The University of Queensland

Research Computing Centre

Batch Computing using PBSPro

Document Status

This update: 20200716 by David Green

PBSPro is the batch system used on Awoonga, FlashLite and Tinaroo.

The PBSPro Batch System

In late 2017, all three clusters (Awoonga, FlashLite and Tinaroo) changed their batch system from Torque to PBSPro.

This was necessary to obtain better performance of the batch system under extreme loads and also to provide a number of diagnostic tools to help us, to help users.

A "wrapper script" has been deployed so that, in most cases, existing Torque job submissions will be understood and translated into PBSPro systax as the job is submitted.
You should work to migrate your job scripts to PBSPro syntax, and can use the `qstat -f JOB_NUMBER` command to learn about the PBSPro syntax corresponding to your submitted torque syntax.

Batch Computing in General

Using batch jobs, managed the batch system, is the preferred way for HPC users to fairly utilise HPC resources.
Because it it shared facility, you are expected to request resources and wait to be allocated those resources.
When the system is very busy, you may need to wait longer.

PBS Pro provides 2 categories of jobs (Interactive and Regular) that are described elsewhere in this document.

In addition, on UQ HPC, the Nimrod workload system is available to assist you with managing ensembles of many jobs. Nimrod operates in two modes on the HPC, embedded (where the Nimrod subsystem is deployed as part of a regular batch job) and Portal which provides more convenient web access.

The following diagram is an overview of the modes for batch jobs.

```
.
└── Batch_Jobs
    ├── Interactive
    │   └── Use_qsub_-I_Command
    │       ├── 0.Wait_for_Job_to_Start
    │       ├── 1.Load_Software_Module
    │       ├── 2.Start_Using_Software
    │       └── 3.Exit_Job_Using_exit_Command
    ├── Nimrod
    │   ├── Embedded
    │   │   ├── 0.Write_nsh_Job_Script
    │   │   ├── 1.Submit_Job_Script_Using_qsub_Command
    │   │   ├── 2.Nimrod_Manages_Workload
    │   │   └── 3.Wait_for_Job_to_Finish
    │   └── Portal
    │       ├── 0.Access_Nimrod_Portal
    │       ├── 1.Create_Nimrod_Experiment
    │       ├── 2.Start_Experiment_in_Portal
    │       ├── 3.Monitor_Progress_in_Portal
    │       └── 4.Wait_for_Experiment_to_Finish
    └── Regular
        ├── 0.Write_PBS_Job_Script
        ├── 1.Submit_Job_Script_Using_qsub_Command
        ├── 2.Script_does_it_all_unattended
        └── 3.Wait_for_Job_to_Finish
```

Getting Started with the Batch System

Detailed information about PBSPro V 18.2.3 is available in the User Guide
The three clusters all use the same batch system although they have individual batch servers on each cluster.

PBSPro Commands

The manual pages on the system also have information about the various PBS related "q" commands:
qalter, qdel, qhold, qmove, qmsg, qrerun, qrls, qselect, qsig, qstat

Here is a summary of the commands that are most relevant to a HPC user.
All these commands have a man page and you should familiarize yourself with the syntax for using them.

Command	Purpose	Example
qalter	To alter the resource request of a queued job. (see Notes below)	`qalter -l walltime=02:00:00 jobID`
qdel	To delete a queued or running job immediately.	`qdel jobID`
qhold	Places a hold on a job so that it remains in the queue but unable to start. User can hold their jobs.	`qhold jobID`
qmgr	To read (and for admins modify) batch system configuration.	`qmgr -c "list queue Short"`
qmove	To move a queued job to a different queue. Should not need to be used.	`qmove queueName jobID@`
qmsg	To send a text message to a running job that gets written into the O or E file.	`qmsg -E "Are we there yet?" jobID`
qorder	To swap the order of two queued job.	`qorder jobID1 jobID2`
qrerun	To immediately kill and requeue a job.	`qrerun jobID`
qrls	To release a hold previously placed on a job.	`qrls -h u jobID`
qselect	To generate a list of job identifiers based on criteria.	`qstat -a `qselect s R -q Multiple``
qsig	To send a unix system signal to your job. Your job will need to be able to respond to that.	`qsig -s 10 jobID`
qstat	To generate an output of the status of queues or jobs.	`qstat -aw*`
qsub	To submit a job.	`qsub jobscript.pbs`

Command Notes:

- **qalter**
  - A job that is still queued may be altered by the owner.
  - The non-walltime resource settings are fixed once job starts running.
  - A regular user cannot increase walltime once a job has started.
  - HPC admin team can increase the walltime of a running job, however we are unable to do so immediately. Submit a support request at least 48 hours before the job is due to be killed.
- **qrerun**
  - You may wish that your job never restarts automatically when there is a problem, or a qrerun issued (i.e. it should not be rerunnable).
    To ensure this is the case you must mark it as with `#PBS -r n` in your job script, otherwise your job may restart if interrupted.
- **qrls**
  - A regular user can only release a hold that is of the user type (u).
    There are also operator (o) and system (s) holds that may be applied by the admins.
    To see the holds on your job/jobs use the following command pipeline as an example

```
qstat -f | grep -e ^Job -e Hold_Types
```

Additionally there are commands that are only available to the system administrators.

