Running jobs on the NYU HPC clusters

Quick Links

HPC at NYU
HPC Newsletter
Getting started on Prince
Getting an account
Prince How-to Articles
Logging in
Windows
Mac / Linux
Clusters and Storage
Mercer (HPC, retired)
Prince (HPC)
Dumbo (Hadoop)
Dalmata (NYU Abu Dhabi)

Transferring data to/from the clusters

Transferring data to/from Prince cluster using Globus

Submitting jobs with sbatch

Available software

Building Software Tutorials

Programming for Biologists

Research Gallery

FAQs

NYU HPC Cheat Sheet

Wiki Map

HPC Policies
Running jobs on the NYU HPC clusters

Job Scheduling Overview

Login and compute nodes

Queues

Writing a job script

Job stdout and stderr

Which filesystems should I use?

Submitting a Job

Basic qsub options and directives

Setting resource limits

Running MPI jobs

Running GPU jobs

Running R jobs

Running Matlab, Gaussian, etc jobs

Working interactively
Working on the HPC clusters is not the same as working at a desktop workstation: in order to provide high performance computing to many users simultaneously, computational work must be packaged into a job - a script specifying what resources the job will need and the commands necessary to perform the work - and submitted to the system to be run without further input from the user. The system then schedules and runs the job on a dedicated portion of the cluster. Note that there is a way to work interactively within this model, for work which cannot be scripted, such as debugging.

**Job Scheduling**

On the NYU clusters, Torque and Moab manage the running and scheduling of jobs. As a user you will interact...
mostly with Torque, which accepts and runs job scripts and manages and monitors the cluster's compute resources. Moab does the heavy thinking: the planning of which job should be run where and when.

Avoid requesting vastly more CPUs, memory or walltime than you actually need. Jobs needing fewer resources are easier to schedule - in our scheduling diagram, a job requiring just 1 CPU for 1 hour could be inserted into the gap on Node 1 CPU 4. Smaller jobs are also more likely to receive priority when being scheduled.

Note that a small overestimate, such as 10%-20%, is wise, lest your job run out of time and be killed before it finishes, but requesting several times what you need will result in longer queueing time for your job and less efficient system utilization for everybody.

Login and Compute Nodes

Note that certain filesystems are visible to the login or compute nodes but not both: specifically at NYU /archive is not visible to the compute nodes, while /state/partition1 is visible and local only to individual compute nodes.

Important
Do not run computationally-heavy or long-running jobs on the login nodes! Not only will you have poor performance, the heavy resource usage of such jobs impacts others ability to use the login nodes for their intended purposes. If you need to run a job interactively (for example, when debugging), please do so through an interactive batch session.

Queues

Not all jobs can be run at once - the cluster is finite! - so when jobs are submitted they are placed into a queue. When a "space" becomes available in the schedule Moab looks down the queue for the first job that will fit into the space.

Jobs are not necessarily placed at the end of the queue - Moab uses the priority (discussed here) to determine where in the queue a job should be placed.

At NYU HPC shorter jobs are given higher priority

There is more than one queue. Each queue is configured for different types of jobs and has resource limits and priorities set accordingly. If you do not specify a queue to submit to, Torque will use the resources requested to select a queue for you. Frequently this is the best option, however in some circumstances you are better off explicitly specifying a queue.

You can see the list of queues with the command "qstat -q", and you can see more detail about a specific queue with "qstat -Qf queue-name".

(more...)
Writing a Job Script

(more...)

Submitting a Job

Jobs are submitted with the qsub command:

```
$ qsub options job-script
```

The options tell Torque information about the job, such as what resources will be needed. These can be specified in the job-script as PBS directives, or on the command line as options, or both (in which case the command line options take precedence should the two contradict each other). For each option there is a corresponding PBS directive with the syntax:

```
#PBS option
```

For example, you can specify that a job needs 2 nodes and 8 cores on each node by adding to the script the directive:

```
#!/bin/bash
#PBS -l nodes=2:ppn=8
```

or as a command-line option to qsub when you submit the job:

```
$ qsub -l nodes=2:ppn=8 my_script.q
```

(more...)

Monitoring Jobs

To see the status of a single job - or a list of specific jobs - pass the Job IDs to qstat, as in the following example:

```
$ qstat 3593014 3593016
```

<table>
<thead>
<tr>
<th>Job id</th>
<th>Name</th>
<th>User</th>
<th>Time Use</th>
<th>S Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>3593014</td>
<td>model_scen_1</td>
<td>ab123</td>
<td>7:23:47 R s48</td>
<td></td>
</tr>
<tr>
<td>3593016</td>
<td>model_scen_1</td>
<td>ab123</td>
<td>7:23:26 R s48</td>
<td></td>
</tr>
</tbody>
</table>

Most of the fields in the output are self-explanatory. The second-last column "S" is the job status, which can be:

- Q meaning "Queued"
- H meaning "Held" - this may be the result of a manual hold or of a job dependency
- R meaning "Running"
- C meaning "Completed". After the job finishes, it will remain with "completed" status for a short time before being removed from the batch system.

Other, less common job status flags are described in the manual (man qsub).

The program pbstop, available on the login nodes, shows which jobs are currently running on which nodes and cores of a cluster.

Jobs belonging to a single user can be highlighted by launching pbstop with the -u switch:

```
pbstop -u <NetID>
```

(of course, replace <NetID> with your NYU NetID). Or, you can use the alias "me":

```
pbstop -u me
```

When you start pbstop you see something like the annotated screenshot below. You might need to resize your terminal to make it all fit:
Canceling a Job

To kill a running job, or remove a queued job from the queue, use `qdel`:

```
$ qdel jobid
```

To cancel ALL of your jobs:

```
$ qdel all
```