BCCD

Goal:

This activity gives students hands-on experience with running parallel programs on a parallel computer.

Materials:

- Computer, one per student.
- One computer connected to a projector for the instructor.
- Ethernet network connecting all computers together, disconnected from any other networks.
- VirtualBox (https://www.virtualbox.org) virtual disk image (VDI) running the Bootable Cluster CD (BCCD) operating system (<u>http://bccd.net</u>) on the instructor's computer.
- VirtualBox virtual machine that boots from the network by default, one per student computer.
- Parallel Computing Notebook, one per student.

Activity:

- 1. The instructor instructs the students to disconnect the Ethernet routers from any outside networks, and to make sure that their computers are each connected to the router, and that all routers are connected together.
- 2. Each student logs in to a local account (the instructor provides the credentials).
- 3. Each student opens the **Activity Monitor (Mac)** application so they can monitor how much of the system resources are being used.
- 4. Each student opens the **VirtualBox** application. The **bccd** virtual machine should be present.
- 5. Each student sets the number of cores for the virtual machine by clicking the **bccd** virtual machine, choosing **Settings**, clicking the **System** tab, clicking the **Processor** tab, and increasing the number of processors to the total number of cores on the computer (you can figure this out on Mac by going to the Apple menu, choosing **About This Mac**, then **System Report**).
- 6. The instructor demonstrates how to boot the BCCD virtual machine. NOTE: it is important that only one computer boot fully first, otherwise some computers will not be able to connect to all the others. The instructor clicks the Start button and explains that this is like booting up a computer that has the BCCD operating system installed on it. BCCD is a Linux operating system developed and maintained with support from Shodor, designed for making it easier to teach parallel computing.
- Once the boot process finishes, the instructor points out that the hostname of the computer is node000. Each node in the BCCD cluster has a unique hostname that starts with node0. The head node of the BCCD cluster is node000.
- 8. Each student clicks the **Start** button to boot their virtual machine. The student should not see the **No DHCP** blue screen, and the hostname of each student's machine should end

up being different. If any are the same, the student should let the instructor know, and together the student and instructor should diagnose any network connection issues.

- 9. The instructor and students review some of the basic Linux commands:
 - a. **pwd**<ENTER> prints the name of the working (current) directory (folder). For now this should print /home/bccd, which is the BCCD user's home directory.
 - b. **Ls**<**ENTER>** lists all the files in the working directory. For now this will list all the files and folders in the BCCD user's home directory.
 - c. cd name-of-directory<ENTER> change the working directory. For example:
 - cd GalaxSee<ENTER> change to the directory named GalaxSee. This directory has the code for a model of stars in a galaxy interacting gravitationally. Enter pwd<ENTER> to confirm it worked; it should print /home/bccd/GalaxSee.
 - d. mv source destination<ENTER> move or rename a file. For example,
 - i. mv GalaxSee.cxx-mpi gal-student-name <ENTER> rename the binary executable file for the galaxy model. Students should replace student-name with their name so they can keep track of their executable files running across all the computers in the room.
- 10. The instructor introduces some of the BCCD-specific commands:
 - a. bccd-snarfhosts -sv<ENTER> find all the other BCCD nodes on the network and lists them out, one per line, sorted. It also indicates how many cores each node has, referred to as slots. Furthermore, the command creates a file in the BCCD user's home directory called machines-openmpi, which can be used when running a program across multiple nodes (it is analogous to a roster of a sports team, which lists the people who will be playing together). NOTE: this command may produce errors related to egrep. These errors can safely be ignored.
 - b. bccd-syncdir . ~/machines-openmpi<ENTER> synchronize (copy) a directory across all the machines listed in the file machines-openmpi. This will create a directory on every node in the cluster; the directory will be named /tmp/node0##-bccd, where ## is replaced by the unique ID of the node that created it.
- 11. The instructor introduces the Linux command for keeping track of which processes are using the most % of the CPU: **top**<<u>ENTER></u>
- 12. In another terminal window, the instructor introduces the command for running a program across multiple computers in parallel:
 - a. time_mpirun_-np_A_-machinefile_~/machines-openmpi_\ /tmp/node0<mark>B</mark>-bccd/GalaxSee.cxx-mpi_C_D_E<ENTER>
 - i. Students should replace the highlighted letters in that command with the following:
 - 1. **A** with the number of **processes** (independent pieces of the program that run in parallel) they want to use.
 - 2. **B** with the unique ID for their node.

- 3. **C** with the number of stars in the galaxy.
- 4. **D** with the mass of each star, measured in solar masses (the mass of our sun).
- 5. **E** with the number of millions of years of model time to run the model.
- 13. Students check the **top** command to see which processes are running on their system. They also check Activity Monitor (Mac) to see how much of the system resources are currently being used.
- 14. When the command finishes, it will list three different times:
 - a. **real** time is the amount of wall clock time from when the program is entered to when it finishes running.
 - b. **user** time is the amount of time the program actually runs code. This does not include the time the program spends waiting for resources.
 - c. **sys** time is the amount of time the Kernel of the operating system runs on behalf of the program.
- 15. The instructor introduces the concept of **scaling**, in which the number of processes is increased across multiple runs of the same program. The **real** time for each run of the program can be measured, and this can be graphed vs. the number of process.