Command	Purpose
qterm	Systems Admin Only- Terminate the batch server.
qdisable/qenable	Systems Admin Only- Controlling queues.
qstart/qstop	Systems Admin Only- Controlling queues.
qrun	Systems Admin Only - To force a job to start.

Wrapped PBSPro Commands

Some of the PBSpro commands are provided with a "wrapper script".
A wrapper script affords additional logic and formatting which enhances the functionality of the "bare" PBSPro commands. For most users, the wrapper script is encountered earlier in the $PATH than the "bare" version, so the wrapper is what gets run as the command.

Currently, two PBS commands have wrappers

- qstat
  - The command **`/usr/local/bin/qstat`** will produce a comprehensive list of your jobs across all 3 clusters.
  - If you wish to only see the output of the standard PBS Pro qstat command, you should use the "bare" command `/opt/pbs/bin/qstat`
- qsub
  - The wrapper does some syntax checking and fixes any Windows end-of-line character issues

If the wrapped **`/usr/local/bin/qstat`** command does not respond, then there could a problem with accessing the PBS Pro service on one of the other HPC clusters.
If this is the case then you should use the "bare" **`/opt/pbs/bin/qstat`** command to obtain status information for the local cluster you are running the command upon.

Queues, Limits, Nodes

Queues

The default queue (workq) is a routing queue and the workq will route jobs to the appropriate execution queue based on the resource requirements.

Most users will never need to specify the queue for their job, or if they do then use `#PBS -q workq`.

The exception would be if you have been told by RCC Support to use a specific queue as part of Enhanced Access or other some special arrangement that has been made.

The job execution queues that are available on each cluster vary depending on the emphasis of the HPC cluster.

You can find out the names, the basic properties, limits and activity levels on each batch queue by using the command **`qstat -q`**

```
[root@awonmgr2 ~]# /opt/pbs/bin/qstat -q

server: awonmgr2

Queue            Memory CPU Time Walltime Node   Run   Que   Lm  State
---------------- ------ -------- -------- ---- ----- ----- ----  -----
CVL                12gb     --    168:00:0    1     0     0   --   E R
Short             250gb     --    24:00:00    1   154    70   --   E R
Interact          250gb     --    48:00:00    1     3     1   --   E R
Long              250gb     --    336:00:0    1    34     7   --   E R
Single            250gb     --    168:00:0    1   172   128   --   E R
GPU                48gb     --    48:00:00    1     0     0   --   E S
DeadEnd             --      --       --     --      0     0   --   E S
Special            60gb     --    48:00:00    1     0     0   --   E R
workq               --      --       --     --      0     0   --   E R
                                               ----- -----
                                                 363   206
```

Note that queued job array elements do not show up in the Q column count.

You can find out more detail about the level of activity on each batch queue by using the command **`qstat -Q`**

```
[root@awonmgr2 ~]# /opt/pbs/bin/qstat -Q
Queue              Max   Tot Ena Str   Que   Run   Hld   Wat   Trn   Ext Type
---------------- ----- ----- --- --- ----- ----- ----- ----- ----- ----- ----
CVL                  0     0 yes yes     0     0     0     0     0     0 Exec
Short                0 26540 yes yes    69   152     1     0     0     0 Exec
Interact             0     4 yes yes     1     3     0     0     0     0 Exec
Long                 0    43 yes yes     5    34     2     0     0     0 Exec
Single               0   375 yes yes   127   172     2     0     0     0 Exec
GPU                  0     0 yes  no     0     0     0     0     0     0 Exec
DeadEnd              0     0 yes  no     0     0     0     0     0     0 Exec
Special              0     0 yes yes     0     0     0     0     0     0 Exec
workq                0     0 yes yes     0     0     0     0     0     0 Rout
```

Note that due to a software bug, the total number of jobs (Tot) listed in qstat -Q output is usually bogus.
The number of queued and running jobs is fairly accurate, although job array elements are not counted separately in those statistics.

A job ends up in the DeadEnd queue when it cannot find a home in one of the other valid queues.
When this happens it usually means that your resource requirements are inappropriate and cannot be match by any queue.
Check the queue limits again and adjust your job's resource request.

Queue and Site Limits

Apart from the limits detailed in the `qstat -q` output, there are other limits in place.
These can be examined using the `qmgr` command.

Command	**Purpose**
`/opt/pbs/bin/qmgr -c "print queue Short"`	Details of a specific queue.
`/opt/pbs/bin/qmgr -c "print node aw009g"`	Details of a specific node.
`/opt/pbs/bin/qmgr -c "print server"`	All details of the server configuration (queues).

See **`man /opt/pbs/share/man/man8/qmgr.8B`** for further details.

The output of the "print server" option includes the site (or complex) wide settings such as

```
set server max_run_res.ncpus = [u:PBS_GENERIC=128]
set server max_run_res.cpu_hrs = [u:PBS_GENERIC=26600]
set server max_array_size = 10000
```

These settings may change to match demand so you should query the current settings if it is important to you.

Nodes Information

The `pbsnodes` command will provide you with information about the state and availability of nodes.

There are several options to the pbsnodes command:

- `pbsnodes -a` shows all nodes in full detail
- `pbsnodes -aS` shows all nodes as a table with one row per node
- `pbsnodes -l` option only shows offline nodes in a tabular format

See `man pbsnodes` for further options.

