The simplest queuing algorithm is "first come, first served". The queuing of jobs on the HPC cluster is a little more sophisticated as we pursue several goals:

- Mininal queuing times, especially for short jobs. Nobody wants to spend 4 hours in the queue for a 1-hour job.
- Efficient use of the available resources. If there is a job ready which can use hardware that would otherwise be idle, run it, even if it's not next in the queue.
- Fair use of resources. If you've made heavy use of the cluster recently, jobs belonging to a user who has had less CPU time will get higher priority.
  At NYU "recently" means "the last 24 hours", so users with large workloads are not excessively penalized.

Moab supports these goals by calculating a priority for each submitted job and placing the job in the queue according to its priority. The schedule of which job will run where and when is built from the job queues. When a job finishes earlier than scheduled (due to an overestimated walltime request), Moab attempts to fill the newly-available space by scanning the queue for the first job which will fit without delaying an already-scheduled, higher priority job. In this way low-priority jobs with smaller resource requirements can jump ahead and be run early.

Monitoring jobs with qstat

The qstat command is described in more detail here.

What is running on the cluster, and where? Interpreting pbstop

When will my job start?

You can get an estimate of the scheduled starting time for a job with showstart:
$ showstart 3546761

job 3546761 requires 12 procs for 15:00:00

Estimated Rsv based start in     00:43:58 on Tue Jul  4 15:04:56
Estimated Rsv based completion in 15:43:58 on Wed Jul  5 06:04:56

Best Partition: torque

Note that showstart is based on the scheduled time - which might be adjusted as other jobs are added to the queue and if already-running jobs finish ahead of time.

Also, if you've only just submitted the job, the scheduler might not have seen it yet. Moab only collects new jobs to schedule every ~15 seconds.

### Setting job priorities

If you have several jobs in the queue, and would like certain of them to be prioritized over others, you can set the relative priority of a job by submitting it with:

```
$ qsub -p priority job-script
```

Here:

- `priority` is a number between -1024 and +1023. A higher number means higher priority. The default priority is 0.
- `job-script` is the name of your job script

You can also pass `-p` as a PBS directive within your job script:

```
#PBS -p priority
```

For more about `qsub` and PBS directives, see [Writing and submitting a job](#).

### Why hasn't my job started?

You can get information about what is preventing a queued job from running with `checkjob`:

```
$ checkjob jobid
```

The output of `checkjob` is complicated and technical. Mostly a job remains in the queue because it is waiting for resources to become available (you can check how busy the system is with `pbstop`). Other likely causes are that it is waiting on a job dependency, or you have reached the limit of simultaneously running jobs for a single user. If your job has been waiting a long time and you would like help understanding why, contact us.

In the example below the job requested 12 large-memory nodes, and the blue text on the last line indicates that the scheduler has not yet found a large enough timeslot slot in which it can run (note that it has found four such nodes available).
$ checkjob 3718378
job 3718378
AName: testme.q
State: Idle
Creds: user:sl151 group:users account:ITS class:p12 qos:p12
WallTime: 00:00:00 of 00:01:00
BecameEligible: Thu Feb 13 12:47:51
SubmitTime: Thu Feb 13 12:47:46
(Time Queued Total: 00:00:10 Eligible: 00:00:04)
NodeMatchPolicy: EXACTNODE
Total Requested Tasks: 24

Req[0] TaskCount: 24 Partition: ALL
Opsys: --- Arch: --- Features: mem48gb
Dedicated Resources Per Task: PROCS: 1 MEM: 1365M

Notification Events: JobFail
IWD: /home/sl151/batch_scheduler
Flags: RESTARTABLE
Attr: checkpoint
StartPriority: 1999
compute-9-0 available: 12 tasks supported
compute-9-4 available: 12 tasks supported
compute-9-7 available: 12 tasks supported
compute-9-13 available: 12 tasks supported

NOTE: job cannot run in partition crunch (insufficient idle nodes available: 4 < 12)