ACSE6 Report

Road of solution

- I constructed basic class for representing data structure of cells and basic updating (serial code) method for cells.
- Applying stripe decomposition strategy to cut matrix of cells with MPI to parallelly updating cells. I realized this by inheriting basic class to write new methods.
- Applying grid decomposition strategy to cut matrix of cells with MPI to parallelly updating cells. I realized this by inheriting basic class to write new methods.

Functionality

My solver with MPI can solve arbitrary size of matrix with arbitrary processors (Basically, you need to ensure the number of cells is larger than that of processors.

In general, when you call grid solver to update cells, if you set the number of processors as prime number, the solver will automatically call stripe solver (because prime number cannot be factorized to multiplication of two number). Otherwise, the solver will factorize processors to multiplication of two number.

Post Processing

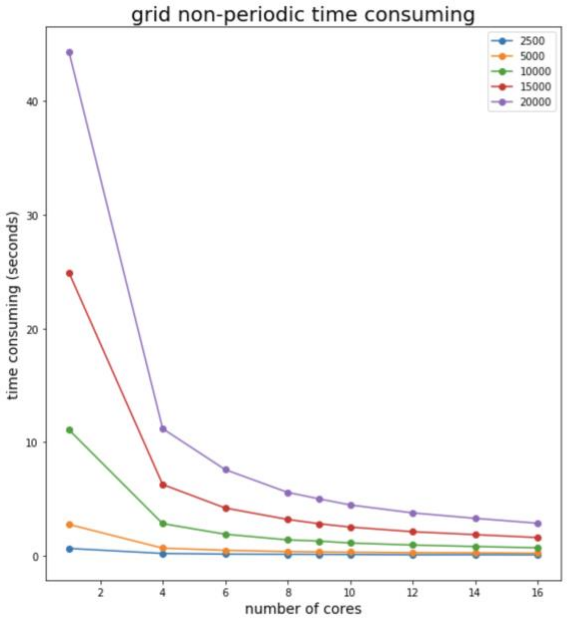
I wrote function to output files and function to record time in two MPI classes used for post-processing. You can control it by adjust the input parameters when you call method.

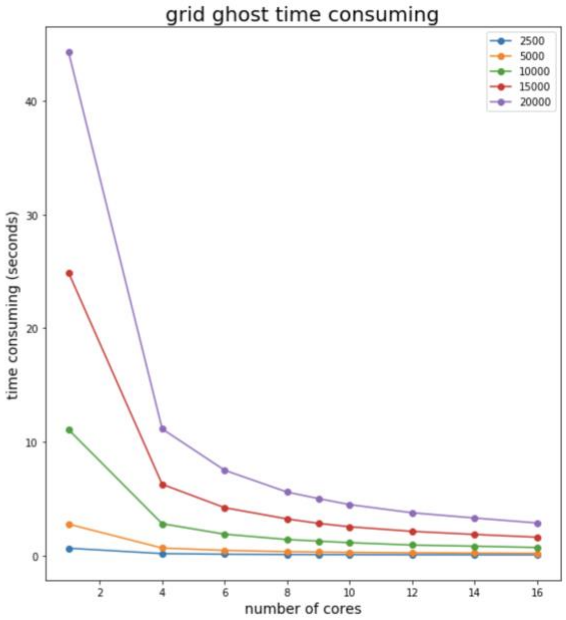
After outputting files, they are stored as “txt” format in local system disk. Then, they are inputted into python to make animation.

After outputting times, they are stored as “txt” format in local system disk. Then, they are inputted into python to plot speedup ratio line and parallel efficiency line.

