

CS3210 – Parallel Computing (AY 2016/2017 Sem 1)
Assignment 2: SETL - Search for Extra-Terrestrial Lifeform (20 marks)
Individual Submission due on **11th November 2016**

Objectives:

- Test your ability to design, implement, optimize and evaluate a parallel program written in MPI.
- Reinforce your understanding of the parallel algorithm design process.

In this assignment, we are going to search for aliens in space! The first section describe the problem statement and the sequential implementation. The second section gives the details of your tasks for this assignment.

Section 1. Problem Statement

The alien world is represented as an $N \times N$ square matrix. Each location (cell) represents whether there is a living alien lifeform on that spot. The world evolves generation to generation base on simple rules as described in section 1.1. Our task is to look for a specific lifeform in the world as described in section 1.2.

1.1 Lifeform and Evolution Rules

Each cell in the world can contain a live or a dead (i.e. empty) lifeform. To evolve from one generation G to the next $G+1$ generation, every cell will interacts with its eight *neighbours*, which are the cells that are horizontally, vertically, or diagonally adjacent. The following rules¹ are then used to determine the status of the cell in the next generation:

1. [Under-population] Any live cell with fewer than two live neighbours dies.
2. [Survive] Any live cell with two or three live neighbours lives on to the next generation.
3. [Overpopulation] Any live cell with more than three live neighbours dies.
4. [Reproduction] Any dead cell with exactly three live neighbours becomes a live cell.

¹ This is essentially Conway's Game of Life ruleset

For example, the middle cell can change according to the rules from one generation to the next in the following table.

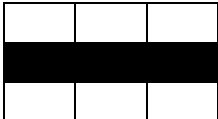
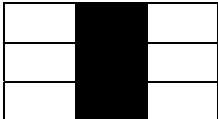
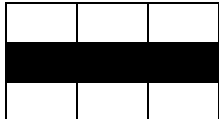
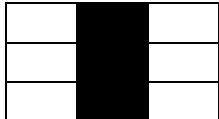
	Rule 1	Rule 2	Rule 3	Rule 4
Generation G				
Generation G+1				

Note that the rules are applied to every single cell in the world. For example, below showed 4 generations of a 9 x 8 world:

Iteration 0	Iteration 1	Iteration 2	Iteration 3

1.2 Lifeform Pattern and Search

Given a lifeform pattern in a $M \times M$ grid, e.g.

	No rotation	Rotate clockwise 90 degrees	Rotate clockwise 180 degrees	Rotate anti-clockwise 90 degrees
Lifeform Pattern				

We will search through the whole world to find **exact match** of the lifeform. As the lifeform can move around, the pattern need to be matched in 4 orientations: i) no rotation; ii) rotate clockwise 90 degrees; iii) rotate clockwise 180 degrees and iv) rotate anticlockwise 90 degrees. For simplicity, we do not care whether the pattern is symmetrical and all four rotations will be searched in all cases. When an exact lifeform is found, the row and column of the upper left corner of the match is reported, along with the iteration number of the world.

For example, using the given lifeform pattern, the result of the search on the first two iterations are:

- Iteration 0, row 6, column 0, zero rotation
- Iteration 0, row 6, column 0, rotate clockwise 180 degrees
- Iteration 1, row 6, column 0, rotate clockwise 90 degrees
- Iteration 1, row 6, column 0, rotate clockwise 180 degrees

1.3 SETL and Sequential Implementation

With the previous two sections, we can now define the problem as follows.

Given:

- **W**, A starting world of size $N \times N$ (always a square)
- **X**, number of generations (iterations for the world evolution)
- **P**, A lifeform pattern of size $M \times M$ ($2 < M < N$)

The world **W** will evolve for **X** iterations. For each iteration, we are going to search for the lifeform pattern **P** in all four orientations.

The given **SETL.c** program is a simple sequential solution. It takes in 3 command line arguments:

- **<world file>**: Starts with a number N and followed by N rows of N characters representing the initial lifeforms in the world. 'X' and 'O' (alphabet Oh) is used to represent live and dead cell respectively.
- **<iterations>**: Number of iterations (generations) of evolution, including the initial world (i.e. generation 0 is counted).
- **<pattern file>**: Format is similar to <world file>. Contains a number M and followed by M lines of M characters each. Specify the lifeform pattern to be searched for.

The program reports ALL matches by encoding the output as a four digit sequence separated by ":". The format is <World iteration>: <Row Number>: <Col Number>:<Rotation>. The <Rotation> uses {0, 1, 2, 3} to represent the four orientations of lifeform pattern: {No rotation, 90 degree clockwise, 180 degree clockwise, 90 degree anticlockwise}.

