Using MPI and the Penguin Cluster at USF

Department of Computer Science
University of San Francisco
Updated September 28, 2015

1 Introduction

You have three options for developing MPI programs: use MPI on one of the lab machines, install OpenMPI or MPICH on your own computer, or use the Penguin cluster.

Regardless of which option you choose, the development process will be the same:

- Write/edit your C-MPI program.
- Compile your program with mpicc:

  $ mpicc -g -Wall -o mpi_prog mpi_prog.c

(Here, the dollar sign ($) is the shell prompt: don’t type it in.)
- Run your program with mpiexec on a lab machine or your computer:

  $ mpiexec -n <p> mpi_prog mpi_prog.c

On the penguin cluster you should run your program with csmpiexec:

  $ csmpiexec -n <p> mpi_prog mpi_prog.c
Here, you should replace \(<p>\) with the number of MPI processes you want to start.

- Repeat these steps until your program works as desired.

Note that there are some differences among the various MPI implementations, and in order to insure consistency, we will test your homework and programs on the Penguin cluster. So you should test your programs on it before your final submission, and you do need to know how to use it.

2 SSH and Logging On

It’s very important that you’re able to log on to the nodes of the Penguin cluster without typing a password. So if you haven’t created an ssh key before, log on to a CS department computer (e.g., stargate.cs.usfca.edu) and type

- \( $\) ssh-keygen \(-t\) rsa

You should follow the prompts, accepting the defaults that the program gives. After you’ve generated your ssh-key, you need to copy the “public” key to a special file:

\[
\begin{align*}
&$\text{cd} ~/\text{.ssh} \\
&$\text{cp} \text{id\_rsa.pub} \text{authorized\_keys2}
\end{align*}
\]

While you’re still in the \text{.ssh} directory, create a file called \text{config} and add the lines

\[
\text{Host } *
\begin{align*}
&\text{ForwardAgent yes}
\end{align*}
\]

The permissions on this need to be 600. So leave the text editor and type

\[
$\text{chmod} \ 600 \text{ config}
\]

Be sure to remember the password you gave to \text{ssh-keygen}. The good news here is that you shouldn’t need to do this step ever again.
Now every time you log on to a CS department computer from a machine outside of the CS department, type

```
$ ssh-add
```

This will prompt you for your ssh password. After doing this, you can log on to any other CS department computer without typing a password.

If typing `ssh-add` gives you the message

```
Could not open a connection to your authentication agent
```

try typing

```
$ exec ssh-agent bash
$ ssh-add
```

This should prompt you for the password.

**Note:** If you’re using tcsh instead of bash, replace `bash` with `tcsh` in the `exec` command.

### 3 The Penguin Cluster

USF’s penguin cluster has 24 compute nodes and an Infiniband interconnect. Each node has two dual-core AMD processors. So in some sense the cluster is a $24 \times 4 = 96$-processor system. The first time you use the cluster, things are a bit complicated. So here’s a step-by-step outline of how to use it. After you’ve used it once, it should be pretty easy to use in the future.

- First, be sure to set up `ssh` as described in the preceding section.
- You should log on to `steelhead` with the command:

```
$ ssh steelhead
```

The first time you log on to a particular CS machine using ssh, ssh may ask you if you’re sure: just say yes. Steelhead is the main host for accessing nodes of the cluster.
• Log on to the cluster by typing

   $ ncs login -n <number of nodes>

Unless you’ve spoken to me, <number of nodes> should be small: four at the most, but two is probably plenty for most program development. The ncs program “checks out” the number of nodes you request (if they’re available). It also creates some special files and starts up some daemons (special processes) that the system uses to help with parallel program startup and communication in parallel programs. Finally it logs you on to one of the nodes.

3.1 Compiling and Running

This is the same regardless of which system you’re using.

• To compile an MPI program, use mpicc. For example, to compile the hello program type

   $ mpicc -g -Wall -o mpi_hello mpi_hello.c

The mpicc command is just a “wrapper” for gcc. It runs gcc and makes sure that gcc knows where the MPI header files are and how to link in the MPI libraries.

Note: Your CS home directory is “mounted” on all of the nodes of the cluster. So you shouldn’t need to make a special copy onto the cluster, and the location of any file on the cluster should be the same as it is on any other CS department machine.

• To run the program, type

   $ csmpiexec -n <p> ./mpi_hello

on the penguin cluster. On other systems type

   $ mpiexec -n <p> ./mpi_hello

You should replace <p> with the number of MPI processes you want to use.
3.2 Finishing Up

- When you’re done using the cluster, you can just log off by typing

  $ exit

  This will log you off of the cluster and return your nodes to the pool of available nodes.

- If you want to check to be sure that your nodes are available for other users (a very good idea), type the following command on steelhead

  $ ncs status

  This will provide you with a list of the statuses of all of the cluster nodes.

- If this shows you still have nodes checked out, note their numbers and use ncs to check them back in. For example, if nodes penguin01, penguin02 and penguin17 are checked out to you, type

  $ ncs checkin 1 2 17