**HW2: MPI Programming**

**Problem 1. [CS4473 and CS5473] (10 points)**

Please complete the two simple exercises created by Dr. Henry Neeman, the Director of OSCER:

* oscer\_hpcexercise\_mpigreetings\_schooner\_20170904
* oscer\_hpcexercise\_mpihelloworld\_schooner\_20170904

Learning outcomes

these two exercises will help you get familiar with the Schooner supercomputer and help you work on the other problems.

What to Hand in

Please submit the output files of your jobs as the proof of work.

* hello\_world\_mpi\_#######\_stdout.txt
* greetings\_#######\_stdout.txt

**Problem 2. [CS4473 and CS5473] (55 points)**

Estimation of parallel computation time (10 points)

Suppose it takes M microseconds to multiply two numbers and A microseconds to add two numbers. Ignore the time for all other operations. How long would a serial algorithm take to compute the dot product of two vectors of length K?

A parallel algorithm can compute the dot product using three processes, including one master process and two worker processes. The master process will collect the two input vectors, send somesubsets to the two worker processes for computation, receive the computation output back, and print out the result. The worker processes will receive the input, work on the computation, and send the output back. Suppose it take T microseconds to transmit a message of arbitrary size from one process to another process and the master process can only send or receive one message at a time. Ignore the time for all other operations. How long would this parallel algorithm take to compute the dot product of two vectors of length K? Suppose K is an even number.

Under what condition, would the parallel algorithm be faster than the serial algorithm?

Please hand in your calculation and a brief explanation.

Socket Programming(20 points)

Please implement the distributed algorithms using socket programming in Python. Use TCP to send and receive messages between the processes. To test, please run all three processes in your computer (but in principle they can run on different computers). The worker and master processes should all print out their input and the output.

For simplicity, please hard-code the two input vectors:

A = [1, 3, 5, 7, 9, 11, 13, 15]

B = [2, 4, 6, 8, 10, 12, 14, 16]

Please hand in your Python codes for both the master side and the worker side and the screenshots of your test. We will test your program by changing the two input vectors in the code.

MPI Programming (25 points)

Please implement the distributed algorithms using MPI programming in C. Please test your program on Schooner and use the same two input vectors as above. The master process should print out the input and output.

Please read the section **“Hints to run the code using batch scheduling”** (after Problem 3) for running your MPI code.

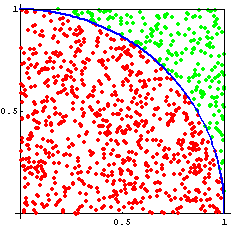
Please hand in your MPI code and the output file of your test run. We will test your program by changing the two input vectors in the code.

Learning outcome:

Both MPI and TCP allows multiple processes to exchange messages and perform coordinated computation. This will allow you to compare how MPI and TCP designate processes and compute nodes, how messages are transmitted in the two methods, and how MPI uses the single-program multiple data programming model (SPMD) and TCP uses the server/client programing model.

**Problem 3. [CS4473 and CS5473] (35 points)**

Write an MPI program using Monte Carlo method to estimate the value of π. For this purpose, we randomly generate a large number of points (x, y), where 0 ≤ x, y ≤ 1. All these points lie in the square with each of its sides of length 1 with its south-west corner at origin.



Note that the area of the square is one and that of the arc of the circle of radius one in the first quadrant is π/4. The point (x, y) will lie in the part of the circle in the first quadrant if x2 + y2 ≤ 1. If the random numbers are distributed uniformly, then the fraction of the points so generated that lie within the circle is approximately π/4.

Your program should generate 2^16 random points (n = 2^16) to estimate π. Each process generates a subset of the random points and determines the number of points that lie within the circular part of the first quadrant. For different processes to generate a sequence of different random numbers, choose a different seed value for generating random numbers for each process. This seed value can be a very simple function of the process id.

Next, the process 0 should sum up the counts of all the points that lie in the sector (red points) calculated by all the processes. Finally, the process 0 should compute π and print out the value of π. The value of π is 4 times of the ratio of (the number of points within circular part)/(the total number of points generated).

Run the program for n = 2^16, with varying processors from 2, 4, 8, 16. Plot a graph of the time taken when you keep the number of points fixed but the number of processes increases.

Please use the point-to-point communication in MPI.

What to submit:

Please submit zip file which includes one code file, one sbatch file, and a PDF file with the results and graph. Format for the submission

OU4x4\_ CS4473\_algorithm.c, OU4x4\_ CS4473\_Report.pdf, OU4x4.sbatch (for CS4473)

OU4x4\_ CS5473\_algorithm.c, OU4x4\_ CS5473\_Report.pdf, OU4x4.sbatch (for CS5473)

**Hints to run the code using batch scheduling.**

#!/bin/bash

#

#SBATCH --partition=debug

#sbatch --nodes=1

#SBATCH --ntasks=4

#SBATCH --mem=1024

#SBATCH --output=mc\_mpi\_%J\_stdout.txt

#SBATCH --error=mc\_mpi\_%J\_stderr.txt

#SBATCH --time=00:30:00

#SBATCH --job-name=mc\_mpi

#SBATCH --mail-user=your-email@ou.edu

#SBATCH --mail-type=ALL

#SBATCH --chdir=/home/oucs0\*\*

#

#################################################

mpirun -n 2 mpi\_many\_msgs 1024 > a.txt

mpirun -n 4 mpi\_many\_msgs 1024 > b.txt

mpirun -n 8 mpi\_many\_msgs 1024 //do not run 8 process in debug partition.(only 1,2,4)

Use debug partition to check the correctness of the code. Please keep in mind, that debug node has limited capacity to run number of process. Maximum time allocated is 30mins and Maximum nodes allowed is just 1.

Use partition=normal for running the final result.

You can run more than one mpirun command in a single batch file and redirect the output to desired files as shown above.

**Problem 4. [CS5473 only] (20 points)**

Ping-Pong to estimate MPI communication time

Write an MPI program to calculate the communication time between multiple processes.

Given an array of integers, Write MPI\_Send and MPI\_Recv to send an entire array back and forth between process 0 and process 1 (two processes) for 100 times. After one process receives the array, it sends this array back to the other process. The array is bounced back and forth between the two processes for a ping-pong ball for 100 times. Record the total time for array sizes 2^10, 2^11, 2^12, 2^13. Please write a sbatch file to create the two processes on the same compute node and make a plot. Please write a sbatch file to create the two processes on two different compute nodes and make a plot. Please compare it with the RTT that you estimated in HW1 using UDP.

What to submit:

Please submit your MPI code and sbatch files and a report containing the two plots (As per “What to submit” for Problem 3).