What's the difference between a resource manager and a scheduler?

A resource manager is responsible for managing the resources available in the cluster. Resources include hardware such as the node itself, the CPUs, memory, but can also be more abstract like network bandwidth or software available only on certain nodes. The resource manager also defines the queues for the cluster and handles job startup and shutdown, but does not determine which jobs to start or stop. The scheduler is in charge of analyzing the running jobs and jobs wanting to run. The scheduler applies the scheduling policy for the cluster to determine what jobs to start and stop based on the information the resource manager reports back about resource availability. Once the scheduler has determined what job, if any, to run next, it instructs the resource manager to start the job on a specific node or set of nodes. The scheduler monitors running jobs to ensure no job exceeds its allotted time or resources and if needed instructs the resource manager to kill jobs. The scheduler is also in charge of reservations.

In short, the resource manager is in charge of the physical aspects of the cluster and the scheduler is in solely in charge of job administration and the two work hand in hand to ensure proper utilization based on the policy for the cluster. Refer to the scheduling FAQ for more information on how the scheduler creates and applies scheduling policy.

What is a PBS job script?

You can place all Torque job submission options into a file instead of specifying them on the command line. This can allow you to do more complex actions such as setting more environment variables before your job runs or by doing things like starting mpda. At it's simplest it looks like:

```bash
#!/bin/bash
/bin/env
```
If you need to specify options to `qsub`, you place them one per line prefixed by `#PBS` like:

```bash
#!/bin/bash
#PBS -N myjob
/bin/env
```

### How do I submit a batch job?

If you saved your job script as `myjob.pbs` you would submit like:

```bash
$qsub myjob.pbs
```

This would submit a job to the default queue using the default resource limits for that queue. In this example it would simply run the `/bin/env` command on the head node. More complex examples are given in this FAQ.

### What is the "head" node?

If you request more than one node for your job, Torque delegates one of the nodes assigned to you as the "head". If you submitted an interactive job, this is the node you are logged into. Otherwise, this is the node your actual job executable runs on. It is your responsibility to then ensure your job is properly run on the rest of the nodes - Torque does not run your executable on every node only the head node. For example, if you submitted the above example to 2 nodes, `/bin/env` would only be run on the head node. The other node would sit idle. Usually a parallel framework, such as MPI, handles proper job startup on the rest of the node. In MPI, the head node has a rank of 0.

### Where is my job output sent?

By default, job output is sent to a local spool filesystem on the head node and is copied to your home directory after the job completes. The default name for these files is `<script name>.o$JOBID` for standard out and `<script name>.e$JOBID` for standard error where `$JOBID` is the ID assigned to your job at submission time. You can override these names by using the `-N` option of `qsub`:

```bash
$qsub -N me myjob.pbs
```

Then the name of these files would be `me.o$JOBID` and `me.e$JOBID` for standard out and standard error, respectively.

### How do I see what queues are available on a cluster and what limits they might have?

Use the `qstat` `-q` command:

```bash
$qstat -q
```

```
server: tp-mgt

<table>
<thead>
<tr>
<th>Queue</th>
<th>Memory</th>
<th>CPU</th>
<th>Time</th>
<th>Walltime</th>
<th>Node</th>
<th>Run</th>
<th>Que</th>
<th>Lm</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>extended</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>57</td>
<td>0</td>
<td>--</td>
<td>--</td>
<td>E R</td>
</tr>
<tr>
<td>batch</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>0</td>
<td>0</td>
<td>--</td>
<td>--</td>
<td>D R</td>
</tr>
<tr>
<td>test</td>
<td>--</td>
<td>--</td>
<td>00:10:00</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>--</td>
<td>--</td>
<td>E R</td>
</tr>
<tr>
<td>fast</td>
<td>--</td>
<td>--</td>
<td>01:00:00</td>
<td>--</td>
<td>0</td>
<td>0</td>
<td>--</td>
<td>--</td>
<td>E R</td>
</tr>
<tr>
<td>special</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>--</td>
<td>--</td>
<td>D S</td>
</tr>
<tr>
<td>long</td>
<td>--</td>
<td>--</td>
<td>24:00:00</td>
<td>--</td>
<td>0</td>
<td>0</td>
<td>--</td>
<td>--</td>
<td>E R</td>
</tr>
<tr>
<td>short</td>
<td>--</td>
<td>--</td>
<td>04:00:00</td>
<td>--</td>
<td>0</td>
<td>0</td>
<td>--</td>
<td>--</td>
<td>E R</td>
</tr>
</tbody>
</table>

57  0
```

Here you see 7 queues, each with various limitations and in different states. For more information on what each column represent, please refer to the `qstat` manual page.

If you try to submit a job that exceeds the queue limits, `qsub` will immediately fail.
Job Submission Parameters

How do I submit to a queue other than the default?

