Running jobs - MPI

<table>
<thead>
<tr>
<th>Quick Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPC at NYU</td>
</tr>
<tr>
<td>HPC Newsletter</td>
</tr>
<tr>
<td>Getting started on Prince</td>
</tr>
<tr>
<td>Getting an account</td>
</tr>
<tr>
<td>Prince How-to Articles</td>
</tr>
<tr>
<td>Logging in</td>
</tr>
<tr>
<td>Windows</td>
</tr>
<tr>
<td>Mac / Linux</td>
</tr>
<tr>
<td>Clusters and Storage July 2017</td>
</tr>
<tr>
<td>Mercer (HPC, retired)</td>
</tr>
<tr>
<td>Prince (HPC)</td>
</tr>
<tr>
<td>Dumbo (Hadoop)</td>
</tr>
</tbody>
</table>
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
### Job dependencies and delaying starting

Running many similar jobs

Setting job priorities

Where in the queue is my job, and why?

### Monitoring jobs with `qstat`

```plaintext
qs
tat
immore
det
ail
```

See what is running where with `pbsstop`

When will my job start?

Why hasn't my job started?

### Canceling a job

### MPI jobs

For MPI parallel jobs you must specify how many cpus you need:
## use 4 CPUs, I don't care if they are on the same node or not
#PBS -l procs=4

## use 4 CPUs all on the same node (eg, for an OpenMP job with no MPI):
#PBS -l nodes=1:ppn=4

## use 4 CPUs, in groups of 2-on-the-same-node (all three groups might land on the same physical node)
#PBS -l nodes=3:ppn=2

## use 2 CPUs on each of 2 distinct nodes. I really need them to be on different nodes!
#PBS -l nodes=2:ppn=2 -W x=nmatchpolicy:exactnode

Running MPI jobs with OpenMPI and MVAPICH2

Mercer has two implementations of MPI: OpenMPI and MVAPICH2. Overall the performance and behavior of each is similar, but with either there are special things to consider:

- With OpenMPI, processes may change which CPU (within the node) they use during execution. This can reduce performance, so OpenMPI jobs should be started with the `--bind-to-core` flag:

  mpirun --bind-to-core -np $PBS_NP ./my_program

- With either MPI implementation, the default behavior is to launch each process on the next available CPU. For hybrid MPI/OpenMP programs, this causes incorrect behavior, as each MPI process needs multiple CPUs for its OpenMP threads.

  For example, if you need two MPI processes with 2 OpenMP threads each, you would request `-l nodes=2:ppn=2`. Your job might then be allocated 2 CPUs on node 14-0 and 2 CPUs on node 14-1. Your nodefile ($PBS_NODEFILE) will then look like:

```
compute-14-0.local
compute-14-0.local
compute-14-1.local
compute-14-1.local
```

  MPI uses this to place its processes, so it will, by default, place the first process on `compute-14-0.local` and the second process on `compute-14-0.local`. But for a hybrid program, you have multiple threads (in this case, 2) for each MPI process. Threads cannot cross host boundaries, so you will have 4 threads sharing the 2 CPUs of `compute-14-0.local`, while `compute-14-1.local` sits idle.

  To get the correct behavior with OpenMPI, add `--bynode` to the mpirun command:

  mpirun --bind-to-core --bynode -np $PBS_NP ./my_program

  With MVAPICH2, you must use a modified nodefile to reserve CPUs for OpenMP threads. The following idiom is effective:
#!/bin/bash
#PBS -l nodes=2:ppn=20

cd $SCRATCH/my_run_dir
module load mvapich2/intel/2.0rc1

# we want 20 threads per MPI process:
export OMP_NUM_THREADS=$PBS_NUM_PPN

# make a new nodefile with only two entries, one for each MPI process:
sed -n "1~${PBS_NUM_PPN}p" $PBS_NODEFILE > my_nodefile
mpirun -f my_nodefile -np $PBS_NUM_NODES my_program