CMPE300 2019-2020 Fall
OpenMpi - Cellular Automata - Game Of Life Project

Ömer Faruk Özdemir – 2016400048

13.12.2019

Introduction:
This report contains Introduction, Program Interface, Program Execution, Input And Output, Program Structure, Examples, Improvements And Extensions, Difficulties Encountered, Conclusions, and Appendices sections.

This a parallel programming Project done with mpi4py package in python3, using OpenMpi.
Cellular Automata model with Game of Life rules is implemented with parallel processors in OpenMpi.
I use the Periodic and Checkered model given in the Project description.

Program Interface:
Problem and the model is given with the Project description.
User gives an input as initial state of the 360x360 map. Though my code is implemented dynamically, user can change the map size with changing BIGEDGE global variable in my code.
OpenMpi environment should be installed and mpi4py pip package should be installed with
pip3 install mpi4py

Program Execution:
Program is run with the following command,

mpiexec --oversubscribe -np [M] python3 test.py [input] [output] [T]

Example:

mpiexec --oversubscribe -np 17 python3 test.py rand.txt output3.txt 19

[M] is number of processors, [input] is input, [output] is output, [T] is number of rounds.
[M] needs to be equal to c^2+1 for some c, where c divides 360 and a multiple of 2.

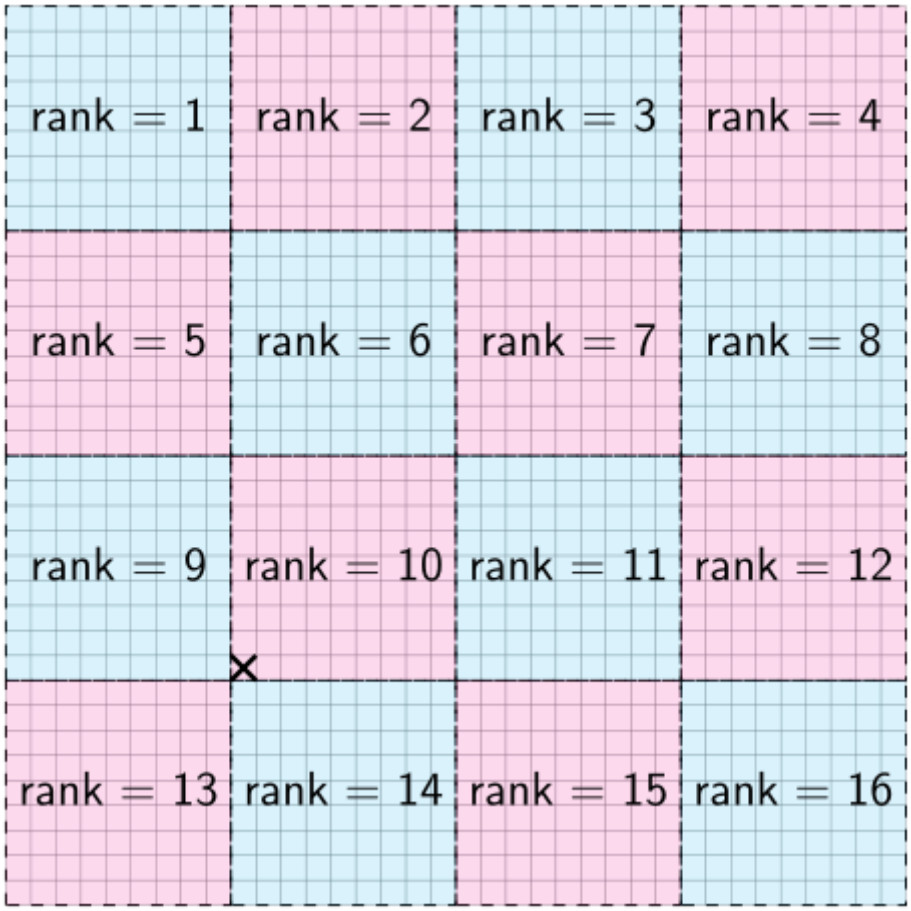
Input and Output:
The input.txt will contain the initial state of the map of size 360 × 360 with;
• rows separated by a single new line (\n) character,
• each cell on a row separated by a single space ( ) character,
• each cell as a 0 or 1 for emptiness and life.

Output is T number of iterations of the input.

Program Structure:
Reading input and writing outputs is done with central processor, distribution of data is done via central processor.
First central processor, reads the input then data is distributed to processors, then communications and round processings is done with processors, then data is sent to central processor, then the central processor writes the output.

- First central processor reads the input,
- Then sends the parts to processors,
- Then processors do the communication(communication is explained below)
- Then round processing is done with the received data from neighbours
- These last 2 steps is done for every iteration
- Every processor sends their data to central processor one by one
- Central processor writes the output

Communication:
Processors are splited in a Checkered manner.



My structure is similar to toroid. When gone to out of bounds for looking a neighbour you teleport to other end of the map. And toroid exceptions is checked and considered in my code.

Processors with odd no columns send their right column to right neighbour,
Processors with even no columns receive data from their left neighbour.

Processors with odd no columns send their left column to left neighbour,
Processors with even no columns receive data from their right neighbour.

Processors with even no columns send their right column to right neighbour,
Processors with odd no columns receive data from their left neighbour.

Processors with even no columns send their left column to left neighbour,
Processors with odd no columns receive data from their right neighbour.

Processors with odd no rows send their up row to above neighbour,
Processors with even no rows receive data from their below neighbour.

Processors with odd no rows send their down column to below neighbour,
Processors with even no rows receive data from their above neighbour.

Processors with even no rows send their up row to above neighbour,
Processors with odd no rows receive data from their below neighbour.

Processors with even no rows send their down row to below neighbour,
Processors with odd no rows receive data from their above neighbour.

Processors with odd no rows send their up-right corner to above-right neighbour,
Processors with even no rows receive data from their below-left neighbour.

Processors with odd no rows send their up-left corner to above-left neighbour,
Processors with even no rows receive data from their below-right neighbour.

Processors with odd no rows send their below-right corner to below-right neighbour,
Processors with even no rows receive data from their up-left neighbour.

Processors with odd no rows send their below-left corner to below-left neighbour,
Processors with even no rows receive data from their above-right neighbour.

Processors with even no rows send their up-right corner to above-right neighbour,
Processors with odd no rows receive data from their below-left neighbour.

Processors with even no rows send their up-left corner to above-left neighbour,
Processors with odd no rows receive data from their below-right neighbour.

Processors with even no rows send their below-left corner to below -left neighbour,
Processors with odd no rows receive data from their above-right neighbour.

Processors with even no rows send their below-right corner to below -right neighbour,
Processors with odd no rows receive data from their above-left neighbour.

Improvements and Extensions:
Number of processors is currently limited to divider of BIGEDGE and multiple of 2. This restriction can be removed, but it would be really hard.

Difficulties Encountered:
Difficulties in this Project was considering out-of-bounds communication sendings.

Appendices:
Project description.