Provide the -q option to qsub:

$ qsub -q short myjob.pbs

How do I request a different job duration than the default for the queue?

Every queue will have a default job duration if none is specified. If your job requires more or less than this you would use the -l walltime parameter to qsub:

$ qsub -l walltime=01:00:00 myjob.pbs

The above command states that the job should take no longer than one hour. The walltime parameter format is HH:MM:SS (hours, minutes, seconds). Keep in mind that if your job executable finishes before the walltime is reached (as should happen in the above example), your job will not continue to run until your walltime is reached - once your job executable returns, successfully or not, your job ends. Also keep in mind that some queues have walltime limits. For instance, a queue may not allow jobs with a walltime greater than one hour.

How do I request more than one node?

Use the -l nodes=# parameter to qsub:

$ qsub -l nodes=2 myjob.pbs

The above command requests two nodes for your job. Remember that it is your responsibility to make sure all nodes are utilized. Also, like walltime, some queues have limits on the amount of nodes you can request. For instance, a queue may not allow jobs requesting more than one node.

How can I request more than one processor per node?

For MPI jobs it can be beneficial to use as many processors per node as possible. To do this specify the -l nodes=#:ppn=# parameter to qsub:

$ qsub -l nodes=1:ppn=2 myjob.pbs

This would request two processors on one node using the default walltime for the default queue. Note that you have to specify a number of nodes when specifying ppn. Again, it is up to you to make sure those processors get utilized.

How do I put all these job parameters together?

Here is an example of submitting a 2 node job with 2 processors per node requiring 10 minutes of walltime and using the short queue:

$ qsub -l nodes=2:ppn=2,walltime=00:00:10 -q short myjob.pbs

or if you were to put this all in the job script file it would look like:

#!/bin/bash
#PBS -l nodes=2:ppn=1,walltime=00:00:10
#PBS -q short
/bin/env

and you would simply do:

$ qsub myjob.pbs
Can Torque email me about certain events regarding my job?

By default you are notified via email only when your job is aborted or terminated abnormally, but you can alter when you are notified by specifying the \(-m\) option to \(qsub\):

\[\text{n Do not notify via email} \]

\[
\$ \ qsub \ -m \ n \ \text{myjob.pbs}
\]

or combine any of the following:

\[\text{a Mail is sent when the job is aborted by the batch system (default)} \]
\[\text{b Mail is sent when the job begins execution} \]
\[\text{e Mail is sent when the job terminates} \]

\[
\$ \ qsub \ -m \ ab \ \text{myjob.pbs}
\]

Projects

How do become a member of a project?

You can request access to Beagle cluster on the HPC Projects page.

How do I see what projects I'm a member of?

From a login node of one of the clusters you issue the \(\text{projects}\) command:

\[
\$ \ \text{projects --available}
\]

The following projects are available for your use

<table>
<thead>
<tr>
<th>Project</th>
<th>PI</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>CI-STA000015</td>
<td>Beth Cerny</td>
<td>CI Media</td>
</tr>
</tbody>
</table>

How do I see who is a member of a given project?

\[
\$ \ \text{getent group CI-STA000015}
\]

CI-STA000015:*:1084:grog,leggett,sneedham,forero,hereld,insley,fdech,bcerny,nmp

How do I see what my default project currently is?

Use the \(\text{projects}\) command from a login node of one of the clusters. If you have a default project for the cluster you're currently on the output will look like:

\[
\$ \ \text{projects}
\]

Your default project for this cluster is CI-STA000015

If you don't have a cluster default set but do have a default project for all CI clusters, the output will look like:

\[
\$ \ \text{projects}
\]

Your default project for all CI clusters is CI-STA000015
How do I set a default project to use?
You either set a default project for all CI clusters or for just the cluster you're currently logged into. To set your default project for a specific cluster, login to that cluster and use the `projects` command:

```
$ projects --set CI-STA000015
Your default project for this cluster has been set to CI-STA000015.
```

To set your default project for all CI clusters use the `--all` option:

```
$ projects --set CI-STA000015 --all
Your default project for all CI clusters has been set to CI-STA000015.
```

How can I override my default project?
You can override what project you use for job submission using the `-A` option to `qsub`:

```
$ qsub -A CI-STA000015 myjob.pbs
```

Job Status

How do I see all the jobs I have in the queue?
You use the `qstat` `-u` command:

```
$ qstat -u
```

```
tp-mgt.ci.uchicago.edu:
Job ID               Username Queue    Jobname          SessID NDS   TSK Memory Time  S Time
-------------------- -------- -------- ---------------- ------ ----- --- ------ ----- - ----- 
854022.tp-mgt.ci     leggett  extended STDIN             21535     1  --    --  75:00 R  --
```

How can I find out what nodes my job was given?
You use the `qstat` `-n` command:

```
$ qstat -n
```

```
tp-mgt.ci.uchicago.edu:
Job ID               Username Queue    Jobname          SessID NDS   TSK Memory Time  S Time
-------------------- -------- -------- ---------------- ------ ----- --- ------ ----- - ----- 
854022.tp-mgt.ci     leggett  extended STDIN             21535     1  --    --  75:00 R 00:01
```

How can I find out everything I can about my job?
Use the `qstat` `-f` command:
How do I delete a job?

