Lab Session 5

Submission deadline: Mar 4, 11:59pm

Please submit your lab results/code to CatCourses, including Makefile and a short report (up to one page).

1. MPI hello world

The first task aims to help you compile and run a simple MPI program.

Copy the following program into a file named "MPI_simple.c"

```
#include <stdio.h>
#include <mpi.h>

int main(int argc, char *argv[]) {
    int numprocs, rank, namelen;
    char processor_name[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Get_processor_name(processor_name, &namelen);

    printf("Process %d on %s out of %d\n", rank, processor_name, numprocs);
    MPI_Finalize();
}
```

Compile the above program using the MPI compiler

```
mpicc MPI_simple.c –o hello
```

You can provide a list of machines on which your MPI program should run. For example, if you have four machines, a.ucm.edu, b. ucm.edu, c.ucm.edu, d.ucm.edu (note you can also use IP addresses of the four machines). Please save the following list into a file named "machines".

```
a.ucm.edu
b.ucm.edu
c.ucm.edu
d.ucm.edu
```

You can run "hello" on the machines listed above using

```
mpirun -machinefile machines -np 4 hello
```

-np flag above specifies the number of processes to run. mpirun -help provides details about all other flags and options to run MPI programs.

Please note that you need passwordless ssh setup on the above list of machines in order to use mpirun. In case you do not have it setup already please use

ssh-keygen

More details can be found from "SSH login without password" (http://www.linuxproblem.org/art_9.html)

Note: if you cannot run multiple MPI processes across machines due to firewall setting in the Linux lab, you can run multiple MPI processes within a single machine.

2. MPI ring

Using the above MPI_simple.c as a model, write a program that sends a message around a ring of processes. More precisely, processes i>0 should wait to receive a message from the process with rank i-1, verify that the message contains the integer i-1, replace the integer i-1 with the integer i, and then send the modified message to the process with rank i+1. For example, the process with rank 0 should send a message containing its rank to the process with rank 1, receive a message from the process with rank n-1, and then print the message to the screen.

Test the program with different arguments for –np witch, so that you are confident it works correctly. Measure the execution time with different number of MPI processes.