub -t 1-32 ArrayJob.pbs` to place it into the queueing system.

Note: I haven't done extensive testing of job arrays.

Example 5:

The following script is a rework of Example 2 to use the `/fast/tmp` filesystem for a hyperthetical workflow that is I/O intensive. This example assumes 1 output file per job.

Usage of /fast/tmp

Please make sure you first **create** an place all files in a folder that matches your jc number eg: jcXXXXXXXX

```
#!/bin/bash
#PBS -j oe
#PBS -m ae
#PBS -N JobName2
#PBS -M FIRSTNAME.LASTNAME@my.jcu.edu.au
#PBS -l walltime=3:00:00
#PBS -l select=1:ncpus=8:mem=32gb

cd $PBS_O_WORKDIR
shopt -s expand_aliases
source /etc/profile.d/modules.sh
echo "Job identifier is $PBS_JOBID"
echo "Working directory is $PBS_O_WORKDIR"

mkdir -p /fast/tmp/jc012345/myjobs
cp -a myjob1.m myjob2.m myjob3.m myjob4.m myjob5.m myjob6.m myjob7.m myjob8.m /fast/tmp/jc012345/myjobs/
pushd /fast/tmp/jc012345/myjobs

module load matlab
matlab -r myjob1 &
matlab -r myjob2 &
matlab -r myjob3 &
matlab -r myjob4 &
matlab -r myjob5 &
matlab -r myjob6 &
matlab -r myjob7 &
matlab -r myjob8 &
wait    # Wait for background jobs to finish.

cp -a out1.mat out2.mat out3.mat out4.mat out5.mat out6.mat out7.mat out8.mat $PBS_O_WORKDIR/
popd
rm -rf /fast/tmp/jc012345/myjobs
```

Consider the possibility that you may be running more than one workflow at any given time. Using subdirectories is a good way of segregating workflows (at a storage layer).

Brief Explanation of PBS directive used in examples above

Directive	Description of impact
`#PBS -j oe`	Merge STDOUT & STDERR streams into a single file
`#PBS -m ae`	Send an Email upon job abort/exit.
`#PBS -N ...`	Assign a meaningful name to the job (replace `...` with 1 "word" - e.g., test_job).
`#PBS -M ...`	Email address that PBSPro will use to provide job information (if desired)
`#PBS -l walltime=HH:MM:SS`	Amount of clock time that your job is likely to required.
`#PBS -l select=1:ncpus=X:mem=Ygb`	Request 1 chunk of "X CPU cores" & "Y GB of RAM". The "select=1:" is not really required as it is the default. Due to JCU cluster size, requests for more than 1 chunk should/will be rejected.

Brief Details on some extra PBS/Torque directives

Directive(s)	Description of purpose
`#PBS -d <PATH_TO_DIRECTORY>`	Sets the working directory for you job to <PATH>.
`#PBS -o <OUTPUT_FILE_PATH>`	Explicit specification of file that will hold the standard output stream from you job.
`#PBS -V`	Export environment variables to the batch job

For full details on directives that can be used, use "`man qsub`" on a HPC login node or look at online documentation for Torque.

PBS/Torque Variables

The following variables can be useful within your PBS job script. Some are present in the examples above.

Variable	Description
`PBS_JOBNAME`	Job name specified by the user
`PBS_O_WORKDIR`	Working directory from which the job was submitted
`PBS_O_HOME`	Home directory of user submitting the job
`PBS_O_LOGNAME`	Name of user submitting the job
`PBS_O_SHELL`	Script shell
`PBS_O_JOBID`	Unique PBS job id
`PBS_O_HOST`	Host on which job script is running
`PBS_QUEUE`	Name of the job queue
`PBS_NODEFILE`	File containing line delimited list on nodes allocated to the job (may be required for MPI jobs).
`PBS_O_PATH`	Path variable used to locate executables within the job script