For example the results from previous section will be reported as:

0:6:0:0
0:6:0:2
1:6:0:1
1:6:0:3

A few notes on the sequential implementation:

- Used the "halo" approach to handle boundary condition in the world. There is an empty row at the topmost and bottommost location and empty column at the leftmost and rightmost location. Effectively, we use a $(N + 2) \times (N + 2)$ matrix to represent a $N \times N$ world.
- The four orientations of lifeform pattern are stored as separate matrices.
- A simple circular linked list is used to store the match results.
- Several design choices were made to facilitate your work later:
 - o The matrices (world and lifeform pattern) are allocated in a contiguous stretch of memory.
 - o The code is heavily modularized to allow quick adaptation.
 - o No "clever" algorithm is used.
- You can add additional flag during compilation to turn on additional debug messages:
 - o "-DDEBUG" : Print out the world generations.
 - o "-DDEBUGMORE": Print out the pattern searching iterations.
 - o Note: the debug messages are only intended for tiny world, don't use them for large worlds!

Section 2. Parallel SETL in MPI

Your task is to parallelize the sequential SETL using MPI. There are two target machines for this assignment: a) the 3-node cluster in the lab b) the NSCC supercomputer.

The only restrictions that you need to respect are:

1. The root process must distribute the relevant data to other processes, i.e. the other processes cannot directly read the files for the initial data.
2. The basic algorithm of world evolution and lifeform matching cannot be changed. i.e. we are not looking for "better" algorithm but an optimized parallelization approach with essentially the same core processing.
3. OpenMP / CUDA cannot be used. This is a pure MPI assignment.
4. Timer should starts **before data distribution** and **ends after result printed on screen**.
5. Final results should be sorted by iteration → rotation → row → column, i.e. the same as the sequential version.

The following data sets are given:

World Files	Pattern Files
simple.w: 20 x 20 tiny world. The top left corner is the same as the example used in section 1.2.	glider3.p: 3 x 3 file containing the famous "glider" shape.
gliderArmy.w: 100 x 100 with multiple gliders on regular intervals.	glider5.p: 5 x 5 file containing the famous "glider" shape in the center.
random1000.w: 1000 x 1000 world with about 20% live cell randomly scattered.	
random3000.w 3000 x 3000 world with about 10% live cell randomly scattered.	

You are free to generate your own world / pattern file and share it with your classmates. The given **genWorld** program can randomly generate world of any size for your own testing. Note that the world file takes up significant space at large size (3000 x 3000 = ~9Mb).

2.1 Submission timeline

As this assignment is quite challenging, a "last-minute over-night coding camp" will definitely not be enough. To encourage a systematic, timely approach to the assignment, we have inserted a "kick-start check" in week 11.

Kick-Start Check (26th October) [3 marks]
<ul style="list-style-type: none">- Demonstrate to your lab TA (during tutorial / lab session)- Compilation and running of your MPI program. See 2.2 about the compilation and running requirement.- You only need to make sure your MPI program use more than 1 process across two machine. Speedup is not important (i.e. it is OK even if it is slower than sequential version at this point).
Final Submission (11th November, 5pm) [17 marks]
<ul style="list-style-type: none">- Full submission to IVLE workbin. See section 2.2 for complete submission requirement.

2.2. Final Submission and Evaluation

Prepare a **zip file** using your **student id (matric no)**, e.g. A01234567X.zip, which contains:

SETL_par.c	Your optimized parallel MPI version of SETL with appropriate comments, modularization and indentation. Please remove all debug messages .
makefile.lab	For compiling your SETL_par.c on the lab machine.
runfile.lab	The command to run your SETL_par.c. We will execute this command from node 1 (intel i7).
machinefile.lab rankfile.lab	Additional configuration file that you need for the execution in the lab.
makefile.nsc	For compiling your SETL_par.c on NSCC.
SETL.pbs	PBS job script for submitting your program to NSCC supercomputer.
SETL.pbs.o SETL.pbs.e	Summary reports from your execution on the NSCC supercomputer.

as2_report.pdf	Contains: 1. Describe your parallelization approach for SETL. 2. Execution results and analysis on lab machine and NSCC supercomputer.
----------------	--

Score breakdown for final submission **[17 marks]** :

Design [5 marks]	Evaluation of your approach on partitioning, communication, agglomeration and mapping. We look for well thought out design with strong rationale .
Implementation [2 marks]	Code quality. We look for modularization, documentation and good coding style.
Result Analysis [4 marks]	The execution timing and analysis from various test cases. We look for careful collection of results and thoughtful analysis of the result.
Performance [6 marks]	[3 marks] for performance on lab machine. We compare the speedup you achieved with your classmates'. The faster the better (duh!). [3 marks] for performance on NSCC machine.
Bonus [2 marks]	[1 mark] for the best two performing implementation in the lab. [1 mark] for the best two performing implementation on the NSCC machine.

The final evaluation is based on a 3000 x 3000 world with a lifeform pattern size $3 \leq M \leq 5$ and with iterations ≥ 100 .

Submit the .zip archive to IVLE workbin by 11th November, 5pm.

~~~ End of Assignment 2 ~~~