A node can be in one of the following states

- offline (unable to run any jobs)
- free (has some available capacity for running jobs - not necessarily that it is completely idle)
- job-busy (has no space left for running jobs due to committed CPUs or RAM)

When a node has a specific queue associated with it, it means it will only accept jobs from that queue.
Usually, you would need to eligible to use that queue and node.

Batch Jobs

Batch computing takes a small amount of additional organisation and testing, but the reward can be found in the volume of work that can be achieved with incremental human effort.

Job Submission Script Structure

Batch jobs in PBS Pro can be submitted completely from the command line, however, most people will use a job submission script.

A job submission script has a simple structure with three sections:

	Looks like	Description	Details
1	`#!/bin/bash`	(optional)	shell invocation line if you want to test the submission outside of the batch system
2	`#PBS ...`	the job setup/structure	multiple job specification lines
3	Linux commands/comments	the job payload	multiple lines of Unix commands to execute once the job commences

Note: Any #PBS directives are ignored if they occur after the first unix command in the script file.

Examples of job submission scripts appear elsewhere in this guide.

Transferring PBS Job Scripts from Windows Computers

Be careful if you are editing or saving your PBS script on a Windows computer and transferring it to HPC.

There can be control characters (^M) embedded in the file that are not visible but may cause the batch job to spontaneously fail when it starts to run.
You can avoid this by always transferring the text file using text transfer mode of WinSCP/FileZilla/CyberDuck.
If you want to be sure, you can also use the `dos2unix` command on the HPC to ensure the file is 100% Unix (and PBS) compatible.

GOOD NEWS:
The qsub wrapper script `/usr/local/bin/qsub` will now automatically perform a `dos2unix` on your PBS script before passing it to the batch system.

Batch Job Attributes and Options

The `#PBS` directives are used to specify a variety of attributes of the job.
Most important of these are the job resources which includes the number of "chunks".

One way to think about these chunks of resources is to imagine the job as running within a box (or a number of identical boxes).
The box (or boxes) would have "dimensions" of

- the number of CPU cores (`ncpus=`),
- the amount of RAM required (`mem=`),
- the amount of time the job is allowed to run for before it is forcibly stopped (walltime=).

Different jobs might use a combinations of these dimensions as their chunks. For example,

- A job might have a large number of CPUs and a short walltime- e.g. a single node shared memory (OpenMP) code.
- A job might run on a single CPU and have a long walltime and a large memory footprint- e.g. a genome assembly.
- A job might run on multiple whole nodes and have a moderate walltime- e.g. a CFD or computational chemistry simulation based on message passing (MPI) code.

The number of chunks is governed by the `select=` parameter.
A typical resource request (within a job script) might look like

```
#PBS -l select=1:ncpus=2:mem=4GB
#PBS -l walltime=08:00:00
```

Such a job would be requesting

- 1 chunk with "dimensions" of
- 2 CPUs
- 4 GB of RAM
- 8 hours of walltime

The `select=` parameter is a multiplier for the other non-walltime resources.
That is, `#PBS -l select=5:ncpus=2:mem=4GB` is requesting a total of 10 CPUs and 20GB of RAM

Jobs that are meant to run across multiple cpu cores, or across multiple physical nodes, will have additional parameters to specify (that is, there are additional dimensions to their resource "box(es)" referred to earlier). These parameters will include

- the number of parallel shared memory threads to use (`ompthreads=`)
- the number of message passing threads to run per node (`mpiprocs=`)
  It is perfectly reasonable, in some circumstances, for a job to request a smaller number of mpiprocs or ompthreads than the number of ncpus for that job.

Job submissions may have a number of other parameters to specify

- the job name,
- dependencies on other jobs,
- output options
- notification options, amongst others.

Job options can be set in a script file (using the #PBS) directives or on the command line as options.
Job options provided at the qsub command line will override the corresponding job options in a job script.

Option	Purpose	Default Behavior
`-A`	accounting string for usage reporting	if possible it will work out the value from your group memberships
`-l`	specify the resource requirements (cpu, mem, walltime etc.)	some reasonable defaults are set but a poorly specified job may be rejected
`-N`	job name upto 15 characters	will be the name of the submission script file
`-S`	the Unix shell variant to run the job with	same as your login shell- for most people this is /bin/bash
`-e`	the path (full or relative) for the job's stderr	a file called jobname.*e*JOBID in $PBS_O_WORKDIR
`-o`	the path (full or relative) for the job's stdout	a file called jobname.*o*JOBID in $PBS_O_WORKDIR
`-h`	place a hold on the job as it is submitted so it cannot start until released	
`-J`	specify a job array	you need to provide the range values for the array
`-v`	environment variables and shell functions to be exported to the job	
`-I`	interactive job submission - starts the job and takes you with it !	This option only makes sense at the command line.

The following email notification related options are currently not fully supported on Awoonga/FlashLite/Tinaroo HPCs.
You are unable to email to addresses outside of the HPC.

Option	Purpose	Default Behavior
`-m`	select when to be emailed ... i.e. on **a**(bort) **b**(egin) and/or **e**(nd)	This option will only be valid for email addresses that are internal to the HPC.

-M	email address email notifications	This currently has no effect for email addresses that are external to the HPC.

See **qsub manual page** or the PBS Pro User Guide for further details.

How long should the walltime be for my job?

