Lab #2 MPI Programming I

Purpose: to learn how to compile and run MPI program, and write simple MPI programs.

1. (10 points) Compile and run your first MPI program.

a. Create a file (e.g. mpihello.c) using Vi. Enter the following code and save your file.

```
/*
 * MPI Example - Hello world
*/
#include <stdio.h>
#include <mpi.h>

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

        MPI_Init(&argc, &argv);
        MPI_Comm_rank(MPI_COMM_WORLD, &rank);
        MPI_Comm_size(MPI_COMM_WORLD, &size);
        MPI_Get_processor_name(processor_name,&namelen);
        printf("Hello world! I am %d of %d on %s\n", rank, size, processor_name);
        MPI_Finalize();
        return 0;
}
```

b. Compile your MPI code using ***mpicc***. E.g. mpicc -o mpihello mpihello.c

c. Go to webpage: *http://www.cs.gonzaga.edu/iccsl/iccsl_training.html*, and study how to submit a parallel job to the cluster by looking at the Step by Step guide – Launching a Generic Job. The following is a sample scriptfile:

```
#!/bin/bash
#PBS -l nodes=4:ppn=2
#PBS -l walltime=999:00:00
#PBS -o /home/USERNAME/lab2/job_out_lab2a
#PBS -j oe
#PBS -N my_pbs_job

date

export NPROCS=`wc -l $PBS_NODEFILE |gawk '//{print $1}'`
```

```
export MCA_OPTS="--mca btl_tcp_if_include eth0 --mca oob_tcp_if_include eth0 --mca
btl_tcp_endpoint_cache 65536 --mca oob_tcp_peer_retries 120 --mca oob_tcp_listen_mode
listen_thread --mca btl self,tcp"

export PROGRAM="/home/ USERNAME /lab2/mpihello"

mpirun -np $NPROCS -machinefile $PBS_NODEFILE $MCA_OPTS $PROGRAM

date

exit 0
```

d. Run your program multiple times using various number of processes. The grader or instructor will check the performance your program.

2. (20 points) Write an MPI program so that

a. All processes get their process IDs and exchange them with each other using *MPI_Send* and *MPI_Recv*.

b. All processes find out the smallest process ID and print out their results. Output could be:

Process 0: my ID is 18922; the smallest process ID 10150 is the one from Process 2

Process 1: my ID is 18923; the smallest process ID 10150 is the one from Process 2

Process 6: my ID is 11311; the smallest process ID 10150 is the one from Process 2

Process 4: my ID is 10195; the smallest process ID 10150 is the one from Process 2

Process 2: my ID is 10150; the smallest process ID 10150 is the one from Process 2

Process 7: my ID is 11312; the smallest process ID 10150 is the one from Process 2

Process 5: my ID is 10196; the smallest process ID 10150 is the one from Process 2

Process 3: my ID is 10151; the smallest process ID 10150 is the one from Process 2
(The output order is not important)

c. Note that you need to include <sys/types.h> in order to get a process id. You can define and get process id in the following way:

```
pid_t pid;
pid = getpid();
```

d. Run your program with various numbers of processes. The grader or instructor will check the performance your program.