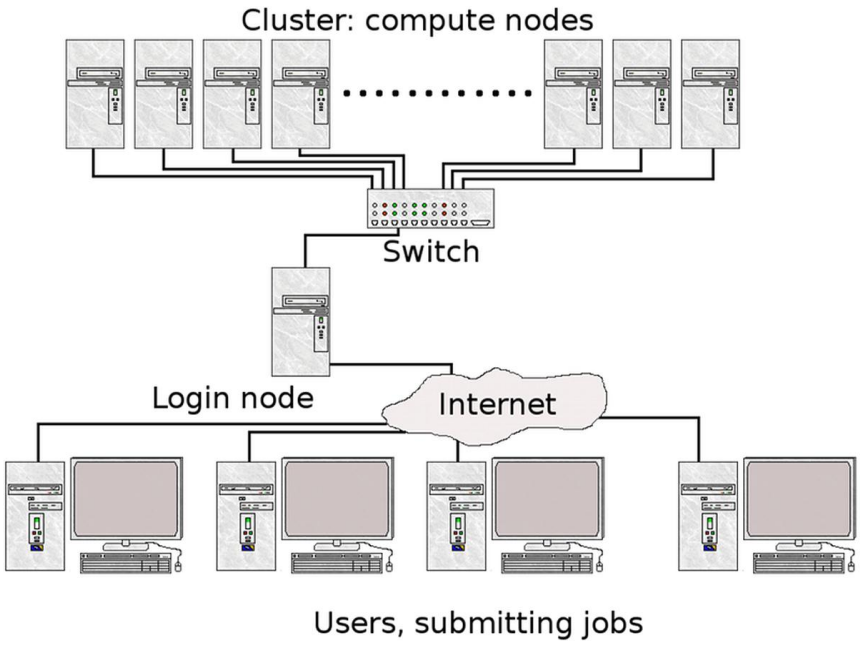
SE498 Parallel Computing
Lab 1: Getting used to the environment
Due: end of class

1. Objectives

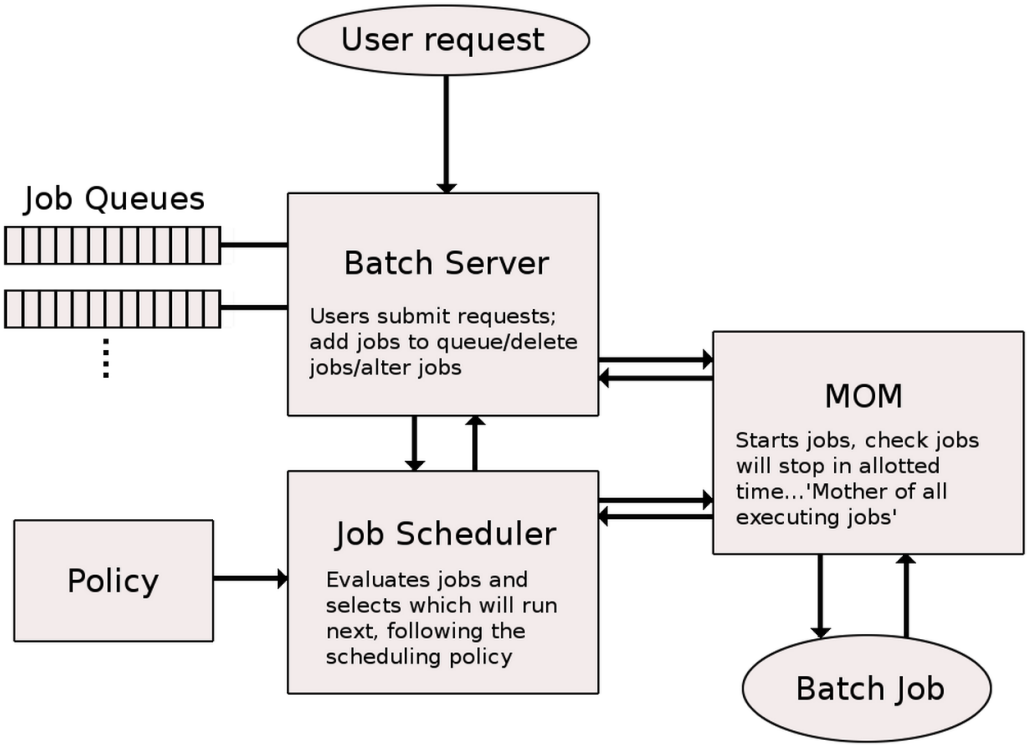
- Explain the difference between interactive and batch processing systems
- Understand how programs execute in a supercomputing environment.
- Write and submit a simple job to a supercomputer for processing.

