Auburn University
High Performance and Parallel Computing

HPC Training: MPI Fundamentals

LAB EXERCISES

Prerequisites

- Valid Hopper HPC account
- Laptop connected to the AU wireless network
- Secure shell client, with connection settings for Hopper:
    - Linux and Mac OS users use your favorite terminal app
    - PuTTY (putty.org) is recommended for Window users

Reference

- Hopper User Guide: http://aub.ie/hug
- Slides: http://aub.ie/hpctrain
- Code: http://aub.ie/mpi
- Lynda.com

Lab Overview

-

I. Getting Started

Reminder: command syntax is:

```
$ <command> <required> [optional]
```

The dollar sign ($) is a symbol for the command prompt, which is a way for Linux to tell you it is ready to accept a command. You do not need to type the first $ you see for each command. Wherever you see $, you should enter the command that immediately follows it.

Windows users can use a terminal emulation program like SecureCRT or PuTTY. To connect, you will create a new server profile for hopper.auburn.edu, using your normal AU username and password for the login credentials.

Getting Connected

If you are using a terminal program, you can simply type:

```
$ ssh <auburn userid>@hopper.auburn.edu
```

Setting the Environment/Compiling Code

To set your environment to use MPI, load an MPI module. For this lab, we can just use the default version of OpenMPI.

```
$ module list
$ module avail openmpi
$ module load openmpi
$ module list
```

Downloading the Sample Code

```
$ git clone https://github.com/auburn-research-computing/mpi_fundamentals.git
$ cd mpi_fundamentals
```

II. Compiling

With the MPI module loaded to set our environment to use the MPI libraries, compilers, and runtime environment, we can begin compiling the sample code.

Remember, to compile code that uses MPI library functions, we need to use the MPI compiler wrappers. These usually begin with a prefix of "mpi" followed by a suffix that describes the language. To see some of the available compilers, you can look in the bin directory of the version of OpenMPI we have loaded.

```
$ ls /tools/openmpi-1.10.2/gcc/4.9.3/bin/
```

Let's try compiling our first sample code. In this case we have C++ code, so we will use the mpic++ compiler…

```
$ mpic++ hello_mpi.cpp -o hello_mpi
```

If you don't see any errors or warnings, you should now have an MPI capable program called "hello_mpi" in your directory.

Let's try this for the remaining .cpp files …

```
$ mpic++ mpi_send.cpp -o mpi_send
$ mpic++ mpi_bcast.cpp -o mpi_bcast
$ mpic++ mpi_scatter.cpp -o mpi_scatter
```

III. Testing

Testing Code

Remember, it's not a good practice to run any code extensively on the login node. However it is certainly acceptable to quickly test an executable just to see if it will run at all.

Once an application is running in the foreground, you can enter [CTRL-C] to stop it.

If you are unsure about what you might be running, you can use the "ps" or "top" commands to see which processes are associated with your account. These commands will return Process IDs (or PIDs). If necessary, you can use the "kill" command along with the PID to stop a process. See "man kill" "man ps" "man

Try some quick tests on the compiled MPI programs.

```
$ mpirun -np 4 hello_mpi
```

```
hello from rank 3
hello from rank 0
hello from rank 1
hello from rank 2
```

III. Next Steps

Take a look at the sample code and use your favorite editor to try changing some of the functions.

Try writing your own MPI code and look through the MPI documentation for features that might be useful for your workload.

Let's use qsub to run your program in interactive mode. This will run your program as a job on one of the compute nodes, but will allow us to issue commands to the node interactively.

```
$ qsub -q gen28 -l nodes=1:ppn=2 -I
$ module load openmpi/gcc
$ mpirun -np 2 ~/prime/prime
$ exit
```

Finally, submit your job as a batch job to the cluster.

```
$ qsub -q gen28 -l nodes=1:ppn=2 prime.sh -d ~/prime
```

What happened? Did the job run successfully?

Here's the prime.sh script:

```
$ mpirun -np 2 prime > Prime.out
```

Anything missing?

Here are some other important sub parameters by example (man qsub for more details):

```
$ qsub -l nodes=1,walltime=4:00:00 -m abe -M abc@auburn.edu prime.sh
```

Advanced Users

Another way to submit a batch job is by using PBS directives:

$ qsub prime.pbs

See **Next Steps** section at the end of this document for more detail.