This question is often asked.
The answer is nearly always, measure it, or grossly overestimate it!

We recommend that you do so with the following method

- over-estimate the required walltime (eg. 2 week is **`-l walltime=336:00:00`** is the maximum for most queues on the UQ HPC)
- let a first job(s) finish, use job holds to prevent other jobs from starting
- inspect that job's "e" file or use **`qstat -xf JOBID`** to assess the walltime used (not the cputime)
- submit all similar future jobs using that runtime plus 10% to allow for delays due to storage or network.

You will need to repeat this experimentation if you are using different software or markedly different inputs because the run time may vary considerably.

Note:
If you grossly overestimate your walltimes, your jobs may wait a little longer to get started.
For walltimes greater than 24 hours and upto 168 hours there will be very little difference in the wait time.
It is not possible for a regular user to `qalter` an increase to the walltime of their job once it starts running.
If you underestimate the amount of walltime, your jobs will be terminated once the requested walltime is exceeded.
Please do not expect that RCC staff will always be able to respond to a request to have the walltime of a job extended.
Avoid that situation by following the advice provided above.

A Basic Batch Job

The following job will be called SimpleDemo, will wait on hold until I qrls the job number, then it will run and finish quickly

```
uqdgree5@awoonga1:~/Tests/PBSPro> cat simpledemo.pbs
#!/bin/bash
#
#PBS -A UQ-RCC
#
#PBS -l select=1:ncpus=1:mem=4GB
#PBS -l walltime=01:00:00
#PBS -N SimpleDemo
#PBS -h
#
echo $HOSTNAME
```

```
uqdgree5@awoonga1:~/Tests/PBSPro> qsub simpledemo.pbs
40562.awongmgmr1

uqdgree5@awoonga1:~/Tests/PBSPro> qstat -aw1n

awongmgmr1:
                                                                                  Req'd  Req'd   Ela
Job ID                          Username        Queue           Jobname         SessID   NDS  TSK   Memory Time  S Tim
------------------------------ --------------- --------------- --------------- -------- ---- ----- ------ ----- - ---
40562.awongmgmr1                uqdgree5        Short           SimpleDemo           --     1     1    4gb 01:00 H  --
uqdgree5@awoonga1:~/Tests/PBSPro>
uqdgree5@awoonga1:~/Tests/PBSPro> qrls 40562.awongmgmr1
uqdgree5@awoonga1:~/Tests/PBSPro> qstat -aw1n
```

Each job will generate an "e" file and an "o" file.
The "o" file contains any output from the commands in the script.
The "e" files contains any error messages as well as a summary of the job's performance.

```
uqdgree5@awoonga1:~/Tests/PBSPro> ls -salrt SimpleDemo.*
0 -rw------- 1 uqdgree5 qris-uq  12 Feb 23 02:13 SimpleDemo.o40562
0 -rw------- 1 uqdgree5 qris-uq 688 Feb 23 02:13 SimpleDemo.e40562

uqdgree5@awoonga1:~/Tests/PBSPro> cat SimpleDemo.o40562
aw031.local

uqdgree5@awoonga1:~/Tests/PBSPro> cat SimpleDemo.e40562
########################## Execution Started ############################
JobId:40562.awongmgmr1
```

```
UserName:uqdgree5
GroupName:qris-uq
ExecutionHost:aw031
################################################################################
######################### Job Execution History ################################
JobId:40562.awongmgmr1
UserName:uqdgree5
GroupName:qris-uq
JobName:SimpleDemo
SessionId:123879
ResourcesRequested:mem=4gb,ncpus=1,place=free,walltime=01:00:00
ResourcesUsed:cpupercent=0,cput=00:00:00,mem=0kb,ncpus=1,vmem=0kb,walltime=00:00:00
QueueUsed:Short
AccountString:UQ-RCC
ExitStatus:0
################################################################################
uqdgree5@awoonga1:~/Tests/PBSPro>
```

Note:
There have been problems with the delivery of e and o files at the end of otherwise successful batch jobs because the directory path included spaces or punctuation characters.
It is safer to use _ in place of a space and to avoid punctuation characters in your directory names.

Your Account String

In order to track and report usage on HPC, every user has been allocated to one or more accounting groups.
We need to be able to account for all usage on the cluster to satisfy our stakeholders that their entitlements are being met, and to assist with planning the resource management into the future.

The accounting groups are unix system groups.
Every UQ user belongs to a group like UQ-FACULTY-SCHOOL which reflects your organisational unit.
Additionally, for project work, users may have been allocated one or more project account groups.

- Individuals may be working on multiple projects, or have allocations granted via a number of pathways.
- Groups of users may belong to the same project and need to expend their project allocation collectively.
- Those with specific allocations will need to ensure that they expend their allocations within the granting period.
- We may need to be able to easily and accurately know how much resource was needed to run your jobs and what remains of your allocation or partner share.

A valid account string must be used when submitting jobs.
The job submission mechanism checks your membership of the group you selected and will reject a job that does not use a valid account string.

You can find out what your group memberships (valid account string values) are by using the command `groups`

```
uqdgree5@awoonga1:~> groups
qris-uq support1 flashlite1 Q0030RW Q0224RW Q0286RW Q0030RO UQ-RCC sw-SLIM sw-ORCA
```

You can see that this user is a member of several groups:

- primary group (qris-uq)
- admin support groups (support1 and flashlite1),
- collection storage groups (Q0030RW Q0224RW etc.)
- their main organisational unit group (UQ-RCC)
- and a couple of software access control groups (sw-SLIM and sw-ORCA).
  To submit jobs, always use the account string (group name) that begins with UQ- unless you are submitting project based workloads.