2. Introduction

Welcome to the world of supercomputers. In this lab, you will execute your first program in a supercomputer environment. In doing so, you will learn a bit about how a supercomputer works.



Unlike your laptop or desktop PC, a supercomputer is composed of multiple computers making up a cluster. You log onto the supercomputer using one of the login nodes. The login nodes allow you to interact with the system, submit jobs, and manipulate files. Typically, for supercomputers, the login node is running some form of a UNIX operating system. At the Ohio super computer center, you can access the login node by connecting with putty (or another ssh client) to glenn.osc.edu or oakley.osc.edu.



When you execute a program on a supercomputer, it is typically not executed in an interactive fashion the same way it is done. Instead, you submit a job to the system. The job is a script specifying one or more programs that are to run to completion. You then submit this job to the scheduler and the scheduler determines exactly when the job is to execute.

The machines at the Ohio Supercomputer center use the Portable Batch System (or simply PBS) scheduling system. This system allows one to write a simple script and schedule commands to execute in the future. PBS scripts generally have three portions, commands to the scheduler to tell it how to schedule the job, comments, and the commands that are to be invoked when the job executes. Comments are lines that start with a # symbol. Commands to the scheduler start with a #PBS. Commands that are part of the script are simple unix commands without a # or #PBS prefix.

The following shows a simple PBS script. It tells the user which computer node is being used, the working directory, and does some associated output.

```
#PBS -l walltime=00:02:00
#PBS -l nodes=1:ppn=1
#PBS -N helloWorldJob
#PBS -j oe
#PBS -r n

echo ----
echo Job started at `date`
echo ----
echo This job is working on compute node `cat $PBS_NODEFILE`

cd $PBS_O_WORKDIR
echo show what PBS_O_WORKDIR is
echo PBS_O_WORKDIR IS `pwd`
echo ----
echo The contents of PBS_O_WORKDIR:
ls -ltr
echo
echo ----
echo
echo creating a file in PBS_O_WORKDIR
whoami > whoami-pbs-o-workdir

cd $TMPDIR
echo ----
echo TMPDIR IS `pwd`
echo ----
echo wait for 10 seconds
sleep 10
echo ----
echo creating a file in TMPDIR
whoami > whoami-tmpdir

# copy the file back to the output subdirectory
pbsdcp -g $TMPDIR/whoami-tmpdir $PBS_O_WORKDIR/output

echo ----
echo Job ended at `date`
```

On the Ohio Supercomputer center machines, this can easily be entered by issuing the command

`qsub demo.job`

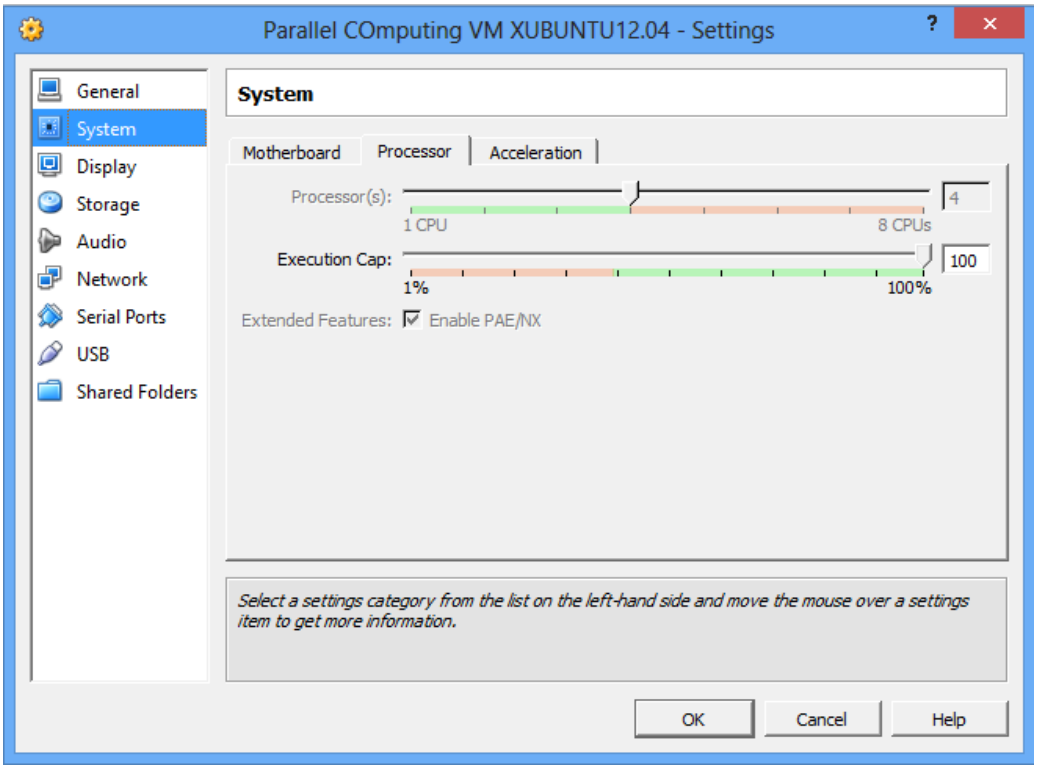
assuming that the script is named demo.job.[1]

