Development on MIST*

University of Oregon, Department of Computer and Information Science
Parallel Programming Course

Rev: 1.0, February 2014

*This document is meant to be used as a reference only. Information presented here was accurate and true at the time of its creation, but all software changes and evolves, so use your best judgment in the creation of your development environment.

Table of Contents

1 Introduction

This document has been designed to go through all of the steps necessary to access, gain a basic understanding of the scheduling system, as well as how to load and run the provided example code on MIST.

This document is broken into two sections to break up the work. First, we will go over some basic information about how MIST is setup and configured in Section 2 followed by how to access and load the necessary modules on MIST to run the example code in Section 3.

2 MIST Specifications

This section will detail how to get an account on MIST, as well as some of the statistics about the MIST compute cluster.

2.1 Requesting an Account

Accounts for MIST must be requested online through the systems account portal, and this can be achieved with the following steps:

Step 1: Navigate to the systems account request portal at https://systems.nic.uoregon.edu/account/. You will be taken to a page similar to that of Figure 1.

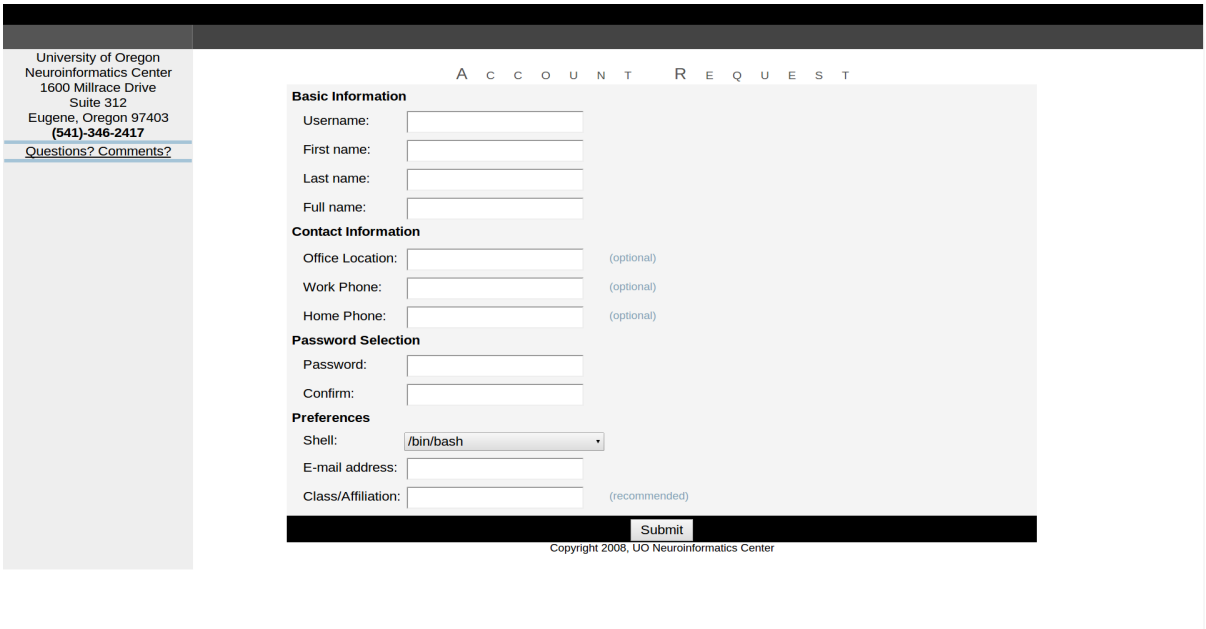

Figure 1: Systems Account Request Page

Step 2: Once on the Systems Account request page, fill in your basic information with your desired username, first and last name, full name, desired password, email, and your **Class/Affiliation**. This last item is important so that you are given access to the correct machine in a timely manner. Your basic information should look something like that of Figure 2.

Step 3: Submit your information and wait for the request to be filled.

2.2 MIST Basics

- 1U Dell PowerEdge 1950 (II and III)
- 130 Xeon processors:
  - Frontend: Dual-core 5130 (Woodcrest) @2.0Ghz
  - Nodes 0-3: Dual Quad-core E5345 (Clovertown) @2.33Ghz; 4MB L2/die (8MB total)
  - Nodes 4-15: Dual Quad-core E5410 (Harpertown) @2.33Ghz; 6MB L2/die (12MB total)