The first group named by the groups command is usually your primary group (unless you actively changed group using the newgrp command). The simplest way to check is to login afresh and issue the groups command.
Not all groups you belong to will necessarily be valid account strings. For example, the group qris-uq is not a valid account string.

To find out the list of all group names and members, issue the following command at the command line:

```
getent group
```

Consider the situation of a UQ researcher who is a member of two groups in additional to their primary group (qris-uq).
One groups is a special project group (ProjectXYZ) and the other is a UQ org unit based group (UQ-FACULTY-SCHOOL).
To submit computations that are under the umbrella of the project use

```
qsub -A ProjectXYZ ...
```

To submit computations that are under the umbrella of their UQ Org Unit use

```
qsub -A UQ-FACULTY-SCHOOL ...
```

Regular (Non-Interactive) Jobs

For a regular batch job, you should combine

- the resource settings and
- job task lines

into a submission script file (e.g. regular_batch_job.pbs)

Job Submission Script

```
#!/bin/bash
#
#This example will request 4 cores and 4x5GB of a compute node on Tinaroo (all nodes 24 CPUs and 120GB of usable mem
#PBS -l select=1:ncpus=4:mem=20GB
#PBS -l walltime=02:00:00

#CHANGE THIS TO YOUR UQ-FACULTY-SCHOOL group name.
#USE the groups command to find out your exact group name.
#PBS -A UQ-FACULTY-SCHOOL

#A job always starts running in your home directory $HOME.
echo "The job just started and my present location is ..."
pwd

#To get into the directory where you ran the qsub command for this job you need to
echo "This job was submitted from the PBS_O_WORKDIR directory $PBS_O_WORKDIR "
cd $PBS_O_WORKDIR
pwd

#Each job has a temporary directory created on the compute node running the job.
echo "The TMPDIR for this job is $TMPDIR so change into that directory."
cd $TMPDIR
pwd

echo "Hello World ... what else would we say?!"

echo "Now sleep for 30 minutes so we can check the job using qstat ... "
echo "otherwise it may finish before we have had a chance to look at it with qstat"
sleep 30m

#Refer to the table of job environment variables section of the RCC PBS Pro User Guide
```

Be aware that the batch system stops parsing those `#PBS` lines once it hits a shell command, so put all of your resource request settings towards to the top of the file.
Comments and blank lines are OK just not anything that looks like a command

You **must** change the -A option to a valid account string for you.

Submit Job to Batch System

Then to submit the jobs to the batch system you just need to qsub that script file:

```
qsub regular_batch_job.pbs
```

Checking Job Status

To check the job status, you just need to use the qstat command.

```
uqdgree5@tinaroo1:~/Tests/PBSPro> /opt/pbs/bin/qstat -aws1n

tinmgr2:
                                                                                  Req'd  Req'd   Ela
Job ID                          Username        Queue           Jobname         SessID   NDS  TSK   Memory Time  S Tim
------------------------------ --------------- --------------- --------------- -------- ---- ----- ------ ----- - ---
79103.tinmgr2                   uqdgree5        Short           tsepp                --     1     4   16gb 12:00 H  --
    --
99598.tinmgr2                   uqdgree5        Short           regular_batch_j    45896    1     4   20gb 02:00 R 00
```

```
   Job run at Thu Sep 05 at 14:35 on (tn425a:ncpus=4:mem=20971520kb)
96608.tinmgr2                  uqdgree5        Training        TrainingTest         --      1     1    5gb 01:00 H  -
    --
```

Figuring out WHERE your job is running

The "`-1n`" option to qstat will list the name(s) of the node(s) that are running the job when it is running.

Armed with this information, you can ssh to node(s) that is/are running your job.
In the example above, the node name is **tn425a**.
Those other details ("@/1*4") are referring to cores and their placement on the CPUs and are not part of the server name.

Please note that UQ RCC policies will only allow users to access a HPC compute server who are owners of the actively running jobs on that node.
Connections by "non-stakeholders" may appear to initially succeed however the connection will soon be dropped if you are not the owner of a running job in the compute node you are connecting to.

Figuring out WHAT your job is doing

Sometimes, you may not be sure if your job is making progress.
Sometimes, your job may be running in an imbalanced, or sub-optimal way.
Sometimes, your job may not be running correctly, at all.

It is useful to be able to hop on to the compute node that is processing your work and check that things are working as expected.
Noting the section above about "Figuring out WHERE your job is running", you can access that compute server and run some diagnostics on it.

You would ascertain the name of the node, then, from the login node, **`ssh -x NODENAME`**.
Note, you will only be permitted to do this successfully while you have a job running on a compute node.

You should not need your password to ssh from a login node to a compute node in the same HPC.
You will not be able to ssh from a login node to a compute node on a different HPC.

Note, the values of `PBS_O_WORKDIR` and `TMPDIR` will be undefined, or incorrect, in your ssh session.

