Beowulf Cluster Lab
Laboratory Experience

The Beowulf Cluster has 61 nodes: the master node is `beowulf` (also known as `node00` inside the cluster), the compute nodes are `node01`, `node02`, . . ., `node60`. The master node is mainly used for monitoring the cluster and for submitting jobs while the compute nodes do the real work.

In this exercise, we will work through some of the basics of using a cluster and running parallel programs. A more detailed users guide for the Beowulf Cluster Lab is at:

http://cs.boisestate.edu/~amit/research/beowulf/lab-notes/lab-notes.html

There are currently three popular libraries for developing parallel programs: PVM, MPICH-MPI and LAM-MPI. All are supported in the Beowulf Cluster Lab. We also use the PBS batch system to control allocation of nodes to users.

1 Laboratory Session

1.1 Part 1

1. Login in to any workstation in the ET213-214 lab using the account and password provided to you.
2. Login to the master node, `beowulf`, using the command `ssh` in the console window as shown below:

   ```
   ssh yourlogin@beowulf.boisestate.edu
   ```
3. Change your password using the command: `passwd`
4. **Browsing files**. Type in the command `filebrowser` to bring up a graphical file browser for your beowulf account. The browser is useful for looking at your home directory and doing simple editing.
5. **Setting up Email Forwarding**. In the file browser, select the *View* menu, then select *Show Hidden Files* option. Now find the file named `.forward` and double click on it to edit it. When the file opens in the editor, type in your preferred email address on the first line. Then save the file. If you want to forward your email to more than one account, then type in extra email addresses separated by commas. Next, select *View* menu and then unselect *Show Hidden Files*.
6. **Cluster monitoring**. Click your mouse back in the beowulf console. Here we will try a few cluster monitoring commands. Try the following commands.
   - `cchk`: Check the status of the cluster nodes.

- `cdate`: Check the time on each node of the cluster.
- `cfree`: Check the amount of memory available on the cluster.
- `cmips`: Check the amount of computing power of the cluster.
- `cdisks`: Check the amount of disk space available on the cluster.
- `ctemp`: Check the temperature of the cluster CPUs.
- `xpbsmon &`: Starts up a GUI program that lets you watch the allocation status of the cluster. Green means a node is free.

7. **Cluster control with parallel shell**. The `pdsh` shell is a simple parallel shell that is useful for doing tasks across the cluster. The utility `dshbak` helps with sorting the output from `pdsh`. Try the following commands: `pdsh -a date`

```
pdsh -a uptime
pdsh -a ps augx | grep ‘whoami‘
pdsh -w node[16-26] who
pdsh -w node[16-26] who | dshbak
```

8. **Finding out how many nodes are available**. The command `freenodes` tells you the number of free nodes that are available for running your programs. Remember that each node has two CPUs. Try the commands `freenodes` and `freenodes -v`.

9. **Acquiring nodes for running programs**. Acquire 8 nodes using the command:

```
pbsget -6
```

10. Check the nodes allocated to you with the command `qstat -n`. In general, you can use the command `qstat -a` to get a quick summary of everything running on the system

11. **Running parallel programs interactively**. Here we will show how to run parallel programs that use PVM, LAM-MPI and HPF. Let us first run a PVM program. The source for the program is under the directory `examples/parallel_sum/PVM/C`. We will use the command `pvmrun` to run our PVM program as shown below.

```
pvmrun -np 12 spmd_sum 20000000
```

To run a LAM-MPI program, use the following commands:

```
lamsetup
lamboot
cd ~/examples/parallel_sum/MPI
mpirun -np 12 spmd_sum_mpi 20000000
lamhalt
```

To run the HPF version of the sum program, use the command:

```
mpichsetup
cd ~/examples/parallel_sum/HPF
mpirun -np 12 sum -pghpf -np 12
```

Note that we are specifying 12 tasks to be run even though we acquired 6 nodes on the cluster. This is because each node on the cluster has two CPUs and we want to make use of all the CPUs that we were allocated.

12. **Running parallel programs in batch mode**. Normally, you will be running your big jobs in batch mode. This is convenient for several reasons. You can just submit the job and logout of beowulf. You will get informed by email when you job completes. Also there is a time limit of 1 hour for interactive PBS jobs. For batch jobs the default time limit is 24 hours (which can be extended by using PBS options in your batch file).

    Look at the `psum.pbs` batch script in the directory `~/examples/parallel_sum/PVM/C`. (Just click on the file in the file browser or use the `edit` command from the console) We will submit `psum.pbs` to run in batch mode. You don't need to acquire nodes for this purpose since the script acquires the nodes when it runs. Use the command shown below. It names the job as`test1`.

