Write-Up for Project 3

Name: Cuiqing Li
Email: cli92@jhu.edu

Solutions of Questions

Part I

- (a) Answer: Here my steps how I handle the deadlock-free problem. In my solution, I firstly divide the original matrix Grid into num_processor parts based on rows. At the same time, as for every process will get a copy of sub-grid which contains elements of some of rows of original grid. Specifically, if the process id is ID, then that process will get the elements from ID*col_size to (ID+ row_size/num_process)*col_size). After that, in order to make sure deadlock-free, process i whose id is even(i%2==0) will send its bottom row to next process i+1 firstly. Also, process i whose id is even will send its top layer to the previous process i-1 secondly. At the same time, processes whose id are odd will receive information from other process, after receiving those information, those process whose id are odd will continue to send their upper layer to the previous even process and bottom layer to the next even process. If so, it can help us make sure we didn't have deadlock problem. Since they won't receive and send at the same time for every process. After that, before we continue to the next steps, we use barrier function to make sure everything above is done. If not, when we update Grid array probably use some un-updated sub-grid information above, which probably has race condition problem. Finally, we use MPI to gather all the updated grid information.
- (b) Answer: Suppose we have num_process of processes in our system, and the grid's row size and column size is N. As for the number of messages passed per process for a single iteration of the game, we need to have following consideration. First of all, it will send one row of elements to the previous process, and it will also send one row to the next process. In addition to that, it will send sub-grid defined for every process to the root process. The sub-grid's size is (N/num_process+2) * N, so add the sending messages together, we need to send O((N/num_process+4)*N) messages totally for every process in the every iteration. As for the amount of memory every process uses for every iteration, we need to take O((N/num_process+2)*N) size memory, and it is equal to the size of sub-grid defined in every process. If we can ignore the constant value, the number of messages passed per process for a single iteration is O(N^2/num_process), and the amount of memory every process uses for every iteration is O(N^2/num_process).
- (c) Answer: the allocated memory of sub-grid in the every process is dependent on the global grid array since it needs to get information based on grid array. Also, sub-grid array also depends on other sub-grids in the other process since it needs to update informatimation based on their neighboring information. In order get the neighboring information, it needs to receive information of other sub-grids from other processes. Reversely, the global grid array is also dependent on the those updated sub-grids of every process since it needs to update its new state based on those updated sub-grids from all the processes.

Part II

Generally, as for spatial decomposition, it has advantages and disadvantages: Firstly, if we have many processors in our system, and every process will send fewer messages to the other processes in every iteration. If so, it can save some time. However, since spatial decomposition needs to send left,right,top and bottom layers to the other process, so it is harder to control deadlock-free problem. The deadlock problem will be more likely to happen using this way.

- (a) Answer: Suppose the dimension of Grid is N and the number of process is num_process. As for the number of messages every process needs to send, they do have some differences. For the horizontal implementation, we need to send O(2N) elements to the other processes. For the spatial decomposition, O((N/sqrt(num_process)*4-2)) elements to the other process since square has 4 boundaries in this case. Thus, it really depends on the number of process. If the number of process is very large, the second implementation (spatial decomposition) is better since it will send fewer elements for every process. If we don't have too many processes, then they need to send very similar number of elements to the other process for every process.
- (b) Answer: Suppose the dimension of Grid is N and the number of process is num_process. For the horizontal decomposition way, every process need to spend O(N*N/num_process) memory to store updated elements. For the spatial decomposition way, we also need to spend O(N/sqrt(num_process)*N/sqrt(num_process)) = O(N^2/num_process) memory size to store updated sub-grid for every process. Thus, they share similar size of memory overhead.
- (c) Answer: Suppose the dimension of Grid is N and the number of process is num_process. Set d = sqrt(num_process), so every sub-grid's dimension is N/d,so we need to divide grid into num_process number of sub-grids. For every sub-grid, we denote it has some subgrid[j] for some integer j(0<=j<num_process). If the process id is i, then it has subgrid[(i/d)*d + i%d]. Thus, this process needs to send its sub-grid's upper row to process[(i/d-1+d)*d+i%d], and send its left-most column to the process[(i/d)*d+(i-1+d)%d], and send its right-most column to process[(i/d)*d+(i+1)%d], and send its bottom row to the process[(i/d+1)%d+i%d]. In order to make sure deadlock free, we can think of original grid has been divided several squares, and those squares(sub-grids) are displayed row by row and column by column. Firstly, we can let the even row's sub-grids send their top and bottom layer to the target process, and those receiving process has odd rows' sub-grids. After that, the odd rows' subgrids send their top and bottom rows' information to the target processes (which are even rows' subgrids). After finishing the process above, we can let the even columns' sub-grids to send left-most and right-most columns to the target process. After target process have received the information, and the odd columns' sub-grids send left-most and rght-mosy columns' information to the process which has corresponding even columns' sub-grids. If so, we can make sure it is deadlock free.