Use a command like the following to ascertain the value of the **`PBS_O_WORKDIR`** variable as set in your batch job.
So if your job's $PBS_JOBID is 529717.tinmgr2

```
/opt/pbs/bin/qstat -f -F json 529717.tinmgr2 | grep -e PBS_O_WORKDIR
```

The value of **`TMPDIR`** will be (assuming the job ID is JOBID)

- **`/state/partition1/pbs/tmpdir/pbs.JOBID`** on Awoonga and Tinaroo
- **`/nvme/pbs/tmpdir/pbs.JOBID`** on FlashLite

Once logged in, there are some useful diagnostic commands that include:

Diagnostic	Purpose
`uptime`	Shows 1, 5 and 15 minute average load on the server.
`top -u $USER`	Shows the most active processes that are running and are owned by you.
`pstree -u $USER`	Shows a tree structure of the processes you have running on the server.
`ls -salh directory`	Will show the names and sizes of file(s) in a directory.
`tail -f filename`	Will show you the last lines of a file and update as the file grows.

So, if your job is logging output to a text file (in PBS_O_WORKDIR or TMPDIR), then you could check that its size is growing, and check on its contents and look for trouble.

If your job is merely writing to STDOUT and STDERR,
you can still access those because as the job is running the files that ultimately are returned to you as the job's "o" and "e" files are located on the compute node at the locations like

- `/state/partition1/pbs/spool/JOBID.OU` and `/state/partition1/pbs/spool/JOBID.ER` on Awoonga and Tinaroo
- `/nvme/pbs/spool/JOBID.OU` and `/nvme/pbs/spool/JOBID.ER` on FlashLite

Figuring out WHEN your job is behaving badly

Sometimes a job can overload a single node due IO pressure or because a multitude of processing threads have all been started on that node.
Java and python software, and MATLAB, are observed to do this on occasions because they spawn many threads.

Other situations arise for jobs that are supposed to be running evenly across, say, 4 whole nodes.
Instead a job can have been misconfigured in your job script so that it is either

- performing only 24 threads on the first of the four nodes, or
- performing 4*24 threads in just the first of the four nodes thereby overloading the node and causing the job to run much more slowly than when distributed properly across 4 nodes.

Noting the two sections above, you can ssh to each of the nodes involved with your job and run some basic diagnostics to ascertain whether your multi-node job is well balanced and behaving as it should.

Using shell environment aliases in regular batch jobs

Sometimes a software module will define an alias in the shell environment.
This alias will be a shortcut for

For many of the RCC curated Software Containers, there is an alias defined that will run the container like a single command (e.g. the rjags alias for RJAGS container)

```
#We use an alias as a short cut.
#You need to allow this to work in non-interactive settings.
#Insert this line BEFORE you issue the command, or load the software module, that defines alias(es)

#For BASH shell
shopt -s expand_aliases

#For CSH/TCSH shells
```

Where am I when my batch job runs

The key points in relation to locations when your batch job runs are as follows:

- A batch job always starts running in your home directory **`$HOME`**.
  Think of it as the job needs to ssh into your account on the node that is running your job.
  The value of the environment variable **$PBS_O_HOME** should confirm this location.

- The value of the environment variable **`$PBS_O_WORKDIR`** is always the location you were when you issued the qsub command.
  This may be the location of your input data, or perhaps the directory above your Inputs directory.

- The value of environment variable **`$TMPDIR`** is a unique location that is created for each job (or job array element).
  You own it, and only you can access it, however once the job finishes the **`$TMPDIR`** location is completely removed.
  You must copy anything you wish to keep back from **`$TMPDIR`** before the job finishes.
  Using `cp` or `mv` or `rsync` commands at the end of your batch job script should achieve this for you.

Making Maximum Use of $TMPDIR

All our HPC compute nodes have local (i.e. on-board) disk.
The storage user guide provides information about the disk capacity for each cluster.

It is nearly always preferable to utilize the local disk on a node rather than to be reading and writing furiously across the network.

You should always utilize the local disk `$TMPDIR` location in your batch jobs unless there is a valid reason not to do so.
Some reasons you might give for not using `$TMPDIR` could include:

- My job runs on multiple nodes using MPI so I cannot use `$TMPDIR`. (OK, that is a valid reason but see note below about TMPDIR for multi-node jobs)
- My job needs more space than `$TMPDIR` can provide. (OK, that is a valid reason)
- My job writes barely any output until the end of the job. (OK, that maybe valid)
- I am not interested in making my jobs run better. (soooo not valid)
- Using `$TMPDIR` is too hard. (not valid ... it isn't really)

To make use of `$TMPDIR` you need to think about the following

- where your input data resides so you can fetch a copy,
- how your main task(s) will run and access the local copy of the inputs
- where you would like to deposit your outputs as the job comes to an end.

For the purposes of illustration, it will be assumed that

- you will `qsub` the batch job in a particular directory
- your code to be run is in the sub-directory Codes
- your input files are in a sub-directory called Inputs
- your output files should be returned to a sub-directory called Outputs

There are three important steps to follow to make maximum use of the `$TMPDIR`.

