Spring 2018: CS 432/632/732 Parallel Computing
Homework-4

150 points. Individual Work Only. Due March 29, 2018 before 12:30pm.

Objectives:

To design and implement a message-passing version of the "Game of Life" program using Message Passing Interface (MPI).

Problem Statement:

The objectives of this homework are:

1. Design and implement a message-passing version of the Game of Life program (developed in Homework-0) using MPI with one-dimensional data distribution. Use non-blocking point-to-point communications to exchange data between the different processes.
2. Test the program for functionality and correctness (the output from the message-passing version must be identical to the sequential version).
3. Measure the performance of the program and optimize the program to improve performance.
4. Determine speed and efficiency of the message-passing version of the program.
5. Analyze the performance of message-passing version with that of the corresponding sequential and multithreaded (OpenMP) versions and include the findings in the report.

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. For testing purposes use the same seed for the random number generator for both the sequential and parallel version of the programs and write the output matrix to a file and compare the output files.
3. Execute your program on the cluster dmc.asc.edu for 1, 2, 4, 8, 16, and 32 processes and note down the time taken. Use the matrix size 5000x5000 and maximum iterations as 5000 for all the test cases. Compute the speedup and efficiency and plot them separately.
4. Use Vulcan machines in CS for all development and testing only.
5. 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.
6. Make sure you comment out any print statements you might have to print the board when you execute with larger problem sizes. Also execute the program three times and use the average time taken.
7. For details about the MPI functions use the MPI standard or MPI man pages.
8. 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%