200 points. Individual Work Only. Due November 7, 2024 before 12:15 pm.

Objective:

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

Problem Statement:

The objectives of this homework are:

1. Design and implement an efficient message-passing version of the Game of Life program (developed in Homework-0). The program must meet the following requirements (you can use the sample solution provided to Homework-1 as the starting point for this homework):
    a. Take the problem size, maximum number of iterations, number of processes, and the directory to write the output file as command-line arguments.
    b. Use dynamic memory allocation to create the two arrays with the ghost cells.
    c. Check if there is a change in the board state between iterations and exit when there is no change after printing after which iteration the program is exiting.
    d. Write the state of the final board to an output file (the directory where the file will be saved must be passed as command-line argument).
    e. Use one-dimensional data distribution (row-wise distribution of the array) to partition the board across multiple processes.
    f. Use MPI_Sendrecv function to exchange data between different processes and use the ghost cells within each process to store the data received from the neighboring process.
    g. Your program should NOT use scatter/gather operations inside the iteration loop. You are free to use scatter/gather outside the loop to distribute the initial array and gather the final array.
2. Test the program for functionality and correctness (the output from the message-passing version must be identical to the sequential version). You can follow the same approach you used in Homework-3 to check the correctness by gathering the final array in the process with rank 0 at the end of the iterations, write the output array from process with rank 0 to a file, and compare the output for different number of processes.

    On the ASC cluster, make sure you write the output files to the scratch directory (/scratch/$USER) and not in your home directory.
3. Execute your program on the ASC cluster for 1, 2, 4, 8, 10, 16, and 20 processes and note down the time taken. Make sure you run each case **three times** and then use **the average of the three runs**. Use the matrix size 5000x5000 and maximum iterations as 5000 and vary the number of processes.
4. Compute the **speedup** and **efficiency** of the message-passing version of the program and plot them separately. To compute the speedup and efficiency, use $S = T_1/T_p$ and $E = S/P$ where T_1 is the time taken by the MPI program with a single process and T_p is the time taken by the MPI program with P processes.

5. Analyze the performance of message-passing version with that of the corresponding **sequential** and **multithreaded** (OpenMP) versions and include the findings in your report.
6. If necessary, optimize the program to further improve the performance and include these improvements in your report.
7. [**CS 581 students only**] Design and implement a new version of the Game of Life program that uses non-blocking point-to-point MPI calls (MPI_Isend, MPI_Irecv, MPI_Wait/Test) to exchange data between different processes instead of using the MPI_Sendrecv function. Compare the performance of the non-blocking version of the Game of Life program with the blocking version developed in Step 1 above and the OpenMP version from earlier homework. Plot the speed and efficiency plots and include the analysis in your 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 also review the man pages or the MPI standard for the various MPI calls and 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 array to a file and compare the output files.
3. You can use the sample solution to Homework-1 or use your own program. If you are using your own program, make sure that your sequential program meets all the requirements listed above (1a-1d).
4. Make sure that you use your local system for all development and testing and **use the ASA cluster for obtaining the performance results**.

   Note that you must include "`module load openmpi/4.1.4-gcc11`" in your SLURM job submission script (i.e., myscript.sh). Here is a sample SLURM job submission script – myscript.sh:

```
#!/bin/bash
source /apps/profiles/modules_asax.sh.dyn
module load openmpi/4.1.4-gcc11


mpicc -g -Wall -std=c99 -o hw4 hw4.c
mpirun -n 8 hw4
```

5. Make sure you submit jobs to the compute nodes using *run_script_mpi* command and remember to use "mpirun " instead of mpiexec on the DMC cluster. Also 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. Check-in the final version of your program, the job submission script (i.e., myscript.sh), and the output files generated by the job script (i.e., myscriptshSCRIPT.o<jobid> - scheduler output file) to the *github* server and make sure to share your *git* repository

with the instructor. Please make sure you are not writing the array to stdout, this would result in large job output files that fill up your home directory.

Program Documentation:

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:
  Email:
  Course Section: CS 481 or CS 581
  Homework #:
  To Compile: <instructions for compiling the program>
  To Run: <instructions for running the program>
*/
```

Report:

Follow the guidelines provided in Blackboard 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 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 (no object files or executables) and report (.doc or .docx or .pdf file) to Blackboard in the assignment submission section for this homework. Make sure you have committed and pushed the final version of your source code, job submission script, and the scheduler output files to the github repository. Include your github URL in the comment section of the submission. **Do not upload zip/tar files** to Blackboard, upload individual source files and project report.

Grading Rubrics:

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

Description	Points
Fully functional and correct program using MPI (Objectives 1-2; also Objective 7 for CS 581 students)	100
Execution of the MPI program for different number of processes (Objective 3)	20
Plot speedup and efficiency results (Objective 4)	20
Analysis of the performance of the MPI program using the speedup and efficiency plots and comparison with OpenMP version (Objective 5)	20
Improvements and optimization of the MPI program (Objective 6)	10
Documentation of the program design, test plan, results obtained, analysis of the results, and improvements/optimizations (Objectives 1-6)	20

Updating git repository with source files, job submission script, and scheduler output files	10