Running Jobs on the COC-ICE PACE Cluster

The easiest way to run your jobs on the COC-ICE PACE cluster is via a PBS configuration file which specifies different job parameters, such as the number of nodes or cores required. These files are similar to to shell script files and pass parameters to the job scheduler which have been specified in the form of directives tagged with `#PBS`.

General Job Config Info

The following config parameters should be in every PBS file regardless of the type of job:

```
#PBS -l pmem=1gb
```

- This will reserve 1 GB of RAM per node for your job, which should be a sufficient amount for your barrier implementation.

```
#PBS -l walltime=00:05:00
```

- This will limit the max runtime of your job to 5 minutes. In general your barrier implementation should complete in a minute or two, but if it ends up running for 5 minutes or more something is probably wrong (e.g., a deadlock). Setting this value ensures you won't encounter a runaway process that dominates the cluster and won't waste too much of your time waiting on buggy code to time out.

```
#PBS -q coc-ice
```

- This ensures your job will run in the `coc-ice` queue (which is the only queue you should be using)

```
#PBS -j oe
```

- This ensures any output from your implementation and any errors will be captured

```
PBS -o job.out
```

- This will direct errors and output from your job to a file named 'job.out'. Note that you probably want to change 'job.out' to something more descriptive so you can keep track of different log files.

OpenMPI Configs

To ensure you get accurate numbers for your evaluations, your MPI jobs will need to be run on multiple distinct cluster nodes. By default, COC-ICE cluster's scheduler will place jobs in nodes based on the concepts of "chunks" of work, where a "chunk" represents a per-core division. This means that if you rely on the scheduler alone (i.e., if you aren't explicit in your config file directives) your MPI job might get scheduled on a single node. For example, if you only specify `-l nodes=8:ncpus=1:mpiprocs=1` in your PBS config, it's possible the scheduler will simply place all your jobs on a single node that has 8 cores available. If this happens, the measurements you take for barrier synchronization won't make much sense during evaluation since they're all running in the same place instead of on multiple network-connected machines.

To force the scheduler to always use disitinct nodes when running MPI jobs, you'll need to explicitly tell it which nodes to use via a PBS config file. For example, a config file with the line:

```
#PBS -l nodes=rich133-h35-16-l.pace.gatech.edu+rich133-k40-17.pace.gatech.edu
```

would ensure the scheduler runs your MPI job on the two nodes specified.

We are providing a list of available cluster nodes (in the file *nodes.list*) and an example PBS config which shows how to set up a job using 8 nodes. You will need to adapt this config file to work with jobs scaled from 2 to 12 MPI processes. For example, if you were going to run a 4-node job you would select 4 random nodes from the list and place them in your config file.

Note that you should ideally select random nodes from the list to increase the chance that you won't need to contend for cluster resources with other students. If everyone simply selects the first *X* nodes from the list for use in their configs then the queue wait times on the cluster will be longer than usual.

OpenMP Configs

Since your OpenMP implementation only needs to run on one node, you shouldn't need to specify a list of nodes in the config file. You'll need to scale your OpenMP implementation from 2-8 threads, and each node in the cluster has at least 8 cores available, so allowing the scheduler to randomly choose a single node for you should be fine.

Sample Configs

The following sample configs/implementations are included with this file:

- **8-node.pbs**: A config file for running an OpenMPI job across 8 nodes
- **8-thread.pbs**: A config file for running an OpenMP job across 8 threads
- **2-node-8-thread.pbs**: A config file for running a combined OpenMP/MPI job across 8 threads on 2 nodes (4 threads per node)
- **mpi_hello.c**: A *Hello, world!* OpenMPI implementation you can use to test PBS configs for OpenMPI
- **mp_hello.c**: A *Hello, world!* OpenMP implementation you can use to test PBS configs for OpenMP
- **combined_hello.c**: A *Hello, world!* combined OpenMP/MPI implementation you can use to test PBS configs for combined OpenMP/MPI programs

Compiling Code

Because your development environment likely differs from that of COC-ICE, you will need to recompile your source code on the cluster before you can run your tests. You will need to add a line to your PBS config file(s) to instruct the scheduler to compile your code before beginning the job. If you need to perform a standalone compilation (e.g. to test that the compile works properly before submitting a job), you can use commands similar to the examples below Note that **you should not develop on the cluster**. Develop and test your code locally, then upload your implementation source code to the server.

Copy Your Files to coc-ice

```
scp hello.c gtid123@coc-ice.pace.gatech.edu:/nv/coc-ice/gtid123/
```

(Replace 'gtid123' with your account name)

OpenMPI

```
/usr/local/pacerepov1/openmpi/1.8/gcc-4.9.0/bin/mpicc -g -Wall mpi_hello.c -o mpi_hello
```

OpenMP

```
gcc -fopenmp mp_hello.c -o mp_hello
```

Combined OpenMPI/OpenMP

```
/usr/local/pacerepov1/openmpi/1.8/gcc-4.9.0/bin/mpicc -Wall -lm -fopenmp -lgomp combined_hello.c -o combined_hello
```

Example Workflow for Running Evals

For running your evaluations on the cluster, you might use a workflow similar to the following:

1. Create a directory in your homedir on coc-ice for the evaluation you want to run (e.g., `cd ~ && mkdir mpi_hello && cd mpi_hello)
2. Create one or more a PBS config files specific to your evaluation (you may need to create multiple files which specify different numbers of nodes/cores)
3. Upload your .c and .pbs files via scp to the directory you created
4. Login to coc-ice, change to the directory you created
5. Run your job via `qsub mpi_hello.pbs` (where 'mpi_hello.pbs' is the name of the config file you created)
6. You can check the status if your job via `qstat -u gtid123 -n` (where 'gtid123' is your GT username)
7. When the job completes, its output will appear in the same directory with the file name you specified in the PBS config