[1] The Ohio super computer center has a tutorial on this topic at https://www.osc.edu/content/batch-tutorial

3. Specific Activities

3.1. Starting development on the virtual machine

Initially, you will want to start by downloading and installing the VirtualBox virtual machine from the isntructors website. This image contains tools that will allow you to develop all of these labs locally on your machine and test them out before uploading them to the parallel processing machines. As you setup the machine, you will want to make certain that the number of cores is set to 4.



From the instructor's website, download the files lab1.tar.gz, which contains a simple program. The program will use Newton's method to solve for the zeros of the equation $y = x^2 - n$ where n is varied between 1 and 100000. Effectively, this generates a table of the square roots of the numbers between 1 and 100000. These results are then dumped out to the console. In addition, the code times how long it takes to perform the operations.

The program takes one command line argument, the number of cores that should be used for the calculations. For example, issuing the command *newtondemo.exe 1* will generate this data using a single core. Issuing the command *newtondemo.exe 2* will generate the data using two cores. First, build the code on your local machine and execute it using 1 core. The code is built by simply issuing the command make at the console prompt. Record how long it takes to execute. The run the program with two cores and compare the execution time against the initial code.

From this, you can make a couple of useful calculations. One is called speedup. Speedup is defined as $S = \frac{T_{serial}}{T_{parallel}}$ and it explains how much improvement there is by adding additional processors. The second calculation you can make is called efficiency, which is defined as $S = \frac{\frac{T_{serial}}{T_{parallel}}}{p} = \frac{T_{serial}}{p \cdot T_{parallel}}$. Calculate these values for your laptop PC. When this is done, run the program with 4 cores on your laptop PC as well and repeat the calculations.

3.2. Watch the tutorial video

Before proceeding with the next section, watch the short online tutorial on submitting jobs to the Glenn cluster.

3.3. Running your program on the OSC Glenn cluster

Now that you have successful executed your program on the local machine, we want to run it on the supercomputer and do the same thing. In order to do this, we need to first transfer the files to the machine, write a job script, and submit the job for processing.

Below includes a sample job script for process. The details of the syntax will be explained to you in class. It will compile the given program, execute the program multiple times, and report back the results in text files. These text files can then be analyzed to determine the execution time for the program.

To begin, sftp your c, h, and makefiles for the project to sftp.osc.edu. This is the secure ftp server for the file cluster at the Ohio Supercomputing Center. Next, connect via ssh to the machine glenn.osc.edu. This is the login node for the Glenn supercomputer. Using nano or emacs, enter the following job script and save as newton.job. The script will build the code, execute the program 8 times using between 1 and 8 cores, and dump the results to 8 results files.

```
#PBS -N newtonsMethodDemo
#PBS -l walltime=00:01:00
#PBS -l nodes=1:ppn=8
#PBS -j oe
# Change to the directory from which the job was submitted.
cd $PBS_O_WORKDIR

# Make the code cleanly
make clean
make all

# Execute the program, sending the results to output.txt
./newtondemo.exe 1 > output1.txt
./newtondemo.exe 2 > output2.txt
./newtondemo.exe 3 > output3.txt
./newtondemo.exe 4 > output4.txt
./newtondemo.exe 5 > output5.txt
```

```
./newtondemo.exe 6 > output6.txt
./newtondemo.exe 7 > output7.txt
./newtondemo.exe 8 > output8.txt
```

After entering this script and saving it, queue it to the scheduler by issuing the command **qsub newton.job**

This will cause the job to be enqueued for the scheduler to execute.

Once enqueued, you can check on the status of your jobs by issuing the command qstat –u <username>, where <username> is your name on the supercomputer.

Once the job is finished, plot the time, speedup, and efficiency as n varies between 1 and 8.

4. Deliverables

Each person is responsible for submitting a report with the following

1. What did you learn by doing this lab?
2. What things went right and wrong when doing this lab?
3. Include a plot of your execution times, speedup, and efficiencies for both your MSOE laptop and the Glenn supercomputer.
4. What conclusions can you draw from this experience?