1. Copy your Inputs and Codes into `$TMPDIR` from a sub-directory called Inputs
2. Run your program in `$TMPDIR`
3. Copy your outputs back from `$TMPDIR` to a sub-directory called Outputs

The simplest way to achieve these 3 steps is to

```
#1
cp -rp $PBS_O_WORKDIR/Inputs $TMPDIR
cp -rp $PBS_O_WORKDIR/Codes $TMPDIR

#2
cd $TMPDIR
#Run your code that resides in Codes/
#Make sure your program reads inputs files from current working directory (i.e. $TMPDIR on local disk) or Inputs dir
#That is, it uses relative paths (or $TMPDIR) instead of reading from /home, /30days/ or /90days/

#3
cp -rp $TMPDIR/Outputs/* $PBS_O_WORKDIR/Outputs/
```

You do not need to remove the contents of `$TMPDIR` because that will be automatically deleted when the job ends.

If you have hidden files or want to copy a subset of things you will need to modify this template.
If you do not have your inputs and outputs separated into folders, you should consider doing so for ease of working.

You might consider using the `rsync` command if you have inputs and outputs and code messed up in the same directory. This will ensure that only the files that are missing files at the destination get copied.
See the Transferring Files user guide for more about using the `rsync` command.

A Note about TMPDIR for Multi-node (MPI) Jobs

There is a TMPDIR created for the job, but it only happens on the first (rank 0) node of the nodes assigned to the job. For example,

```
-bash-4.2$ qstat -aw1n 230874.tinmgr2

tinmgr2:
                                                                                  Req'd  Req'd   El
Job ID                          Username        Queue           Jobname         SessID   NDS  TSK   Memory Time  S Ti
------------------------------ --------------- --------------- --------------- -------- ---- ----- ------ ----- - --
230874.tinmgr2                  uquser          Multiple        MyJobName          17227    4    96  400gb 168:0 R 23

-bash-4.2$ ssh -x tn417c "ls -sald /state/partition1/pbs/tmpdir/pbs.230874.tinmgr2"
0 drwx------ 2 uquser   qris-uq 6 Jun  4 11:06 /state/partition1/pbs/tmpdir/pbs.230874.tinmgr2

-bash-4.2$ ssh -x tn423d "ls -sald /state/partition1/pbs/tmpdir/pbs.230874.tinmgr2"
ls: cannot access /state/partition1/pbs/tmpdir/pbs.230874.tinmgr2: No such file or directory

-bash-4.2$ ssh -x tn425b "ls -sald /state/partition1/pbs/tmpdir/pbs.230874.tinmgr2"
ls: cannot access /state/partition1/pbs/tmpdir/pbs.230874.tinmgr2: No such file or directory

-bash-4.2$ ssh -x tn221b "ls -sald /state/partition1/pbs/tmpdir/pbs.230874.tinmgr2"
ls: cannot access /state/partition1/pbs/tmpdir/pbs.230874.tinmgr2: No such file or directory
```

So TMPDIR is still usable provided it is only the rank0 process that reads from, or writes to, the TMPDIR on the first node in the list.

Interactive Job Submissions

Batch computing systems support the seemingly contradictory notion of an interactive batch job.
What it is, is a job allocated resources and a compute node destination by the batch scheduler mechanism, but your login session is "transported" to that compute node while you wait.

Please use interactive job submissions for all but the very basic of tasks instead of burdening login nodes and annoying your fellow users!
Using the following command will launch an interactive job with 2 cores and 4GB of RAM for an afternoon of productive code compilation and testing.

```
qsub -I -l select=1:ncpus=2:mem=4GB  -l walltime=03:00:00 -A YourAccountString
```

My personal preference is to use a shell script that I can edit as necessary.

```
uqdgree5@tinaroo1:~> cd $HOME
uqdgree5@tinaroo1:~> chmod 700 bin/interact
uqdgree5@tinaroo1:~> cat bin/interact
#!/bin/bash
qsub -I -A UQ-RCC -l select=1:ncpus=2:mem=4GB  -l walltime=03:00:00
```

The run it using `$HOME/bin/interact` or if you have your bin directory in your PATH then just type `interact` .

You cannot `qsub -I` a regular PBS batch job script and have it run the commands contained therein.
To achieve something like that, you will need to use the `sleep` trick described below for getting long walltime interactivity.

When you are finished with your interactive session, you can either wait for the wall time to elapse, or you can type the command `exit` at the command prompt and this will cause your connection to the allocated compute node to close, and your session will return to the login node from where you qsub-ed.

X11 Forwarding (-X option)

The `-X` option is optional and it enables X11 forwarding. This would allow you to see graphical output generated on the compute node.
See **`man qsub`** for details.

To use this feature you must have your DISPLAY environment variable set and also have a functioning X11 connection to the login node.
See the Connecting to HPC User Guide for details.

Longer Interactive Sessions or Multi-Node Interactive jobs

Unlike our previous (torque) batch system, jobs submitted with the `qsub -I` can have job attributes that exceed the limits on the Interact queue.
What happens in this situation is that the job will route to another queue- one that can accommodate the job's resource request.
So for example,

- if you require an interactive single node job of 50 hours walltime it will route to the Single queue because the walltime exceeds 48 hours,
- if you require an interactive multi-node job (on Tinaroo or FlashLite) it will route to the Multiple queue because the job selected more than 1 node

If you would prefer not to have to maintain your connection to the login node where you launched the `qsub -I`, there is a relatively simple solution.

