ACSE 6 Assignment Report

Yue Wang(yw2619)

Saturday 22nd February, 2020

1 Code Structure

In order to implement the parallel version of Conway's Game of Life under MPI, I choose the grid decomposition as the domain decomposition strategy and peer-to-peer communication as the communication strategy. All functions are generated in the single file(**submit.cpp**).

1.1 Grid Decomposition

In order to divide the whole domain into rectangle sub-grids, I firstly use the function: *void find_dimensions(int p,int id,int& rows, int& columns)* to arrange p processors into a 2D grid.
Then, according to the number of rows and columns of this 2D gird, *void decomposition(int height, int rows, vector< vector<int> > &rows_responsible_list)* is used to calculate the size of sub-grid that each processor is responsible for. For example, 4 processors is arranged into a 2D grid of size 2×2. So that, each processor is responsible for a sub-grid of size 4×4.

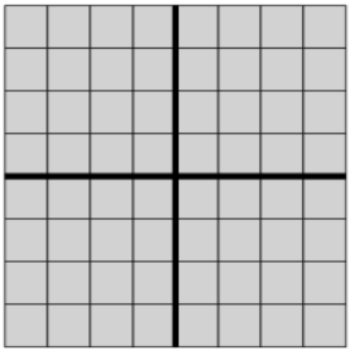

Figure 1: decomposition by 4 processors

1.2 Communication Strategy

For the peer-to-peer communication, I choose the non-blocking point to point Communications and create two separate arrays to receive and send data. *void send_neighbor(int i, int j, int com_id, bool* current, MPI_Request* request, int &cnt)* is responsible for transferring a layer of grid cells just inside the edge of the domain from the neighbouring processors. *void update_neighbor(int i, int j, int com_id, bool* grid_ext, MPI_Request* request, int &cnt)* is responsible for receiving a layer of grid cells situated just outside the domain back from the neighbours.

2 Discussion

To test the Efficiency/Speedup of this program, I set the domain size as 40000×40000,10000×10000, and 5000×5000 respectively, and for each experiment, I run 20 steps of life of game with periodic boundaries. For the processor number, I start it from two to seventy-two. Because for the domain of size 5000×5000, the running time by using 72 cores is less than one second.

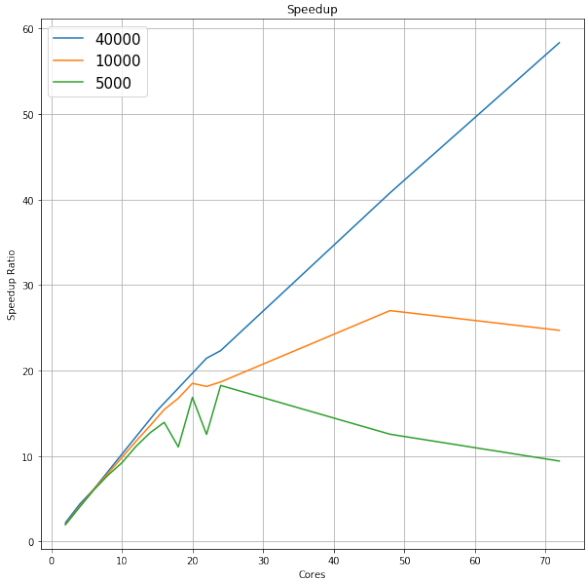

Figure 2: Parallel Speedup

From the figure above, it is shows that, when processor number is less than 10, there is no significant difference of speedup ratio when dealing with different size of domain. When processor number is larger than 20, for the small size of domain(5000×5000), speedup ratio stops to increase , since it costs much more time to build communications than do calculations. Besides, for the larger size of the domain(40000×40000), the speedup ratio still increases dramatically when processor number is larger than 20. In conclusion, the bigger problems will have a better performance of parallel speedup.

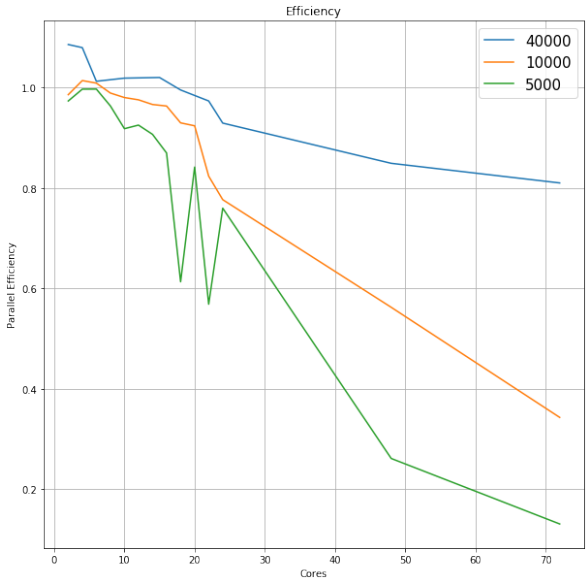

Figure 3: Parallel Efficiency

For the parallel efficiency, it implies the same tendency with the size of the domain. The bigger problems hold a higher efficiency when using same number of cores. In addition, regardless the size of domain, parallel efficiency always decreases when the number of cores used increases.