Fall 2024: CS 481/581 High Performance Computing
Homework-3

200 points. Individual Work Only. Due October 17, 2024 before 12:15 pm.

Objectives:

To design and implement an efficient multithreaded version of the "Game of Life" program using OpenMP.

Problem Statement:

The objectives of this homework are:

1. Design and implement a multithreaded version of the Game of Life program (developed in Homework-1) using OpenMP. The program must meet the following requirements (you can use the sample solution provided to Homework-1):
    a. Take the problem size, maximum number of iterations, number of threads, 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. Create a parallel region only once before the iteration loop, *i.e.,* you should not create a new thread during each iteration.
2. Test the program for functionality and correctness for different number of threads (the values in the array at the end of the iterations in the multithreaded version must be identical to that in the single threaded version if the initial conditions of the board are the same).

   To check the correctness of the multithreaded version, use the same seed for the random number generator and generate the same initial conditions when testing with different number of threads. Then, write the final board output to a file at the end in the main thread, and compare the output generated with different number of threads for a given problem size and number of maximum iterations.

   On the ASC cluster, make sure you write the output to files in the scratch directory (/scratch/$USER) and not in your home directory (if you have not created the scratch directory in Homework-2, then make sure to first create the scratch directory using the command: *mkdir /scratch/$USER*).

   You can use the UNIX command *diff* to compare files, for example, to compare the output of running the problem size 1000x1000 with 5000 iterations on 1 and 2 threads, you can use: *diff output.1000.5000.1 output.1000.5000.2.*
3. Execute your program on the ASC cluster for 1, 2, 4, 8, 10, 16, and 20 threads 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 threads. Compute the **speedup and efficiency and plot them separately**. To compute the speedup, use $S = T_1/T_p$ where T_1 is the time taken by the multithreaded version of the program with a single thread. Compare the

performance of the OpenMP version with different number of threads and include your **analysis** in the report.

4. If necessary, **optimize the program** to further improve the performance and include these improvements in your report. Please make sure to use the **Intel Compiler** for best performance.
5. **Graduate Students only:**

   Test your program using the gcc compiler with appropriate optimizations flags. **Compare and analyze** the performance of your program with the performance results obtained using the Intel compiler and include this analysis in your report.

Guidelines and Hints:

1. Review *Chapter 5. Shared Memory Programming with OpenMP* 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 OpenMP Tutorial at https://computing.llnl.gov/tutorials/openMP/ and the Introduction to OpenMP Tutorial at https://www.openmp.org/uncategorized/tutorial-introduction-to-openmp/.
2. 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).
3. Make sure that you use your local system for all development and testing and **use the ASC cluster for obtaining the performance results**.
4. Make sure you submit jobs to the compute nodes using *run_script*. Do NOT run any jobs on the login node or head node.
5. Note that you have include "module load intel" in your SLURM job script (i.e., myscript.sh).
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 and grader. 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. Also, do not checkin the large output files generated in the scratch directory, leave those files in the scratch 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 (**mhchowdhury@crimson.ua.edu**) and grader (**chgomes@crimson.ua.edu**). If you are using specific compiler flags, please make sure to include that in your report as well as README.md file and check-in the README.md 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. Include your github URL in the comment section of the submission. 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. **Do not upload zip/tar files** to Blackboard, upload individual source files and project report.

Grading Rubrics:

Description	**Points**
Fully functional and correct multithreaded program using OpenMP (Objectives 1-2)	100
Execution of the multithreaded program for different number of threads (Objective 3)	20
Plotting speedup and efficiency (Objective 3)	20
Analysis of the performance of the multithreaded program using the speedup and efficiency plots (Objective 3)	20
Improvements and optimization of the multithreaded program (Objective 4)	10
Documentation of the program design, test plan, results obtained, analysis of the results, and improvements/optimizations (Objectives 2-4)	20
Updating git repository with source files, job submission script, and scheduler output files	10