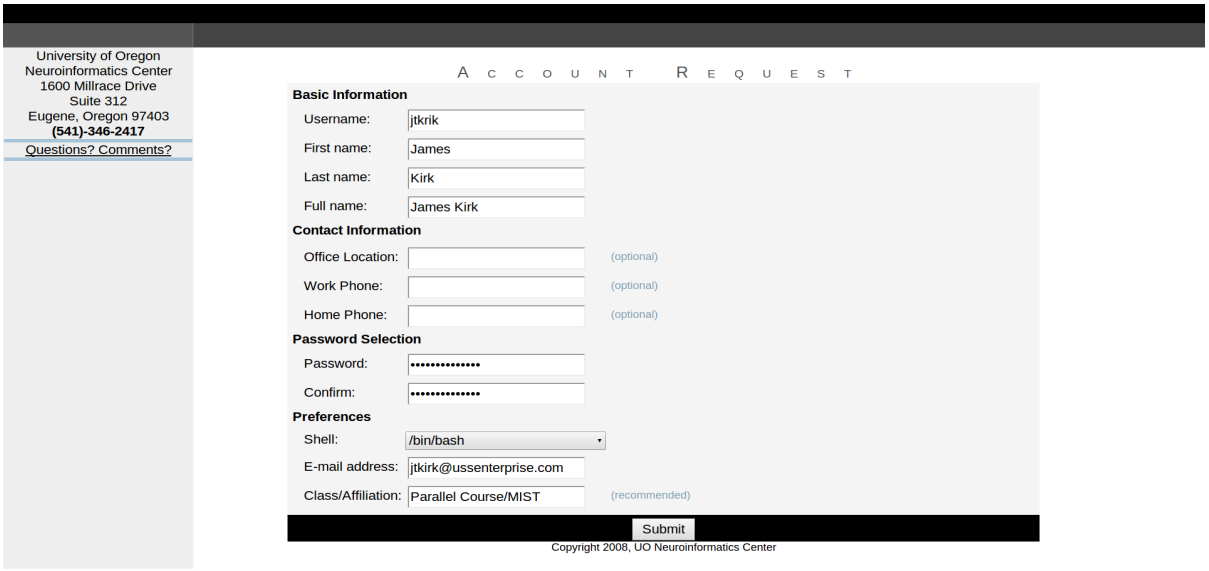

Figure 2: Systems Account Request Page with Filled Information

- 258Gb RAM:
  - Frontend: 2Gb
  - Nodes: 16Gb
- Common to all 17 nodes:
  - 1 73Gb SCSI or SATA Disk
  - 2 onboard Broadcom NetXtreme II copper NICs
  - 2 PCI-X Intel Pro/1000 copper NICs
  - Dell Embedded Remote Access Controllers for remote management
  - Rocks Cluster Linux 5.0 (CentOS 5.9 + perfctr kernel)
- All nodes in a Chatsworth Megaframe rack

2.3 Scratch Space

Approximately 750GB of PVFS2 high performance scratch space is available at /scratch. This space is not for permanent data storage. Files in this directory may be removed if not used for more than 24 hours or when the system is restarted.

2.4 MIST Environment

Our Mist and Neuronic clusters use the University of Tromso's Torque roll, which provides a preconfigured Torque+Maui environment. This environment provides job submission and queue management functions similar to those available at other HPC sites.

2.4.1 Node Access

The only way to gain access to backend nodes is by submitting a job. This is not just limited to interactive logins; you may not boot an MPI ring on backend nodes outside of a job either.

While you may be able to use standard MPI environments to boot a ring on the frontend, and might be able to use SSH between backends as part of a job script, we discourage such activities and will not support them. Please use our provided MPI environments.

2.4.2 Interactive Node Logins

Please be aware that we only have a limited number of nodes! Using an exclusive interactive session for long periods of time significantly reduces the compute resources available to others. Be a good citizen and only use interactive logins when necessary, and for as little time as possible.

Using MIST during high traffic times To use an interactive session with MIST during high traffic times, it is a good idea to request only a fraction of a node. To request a fraction of a node use the following request format (using fewer than 8 processors which is the maximum per node):

```
qsub -I -x -l nodes=1:ppn=2
```

Using MIST for performance testing We provide a qlogin script that will set you up with exclusive node access. The script also enables X11 forwarding when possible. Simply run:

```
qlogin <optional hostname>
```

Or, to specify a set of nodes use:

```
qsub -I -x -l nodes=4:ppn=8
```

2.4.3 Walltime Limits

- Jobs on Mist are limited to 4 hours. If your job cannot run within this time, please contact systems@nic.uoregon.edu to request access to the mist_nolimit queue.
- The scheduler will, under certain circumstances, preferentially run short jobs before long jobs. To inform the scheduler that your job will only take a short amount of time, you may specify the intended duration:

```
# Example of a 16-node job with a 15 minute duration
#PBS -l nodes=16:ppn=8,walltime=00:15:00
```

3 Environment Interaction and Running Example Code

In this section we will go over how to make and run the example code for this course. This section is important as it goes over specific modules that must be loaded each time you start the system in order to have access to the appropriate libraries and compilers.

