COSC 6365 – Introduction to High Performance Computing
Fall 2019
Assignment 6
Out: 11/08/19 Due: 11/18/19

For this assignment you should explore different MPI strategies and use message passing interface method calls for communication among processes. MPI is different from OpenMP in terms that it works on distributed memory paradigm and relies on co-operation among processes for efficient computing. Starter codes will be uploaded on blackboard. All the programs are to be run on *Bridges* cluster.

For this assignment you will use as many **compute nodes** as **MPI processes**. For example, to run MPI programs for 56 processes, you will need to request 56 nodes by using the **batch script** (provided with codes) with N (nodes)=56 and n (cores)=56 and specify **one process per node**.

However, before submitting batch jobs, make sure you test the codes for any errors using the "run_*.sh" scripts while requesting minimal number of nodes (1 node for up to 28 processes and 2 nodes for 56 processes).

Problem 1. (Points: 30) Time the roundtrip time of sending a message from one process to another process and having it return the message ("ping" – "pong") for data set size from 1 Byte to 100 MB in steps by factors of 10 (9 cases). Vary the address of the destination process such that for source process 1 the destination process address is 2, 4, 8, 16, 32 and 56 (6 cases). Thus a total of 9x6=54 cases. Report the max, min and average time for each case and **make a plot of the average time as a function of message size for each processor pair**. Derive the **latency** and **bandwidth** for each process pairing based on the average timing values.

1) Use blocking send and receive.
2) Use non-blocking send and receive

Problem 2. (Points: 30) Time broadcast and reduction operations of 1 MB, 10 MB and 100MB sizes to/from 2, 4, 8, 16, 32 and 56 processes **with randomly selected root process**. For each case, report min, max and average time and **plot the average time as a function of the size of the message being broadcast and the data set being reduced respectively**. The codes broadcast.c and reduce.c are provided.

Problem 3. (Points: 30) This problem deals with Group and Communicator creation. The sample code "exercise3.c" is written to work with 4 processes. Modify it to work for 8, 16, 32 and 56 processes and **report the result** (just print the screenshot of output). Also, briefly comment on local and global ranks, and describe one scenario where partitionning processes can be advantageous and one scenario where it may not.

Problem 4. (Points: 60) Use MPI calls to parallelize an algorithm of your choice. Calculate the theoretical efficiency (in terms of execution time) of algorithm **in single threaded/process environment** and compare it to the **efficiency of your parallel version**. A simple algorithm to use can be Minimum spanning tree but you are free to choose any other algorithm (sorting, etc.). Discuss your parallelization strategies and performance insights.

Submit report and codes and scripts together with a README with instructions to run them so we can verify your code runs correctly.