Performance Report

In this assignment I implement a parallel version of Conway's Game of Life. I use striped decomposition to decompose the whole domain into several horizontal stripes and each processor is responsible for one stripes. And in every stripe, there a certain number of rows and columns, which depend on the size of whole domain and number of processors we use to run the program.

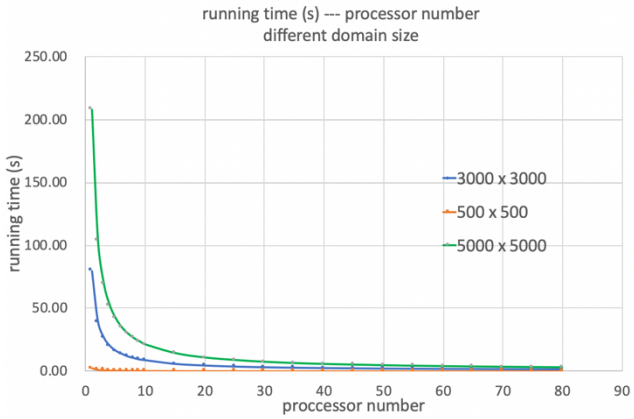
In this report, I analyzed: ***A****. how running time, speedup ratio and parallel efficiency change with the size of the problem and the number of processors used.* ***B.*** *how they change with different shape of domain but same size of problem(same total number of cells) and number of processors used.*

For number of processors used, the number varies from 1 to 80. Between 1 processor and 10 processors with an interval of 1, while between 10 processors and 80 processors, the interval is 5. In another word, by using HPC system, I chose 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80 as number of processors used to conduct the test.

A. How running time, speedup ratio and parallel efficiency change with the size of the problem and number of processors used.

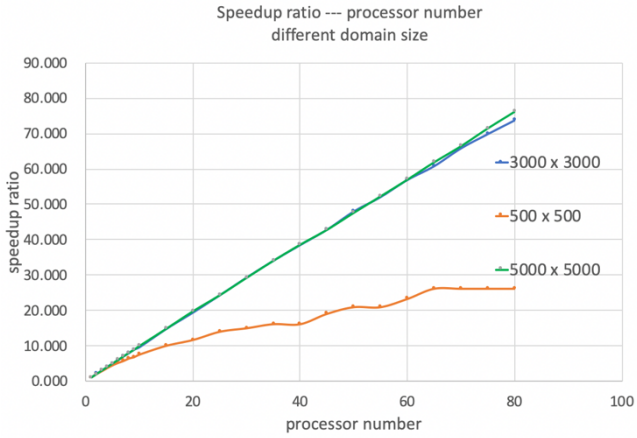
I choose 3 different size, which are *500 x 500*, *3000 x 3000*, and *5000 x 5000*, to test how different size of problem will vary running time, speedup ratio and parallel efficiency.

(1) Running time:



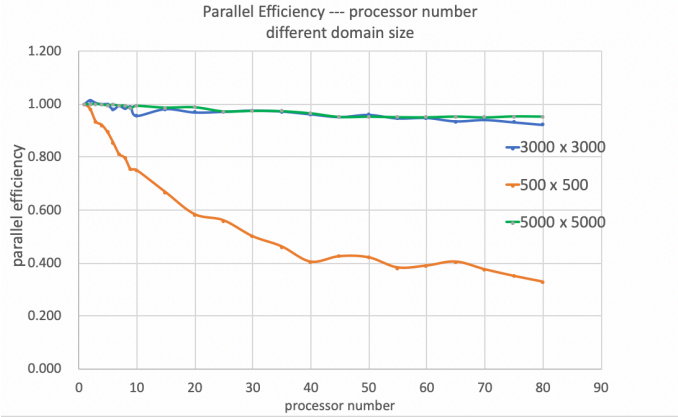
As is shown above, the running time for all of three problem in different problem sizes experience a decreasing trend, however for relative large size (5000 x 5000) drops sharply, while for relative small size (500 x 500), only trivial decrease can be find in running time. The common thing is that with number of processors increase, at the very beginning (number of processors < 10), the rate of decrease is much more significant than later (number of processors > 10).

(2) Speedup ratio:



On the one hand, for problem with 3000 x 3000 size and 5000 x 5000 size, the figures for speedup ratio appear quite similar with the growing rate become little bit slowing down for 3000 x 3000 size when number of processors approaching 80. On the other hand, for small size of problem, the speedup ratio grows slightly and even becomes steady when large number of processors used.