    ```
    qsub -N test1 psum.pbs
    ```

13. **Checking on the status of the job abd the output**. Use the command `qstat -a` and you should see your job listed. Once the job completes, the output should be in the files `test1.o<jobid>` and `test1.e<jobid>`. Check those files.The PBS system also will send you an email at the beginning and end of the job if the PBS script has the appropriate settings (see the third line in our example)Check your email. If you forward your email from `beowulf`, then you will see the emails at your regular email address.

14. **Checking your usage of the cluster**. The command `pbs_usage` shows you how much you have used the cluster in the current month. To get the total usage for all months, use the command `pbs_usage -t`.

15. **Compiling a parallel PVM program**. Change directory to the `examples/parallel_sum/PVM` directory. Build the `spmd_sum` program using the `aimk` command:

    ```
    cd ~/examples/parallel_sum/PVM/C
    aimk spmd_sum
    ```

    Similarly, we have provided C++/F77 versions of the PVM sum program.

16. **Compiling a parallel HPF program**. Change directory to the `examples/parallel_sum/HPF` directory. Build the `sum` program using the `make` command:

    ```
    cd ~/examples/parallel_sum/HPF
    make sum
    ```

17. **Compiling a parallel LAM-MPI program**. Change directory to the `examples/parallel_sum/MPI` directory. Build the `sum` program using the `make` command:

    ```
    cd ~/examples/parallel_sum/MPI
    make spmd_sum_mpi
    ```

2 Part 2

In Part 2, we will get some experience with `xpvm` visualization environment for a parallel PVM program.

1. **Acquire nodes from PBS**. Acquire 4 nodes from the cluster.

    ```
    pbsget -4
    ```

2. Start up XPVM by typing in `xpvm`. It should show five machines in your virtual parallel machine. We are going to run a parallel program without looking at the source code first. Under the *Views...* menu, click on *Utilization* to open a cluster utilization window. Also under *Views...* menu, click on the *Message Queue* choice to open another window that shows queue length at each node and finally under the *Views* menu also select the *Task Output* choice to get a window that shows the output from the program.

3. Go to *Tasks...* menu, then click on *Spawn* option to get the spawn window. In the *Command:* field you can type in the name of your program, `lb_demo`. It takes no command line arguments. Next select the *PvmHostCompl* button. Then select the *Host* button. A new field labeled *Host* will show up. Type the internal name of the master node:`node00`. The purpose of the last two steps is to ensure that the PVM processes are not scheduled on the master node. Next type in the number of SPMD tasks you want to spawn in the *NTasks* field (4 in our case). Finally click on the *Start* button to start the parallel program.

4. While the demo is running, we can monitor various properties of the parallel program.

5. Note that you can zoom in the visualization by pressing the middle mouse button and dragging across the area in the visualization that you want to zoom in. Similarly, clicking on the right mouse button zooms out.

6. What inferences can you draw about the parallel program `lb_demo`?

7. Once you are done then halt the PVM system. To halt PVM, go to the *File* menu and then select *Halt PVM* choice. Then exit out of the PBS session.

3 Working from your computer

3.1 Setup

If you are using Linux on your computer, then you will follow exactly the same steps as above to connect to `beowulf` and use it.

If you are using MS Windows, then you will need to install the Cygwin package. The you can use `ssh` to connect to `beowulf`. All the graphical tools that we have used will work as usual.

If you are using MAC OS X, then you will need to install the X11 server/client athat comes with MAC OS X distribution. Once you start the X11 server/client, then you can use `ssh` to connect to `beowulf`. All the graphical tools that we have used will work as usual.

3.2 Copying files from/to the cluster

3.2.1 Command Line tools

From a console, use the `scp` command. For example, to copy a program `prog1.c` from your current directory to the directory `programs` on your account on `beowulf`, use:

```
scp prog1.c beowulf:programs/
```

To copy an entire directory recursively, use the `-r` option. For example:

```
scp -r project beowulf:
```

3.2.2 Graphical File Transfer Programs

For Linux, use the `konqueror` file browser. In the address field, simply type in the address:`sftp://yourlogin@beowulf.boisestate.edu`, where `yourlogin` is your login name on the cluster. After that you can use the *Window* menu and select `Split View Top/Bottom`. Now click your mouse on the bottom window and then click on your *Home* icon. This will give you half a window on your local machine and the other on `beowulf`. Now you can simply drag and drop your files in either direction.

For MS Windows, you can download a graphical SSH client (which includes a graphical File Transfer client) from `http://www.ssh.com`.