3.1 Accessing MIST

After your MIST account has been created you can access MIST using SSH as follows:

```
ssh <username>@mist.cs.uoregon.edu
```

3.2 Obtaining and Running the Example Code

After you have a connection to MIST, there are a few options for compiling and running the example code. This example code is from the book **Structured Parallel Programming**, and is useful to ensure that your system is ready for development. These examples will ensure that the appropriate compilers and libraries are available, and will run several example tests.

3.2.1 Get the Example Code

The example code for the **Structured Parallel Programming** book is available at parallelbook.com/student. To download the most recent version of the code, copy the link and download using the command line as follows:

```
wget http://parallelbook.com/sites/parallelbook.com/files/code20131121.zip
```

Once the download is completed extract the code.

3.2.2 Compiling the Examples

The following steps will go over compiling and running the examples and tests with the various available options.

Step 1: Request Resources Before modules are loaded or you run code, we need to request resources that are not on the head node. Use your desired variation of the following command to request them:

```
qsub -I -x -l nodes=1:ppn=2
```

Step 2: Loading the Appropriate Modules Running on MIST requires that we load an Intel software module to obtain the needed version the compiler. To list all of the available modules on the system you can type ***module avail***. We will need to load one module. The current version of the Intel compiler is 14.0.1. Use the following commands to look at and load the module:

```
module avail
module load intel/14.0.1
```

To see which modules are currently loaded you can use ***module list***.

Step 3 Go to the top level of the examples directory, you should see three directories and three files (See Figure 3).

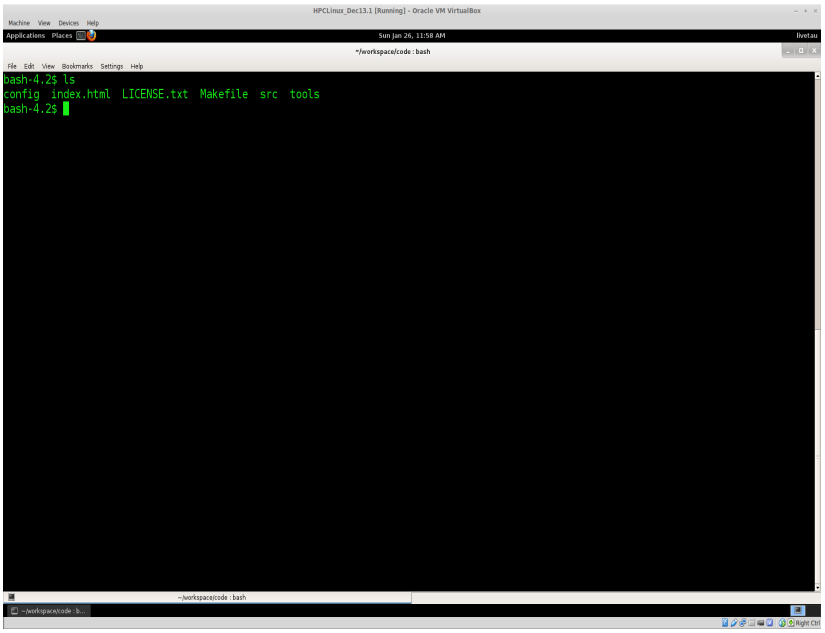

Figure 3: Top level directory of the example code

Step 4 The first thing that needs to be done, is the 'config.inc' file must be modified. Go to the **config** directory and open **config.inc**. To fully compile the example code and to take advantage of all the features, uncomment the second line in the file and set the compiler to 'icpc' (See Figure 4).. This will set the compiler to be the Intel compiler with support for Cilk Plus. Also in this file we need to modify the linker commands to account for how MIST is configured. On line 18, add (-L/usr/local/packages/intel_xe/14.0.1/tbb/lib/intel64/gcc4.1) before the -ltbb option.

