High Performance Computing

Assignment 4

The goal of this assignment is to parallelize Conway's Game of Life using MPI. In the Game of Life, the world consists of a grid of rectangular cells and each cell may be alive or dead. Given a starting configuration of cells, compute the configuration for each generation according to the following rules:

- Any live cell with fewer than two live neighbors dies, as if caused by loneliness.
- Any live cell with two or three live neighbors lives on to the next generation.
- Any live cell with more than three live neighbors dies, as if by overcrowding.
- Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

Note that each cell has exactly 8 neighbors except for those cells on the boundary. For boundary cells, assume that their neighbors are "dead" outside the boundary. See Wikipedia or another source for more information.

(a) **5 marks.** Write a *sequential* program that can be executed as:

```
gameoflife m n numgen infile outfile
```

where $m \times n$ are the dimensions of the grid of cells, *numgen* is the number of generations to simulate, *infile* and *outfile* are input and output file names. The format of these files is a string of mn zeros and ones indicating the configuration of the cells, with 0 indicating dead, and 1 indicating live. The input file gives the starting configuration (generation 0) and the output file should contain the configuration of the last generation simulated (generation *numgen*). If run with 2 generations, the output of your program should look similar to:

```
Reading input file:  myinfile
Number of processes: 1
Starting timer...
Generation 0:  number of live cells:  3
Generation 1:  number of live cells:  5
Generation 2:  number of live cells:  7
Total wallclock time:  0.555 seconds
Writing output file:  myoutfile
```

Test your program on various configurations. For example, for m and n both equal to 5, a possible input file is:

```
0000000000011100000000000
```

(b) **15 marks.** Now write an MPI program that uses the same command line parameters and creates the same output as the above. The program should be run using `mpirun` which specifies the number of processes, p, to use. Your program should partition the grid of cells into p sub-grids of approximately equal size. Use a 1D partitioning, which is simplest.

Note that a single output file should be created. This can be done by sending the final configuration to process 0 and only process 0 writes the output file.

Test your parallel program on various configurations, and verify it is correct by comparing it to your sequential program. By writing the sequential program first, the parallel program should then be easier to write, and you also have a way of testing your parallel program.

For this assignment, you may use multiple processes on a single node (i.e., you do not have to use multiple nodes). This is a way of using the multiple cores on a single node without using OpenMP or any other kind of multithreading.

(c) **15 marks.** Submit a short report of your results as a pdf file. In your report, give evidence that your program works correctly. Also, perform runs with different numbers of processes for a large configuration and show your timings on a log plot and your speedup on a linear plot. Also perform a *weak* scalability study where the grid size is proportional to the number of processes. Plot the speedup in this case. Finally, write the interprocess communication volume for your algorithm for computing one generation in terms of m, n, and p (not including what is necessary for reading or writing files).

Include a listing of your sequential and parallel code in your report.