The solution involves the following steps:

1. Create a regular batch job that just contains a "sleep" command.
   The job would call a command like `sleep 168h`.
2. Submit the "sleeping job" to the batch system as a regular batch job.
3. Use `qstat -a1n` command to ascertain whether the job has started and if so, which node your job has been assigned to.
4. For a multi-node job, the first named node is what you want to connect with.
5. Then `ssh NODENAME` to the node your sleeping job was allocated. Add the -X option if you need X11.
6. With MPI you may need to adapt your mpirun command so it knows which nodes to use.

The command `cat $PBS_NODEFILE|sort|uniq` will give the unique hostnames.
The $PBS_NODEFILE file may contain multiple entries for the same hostname depending on the resource request made in the sleeper job.

An Important Note about MPI Jobs

Whether your MPI job runs on a single node, or multiple nodes, PBSPro sets up that environment and communicates that environment via the API to the message passing interface framework.
So if, for example, your job request was

```
#PBS -l select=4:ncpus=12:mpiprocs=12:mem=60GB
```

then further in your job script you would probably use an `mpirun` command.

Whenever you use mpirun within a PBSPro job, you should avoid specifying the number of MPI ranks to use.
That is, do **not** include a **`-np 48`** in the mpirun command line in the example MPI job stated above.
Not only is it unnecessary, but it may actually be counterproductive as it may confuse the MPI library.

Just say
`mpirun program-name ...`
and leave it up to the batch system and the MPI library to sort out the "-np stuff".

Tips for Creating More Resilient Job Scripts

If you put to one side the #PBS directives, a PBSPro job script is just a unix shell script.
In its simplest form it is a sequence command lines to be executed (as illustrated in the earlier sections).

However you can, and it could be argued, should, be making your job scripts more resilient so that they can withstand, or better handle, situations that sometimes arise when a job runs. The way to do that is to take advantage of features of the shell environment.
The following is based on the "Bourne Again SH" /bin/bash shell which is what the vast majority of HPC users use as their login shell.

A significant tool for this is the `test` command aka the "[ ]" operator.
I normally use it with as the [] operator within an if statement.
You can negate the test using the ! symbol

If your job needs to work with an input file from RDM, it is useful to

1. check that the directory is available `if [ ! -d path_to_directory ] ; then ...` take action if directory is not mounting
2. check that the file is available `if [ -f full_file_path ] ; then ...` proceed
3. ensure that the file is ondisk `/usr/local/bin/recall_medici full_file_path` refer to storage user guide section about RDM

If your job script contains steps or stages and a latter stage needs an earlier stage to have been completed successfully, you should pay attention to the exit codes.

```
#do something
run_first_thing
#check the exit code
if [ $? -eq 0 ]; then
  run_second_thing
fi
```

OR abbreviated shell syntax to perform them together conditionally

```
#only do second if the first is successful
run_first_thing && run_second_thing

#only do the second if the first is unsuccessful
run_first_thing || run_second_thing
```

Utilising Exit Codes in Job Scripts

Another very useful thing you can do is to catch the various failure modes of your job script and exit your script immediately with an informative exit code.
The job "e" will then record the nature of the failure as the exit status for the job.
Job related exit codes need to positive and non-zero.
An exit code of 0 means the job was successful, and a negative error code will occur if the error was server related.

PBSPro Job Environment Variables

The following table is extracted from the PBSPro Reference Guide Table 13-1: PBS Environment Variables

Environment Variable	Meaning
NCPUS	Number of threads or CPUs
OMP_NUM_THREADS	Same as NCPUS (for shared memory jobs)
PBS_ARRAY_ID	Identifier for job arrays. Consists of sequence number and []
PBS_ARRAY_INDEX	Index number of subjob in job array
PBS_CONF_FILE	Path to pbs.conf
PBS_CPUSET_DEDICATED	Asserts exclusive use of resources in assigned cpuset
PBS_DEFAULT	Name of default PBS server
PBS_DATA_SERVICE_USER	Account used by data service
PBS_ENVIRONMENT	Indicates job type: PBS_BATCH or PBS_INTERACTIVE
PBS_JOBCOOKIE	Unique identifier for inter-MoM job-based communication
PBS_JOBDIR	Pathname of job-specific staging and execution directory
PBS_JOBID	The job identifier assigned to the job or job array by the batch system
PBS_JOBNAME	The job name supplied by the user
PBS_MOMPORT	Port number on which this job's MoMs will communicate
PBS_NODEFILE	The filename containing a list of vnodes assigned to the job
PBS_NODENUM	Logical vnode number of this vnode allocated to the job
PBS_O_HOME	Value of HOME from submission environment
PBS_O_HOST	The host name on which the qsub command was executed
PBS_O_LANG	Value of LANG from submission environment
PBS_O_LOGNAME	Value of LOGNAME from submission environment
PBS_O_MAIL	Value of MAIL from submission environment
PBS_O_PATH	Value of PATH from submission environment
PBS_O_QUEUE	The original queue name to which the job was submitted
PBS_O_SHELL	Value of SHELL from submission environment
PBS_O_SYSTEM	The operating system name where qsub was executed
PBS_O_TZ	Value of TZ from submission environment
PBS_O_WORKDIR	The absolute path of directory where qsub was executed
PBS_QUEUE	The name of the queue from which the job is executed
~~**PBS_SERVER**~~	~~The name of the default PBS server.~~
PBS_TASKNUM	The task (process) number for the job on this vnode
TMPDIR	The job-specific temporary directory for this job