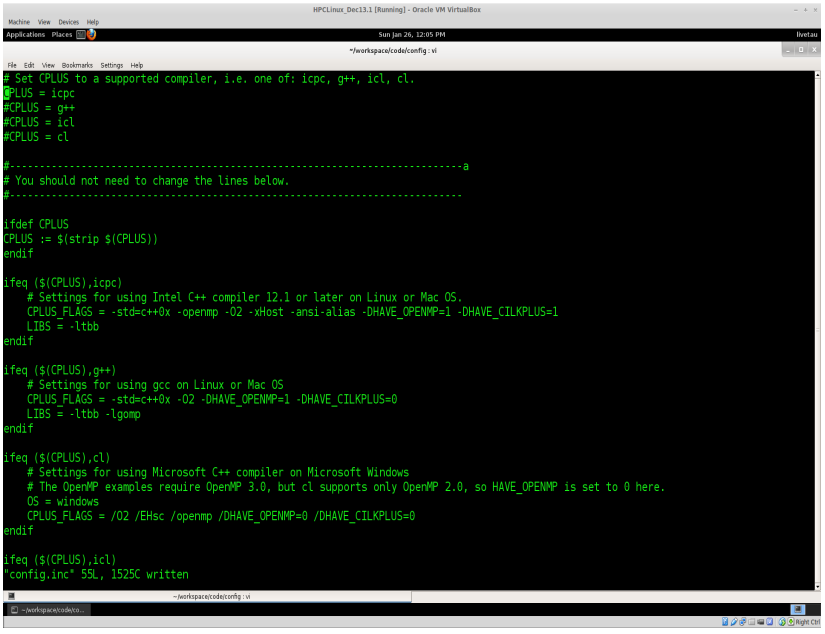

Figure 4: Uncommenting the compiler to allow for CILK Plus, TBB, and OpenMP code

Step 5 Now, go back to the top level directory and type **make**. This will compile the example code with the compiler option that was set in *Step 2* and run some of the test programs (See Figure 5).

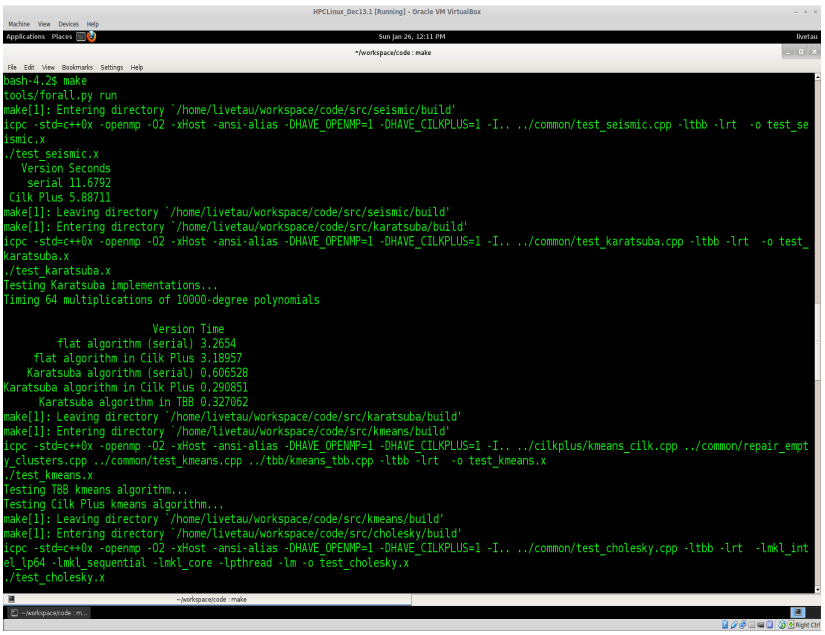

Figure 5: Uncommenting the compiler to allow for CILK Plus, TBB, and OpenMP code

Step 6 If there were no errors during the build and the tests completed, you should have an environment that is ready for development. If you want to look at or run any of the test programs individually they are available under the **src** directory (See Figure 6).

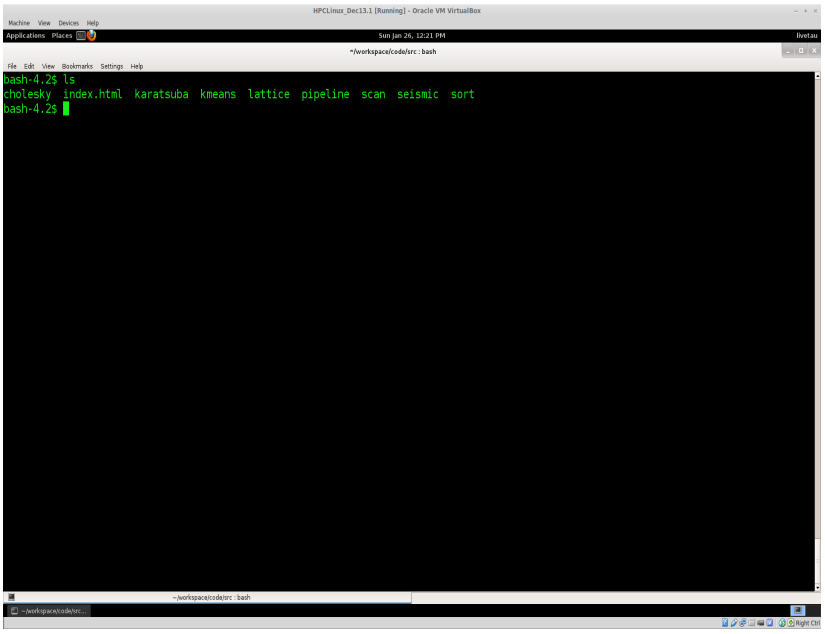

Figure 6: Test code directories available under **src**