How to monitor your job:

```
$ showq -u <userid>
$ qstat -u <userid>
$ qstat -f <job#>
$ checkjob –v –v –v <job#>
```

How to cancel a job:

```
$ mjobctl -c <job#>
$ qdel <job#>
$ canceljob <job#>
```

IV. Job Scheduling

Reservations

The Hopper HPC cluster uses “reservations” to define priority access to a subset of nodes for each lab group. Reservations use preemption to guarantee this access within 1-2 hours. Jobs that are running outside of their reserved capacity are subject to preemption. For more information, visit: https://aub.ie/hpcres

When will my job run? What resources are available?

Before submitting a job, users should consider what's running on the nodes in their reservation:

The showres command will show the reservations to which you have access.

```
$ showres
```

If available nodes in your reservation, then use your reservation. If no available nodes in your reservation, but you do NOT want your job preempted, then use your reservation. However, your job must wait.

Use this script to determine what's running on your reservation:

```
$ whats-running-on-my-reservation.sh
```

How to specify your reservation in your job submission:

1. Use ADVRES flag in qsub

```
$ qsub -l nodes=1:ppn=20 -W x=FLAGS:ADVRES:<reservation_id> job.sh
```

2. Use rsub instead of qsub as it will automatically include your reservation in the job sub:

```
$ rsub -l nodes=1:ppn=20 job.sh
```

Note: Use rsub only when running in the general queue and you are a member of a single group.

Running outside your reservation:

If available nodes and you do NOT care if job is preempted, then submit without reservation.
If no available nodes and you submit without reservation, your job must wait.

Use this script to determine what resources are available:

```
$ /tools/scripts/find-excess-capacity.sh
```

To see what is currently running on the entire cluster:

```
$ showq
```

To see what jobs you have currently running

```
$ showq -u <username>
```

Best Practices Summary

I. Running a new program for the first time:

- First, run *briefly* on login node just to make sure that your code will run. If the program runs without an immediate indication or an error or problem, use CTRL-C to exit.
- Then, run using qsub in interactive mode to make sure that it will run on a compute node.
- Finally, run in batch mode using qsub.

Do not run jobs on the login node except as a test

This means short jobs using small amounts of memory to ensure that your code will run. Processes that violate this will be killed.

II. Don't submit a job, assume it's running correctly and walk away or leave for weekend. Make sure the job is running and, if not, understand why not.

III. Specify walltimes in your job submission.

- Allows Scheduler to maximize utilization which means your jobs run sooner.
- Users should receive an email after a job completes that contains the actual walltime.

Submit short-running jobs with fewer resources in order to reduce likelihood of preemption when not using your group's reservation.

IV. Clean up when your jobs are finished.

- Hopper does not provide archival or long-term storage.
- If files no longer need to be available for work on the system, copy them off and delete them so that the space can be used for active projects.

V. Pay attention to your disk usage. Once the hard limit is reached in disk space or # of files, your program will stop executing.

VI. Do not share passwords or accounts. If you want others to access your files, then set them to read only.

How to Get Help

Because Hopper is regarded as a research (rather than production) system, HPC support is normally available only during regular business hours. When reporting problems, please provide as much relevant information as possible. This should include the following, as appropriate:

- date and time when the problem occurred
- job number
- text of the command(s) which you issued
- exact and complete text of any error messages
- any other information helpful in identifying or resolving the problem

If further assistance is needed, please email to schedule an appointment with one of the HPC admins at hpcadmin@auburn.edu.

Next Steps

You can add qsub options to your shell script. This saves time and lots of typing when you have to specify lots of options

For example, you can run prime using a PBS script: prime.pbs…

```
#!/bin/bash

email=`whoami`@auburn.edu
workdir=/home/`whoami`/prime
cd $workdir
#PBS -N Prime
#PBS -m abe
#PBS -M $email
#PBS -l nodes=1:ppn=2,pmem=1gb
#PBS -q core
#PBS -d $workdir

module load openmpi/gcc
mpirun prime
```

```
^G Get Help ^O WriteOut ^R Read File^Y Prev Page ^K CutText ^C Cur Pos
^X Exit     ^J Justify  ^W Where Is ^V Next Page ^U UnCutTx ^T ToSpell
```

Notice the #PBS directives in this script are equivalent to some of the qsub options that you have specified on the command line.

Find software for your particular research domain and begin experimenting. We have many popular software packages available for use. You can see them with …

```
$ module avail
```

If a software package that you would like to use is not already available, visit https://aub.ie/hpcsw to request a new package.