In order to delete a job first find the job ID number.

$ qstat -u leggett

tp-mgt.ci.uchicago.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>Memory</th>
<th>Time</th>
<th>Start Time</th>
<th>Start Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>854022.tp-mgt.ci</td>
<td>leggett</td>
<td>extended</td>
<td>STDIN</td>
<td>21535</td>
<td>1</td>
<td>--</td>
<td>--</td>
<td>75:00</td>
<td>Wed Feb 11 13:56:25</td>
<td>1</td>
</tr>
<tr>
<td>854023.tp-mgt.ci</td>
<td>leggett</td>
<td>extended</td>
<td>STDIN</td>
<td>21535</td>
<td>1</td>
<td>--</td>
<td>--</td>
<td>75:00</td>
<td>Wed Feb 11 13:56:25</td>
<td>1</td>
</tr>
<tr>
<td>854024.tp-mgt.ci</td>
<td>leggett</td>
<td>extended</td>
<td>STDIN</td>
<td>21535</td>
<td>1</td>
<td>--</td>
<td>--</td>
<td>75:00</td>
<td>Wed Feb 11 13:56:25</td>
<td>1</td>
</tr>
</tbody>
</table>

To delete a job you use the `qdel` command followed by the job ID:

$ qdel 854023

There is no output when `qdel` successfully deletes a job. If the node your job was running on has died, you may need to contact an administrator to delete the job.
How do I delete all my jobs?
To delete all your jobs simply provide the word all instead of a job ID:

$ qdel all

Common Errors

"Warning: no access to tty" in job output
If your shell is tcsh you will see the following in your batch job output:

Warning: no access to tty (Bad file descriptor).
Thus no job control in this shell.
This is a 'feature' of tcsh. You can either ignore the warning or switch your shell to bash or zsh.

Advanced Torque Topics

What environment variables does Torque set that might be useful?
PBS_NODEFILE file containing line delimited list on nodes allocated to the job
PBS_TASKNUM number of tasks requested
PBS_ARRAYID zero-based value of job array index for this job
PBS_NODENUM node offset number
PBS_JOBID unique pbs job id
PBS_JOBNAME user specified jobname
PBS_O_WORKDIR job's submission directory
PBS_O_HOME home directory of submitting user
PBS_O_LOGNAME name of submitting user
PBS_O_HOST host on which job script is currently running
PBS_QUEUE job queue

I have an executable that I need to run many times with different parameters. Is there an easy way to do this? (Task Arrays)
This type of job is called a parameter sweep. Torque has a method, called task arrays, for making parameter sweeps more manageable. You can specify a set or range of indexes Torque uses to generate multiple jobs from one submission. You do this using the -t option to qsub. To best illustrate this, here is a simple example of calculating the squares of the integers between 1 and 8. Here is a job script that accomplishes this:

#!/bin/bash
#PBS -l nodes=1
#PBS -t 1-8
echo $(PBS_ARRAYID*PBS_ARRAYID)

You can see that we are requesting one node and a task array from 1 to 8. An important note is that Torque creates a job for each task. That means that if resources are available, all tasks will run at the same time and operate independent of each other. If you requested -l nodes=5:ppn=2 then each task will require 5 nodes and 2 processors per node, so request resources as you would for a single task to run. In other words if you have 8 tasks as above and each task only needs 1 node, do not request -l nodes=8. Second, once a task has started, Torque sets the environment variable, $PBS_ARRAYID, to the task ID running. So when task 1 starts, $PBS_ARRAYID is set to 1, and when task 5 starts, $PBS_ARRAYID is set to 5. Lastly, when Torque starts the task, the job ID has the format JOBID[$PBS_ARRAYID]. So if your job ID is 854151 and the running task is 1, your full job ID would be 854151[1].

At first this may not seem terribly useful, but what you can do is predefine your parameters for each task in flat file or even a database. You can then lookup what parameters your code should run with based on the $PBS_ARRAYID. This is useful especially if you need to start thousands of jobs and even more useful if you need to restart only certain of those jobs because of failures. You only need know what task IDs failed to restart them.

Is there job scratch space available?
Where should my job save its data?
Depending on the cluster, there are multiple type of scratch filesystems available. The type you use, depends on your job's needs. Please refer to the data management FAQ for more information.