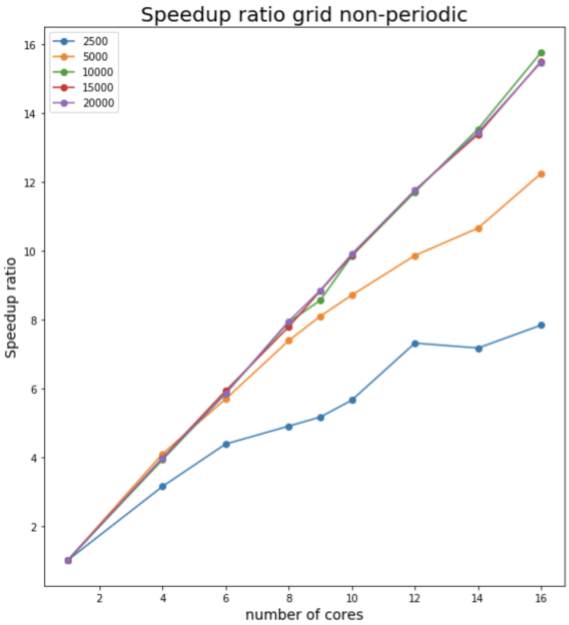
Performance Analysis

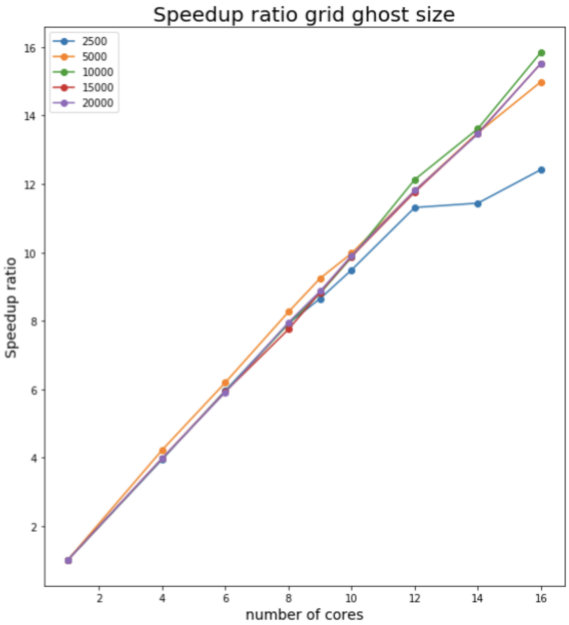
Time consuming





Speedup ratio



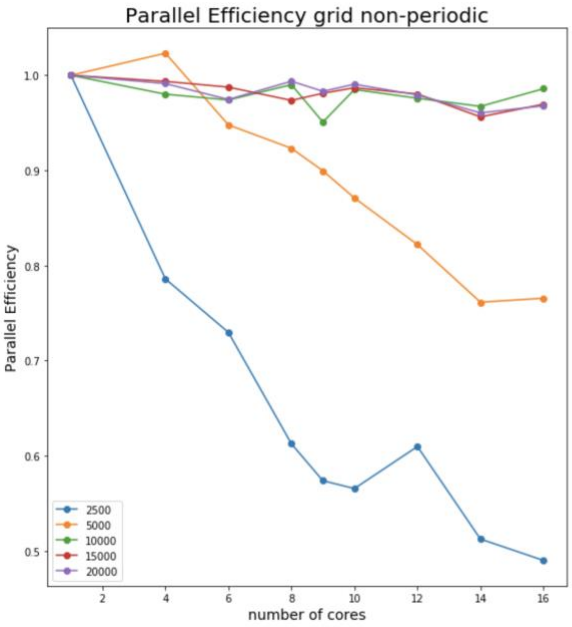


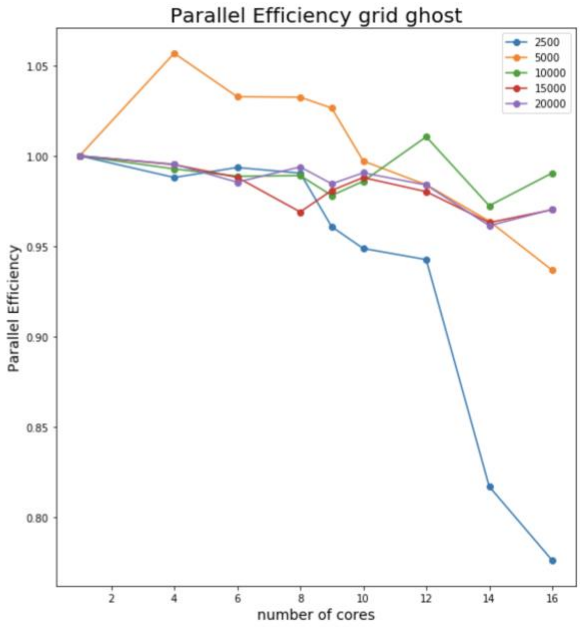
From pictures, we can find that the most of lines are straight lines, that is because I test the time of updating processes.

Meanwhile, we can find that the slope of line is trending to 1 as the size of matrix increase. That is because as the size increase, the performance of MPI is trending to stable.

And there are some noises in line leading to some points out of the trend of lines.

Parallel Efficiency





From the pictures, we can find it coincides the theory that parallel efficiency will usually drop as the number of cores used increases. But there are some unusual points for matrix of 5000 by 5000 sizes.

We can find there are some noise points to out of lines, I think it is from the unstable property of MPI test. When you test for several times for same condition for matrix, it will produce time in a range, so It is a little unstable about record time consuming.