**Individual work only. 150 points. Due April 19, 2018.**

**Objectives:**

To design and implement a collective communication primitive using non-blocking Message Passing Interface (MPI) primitives.

**Program Description:**

1. Implement the broadcast collective communication operation using non-blocking point-to-point message passing primitives provided by MPI using the following algorithms:

   a. linear

   b. ring

   c. double ended ring

   d. tree-based **[Graduate Students Only]**

   For each of the above algorithms plot the time taken by broadcast function and the speedup curve for different number of processes. Note that you cannot use any MPI collective function calls to implement the broadcast function. Use a single driver program to test different algorithms. The driver program should test each broadcast function for different messages sizes also (use a loop inside the driver program to vary the message size from $2^5$ to $2^{20}$).

   Use the following table to include the timing measurements for each algorithm and provide an analysis of these results in your report:

| **PROCESSES** | **1** | **2** | **4** | **8** | **16** | **20** |
|---|---|---|---|---|---|---|
| **BYTES** | | | | | | |
| **32** | | | | | | |
| **64** | | | | | | |
| **128** | | | | | | |
| **256** | | | | | | |
| **512** | | | | | | |
| **1024** | | | | | | |
| **2048** | | | | | | |
| **4098** | | | | | | |
| **8192** | | | | | | |
| **16384** | | | | | | |
| **32768** | | | | | | |
| **65536** | | | | | | |
| **131072** | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| **262144** | | | | | | |
| **524288** | | | | | | |
| **1048576** | | | | | | |

2. If $t_m = t_{send} = t_{recv}$ is the time taken for sending/receiving a message between any two processes develop expressions to compute the total time spent in communication ($t_{comm}$) for each of the above algorithms. [**Graduate Students only**]

**Guidelines and Hints:**

1. Review *Chapter 3. Distributed Memory Programming with MPI* in the textbook, download the source code from the textbook website, compile and test the programs on a Linux system (you can use Vulcan machines in CIS for testing). You can also review the MPI tutorial at https://computing.llnl.gov/tutorials/mpi/.
2. Execute your program on the cluster dmc.asc.edu for 1, 2, 4, 8, 16, and 20 processes and note down the time taken. Also execute the program three times and use the average time taken. Compute the speedup and plot them.
3. Use Vulcan machines in CS for all development and testing only.
4. Make sure you submit jobs to the compute nodes and DO NOT run any jobs on the login node/head node on the ASA cluster. Also make sure you submit your jobs ONLY to the "class" queue.
5. For details about the MPI functions use the MPI standard or MPI man pages.
6. Check-in the final version of your program and the job execution output files (.o<jobid> files) to the CS *git* server and make sure to share your *git* repository with the TAs and the Instructor.

**Program Documentation and Testing:**

1. Use appropriate class name and variables names.
2. Include meaningful comments to indicate various operations performed by the program.
3. Programs must include the following header information within comments:

```
/*
   Name:
   BlazerId:
   Homework #:
*/
```

**Report:**

Follow the guidelines provided in Canvas to write the report. Submit the report as a Word or PDF file. Please include the URL to your *git* location in the report and make sure that you have shared your *git* repository with the TAs and the Instructor. If you are

using specific compiler flags, please make sure to include that in your report as well as README file and check-in the README file to the *git* repository.

**Submission:**

Upload the source files and report (.doc or .pdf file) to Canvas in the assignment submission section for this homework. You can create a zip file with all the source files and lab report and upload the zip file to Canvas. There is NO need to turn in any printed copies in class.

**Grading Rubrics:**

The following grading policy will be used for grading programming assignments:

| | |
|---|---|
| Program Design and Implementation | 50% (program with no compiler errors or logical errors with the required functionality) |
| Program Testing and Performance Analysis | 30% (includes selecting appropriate test cases, performing the tests, tabulating/plotting/graphing the test results, and analyzing performance) |
| Report | 10% (documentation of problem requirements, program design, implementation, instructions for compiling, performance analysis, and test cases) |
| Source Code Formatting | 10% (indentation, variable names, comments, etc.) |

Rubrics for grading the report is as follows:

| | |
|---|---|
| Correct use of English grammar and spelling comprises a baseline requirement for writing. | 20% |
| Clear exposition of the ideas central to the report (e.g., performance evaluation and analysis) is accomplished. | 20% |
| Organizational structure at the high-level, mid-level, paragraph level, and sentence level is reviewed for logic, clarity, uniform continuity, and flow. | 20% |
| The content fulfills the requirements for the technical writing exercise. | 10% |
| Word and language usage is consistent with a scientific report (formality of word choice, person, attention to audience). | 10% |
| Appropriate credit to others (references style and content) is required. | 10% |
| Formatting of the report is appropriate to enable the communication to be effective and professional. | 10% |