(3) Parallel efficiency:



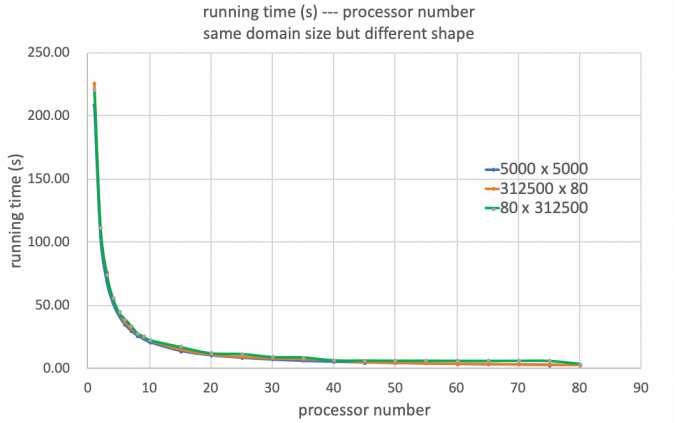
Owing to *efficiency* = *Speedup/N*, three different sizes of problem also showed relative similar pattern

with figures for Speedup ratio. For big size problems, the efficiencies are quite high and even approach ideal efficiency(=1). But for small size problem, efficiency drops sharply at beginning and gradually slows down when number of processors increase.

B. How running time, speedup ratio and parallel efficiency change with different shape of domain but same size of problem (same total number of cells) and number of processors used.

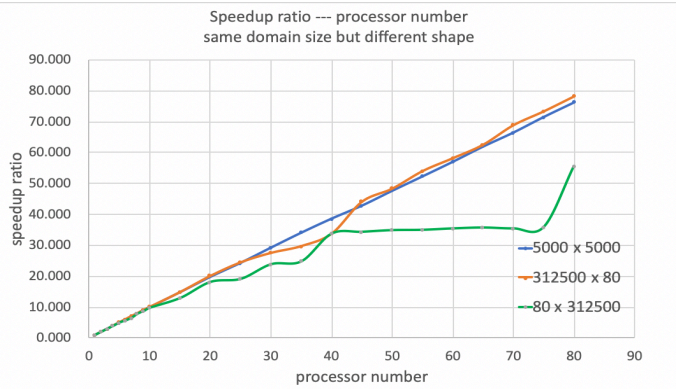
Now I want to test whether shape of domain can vary running time, speedup ratio and parallel efficiency. I fixed total number of cells of game as 25,000,000, use 3 different shapes which are 5000 x 5000 (square), 312500(rows) x 80(columns) which is extremely tall and thin, and 80(rows) x 312500(columns) which is extremely flat and wide.

(1) Running time:



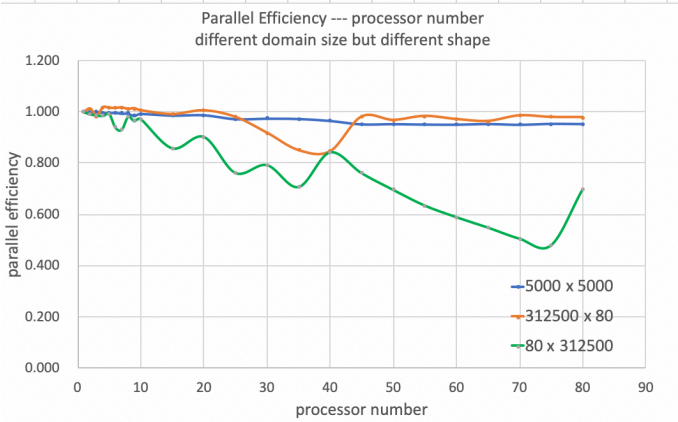
As is shown above, in panoramic view, the running time is quite similar between all of three different shapes.

(2) Speedup ratio:



Problems in 5000 x 5000 and 312500 x 80 have ideal speedup ratio with increasing number of processors, while speedup ratio in problem in 80 x 312500 only witnessed a slight growth when number of processors is large (>40) and finally soar when number of processors reached to 80.

(3) Parallel efficiency:



Both problems in 5000 x 5000 and 312500 x 80 have great efficiency, approaching to 1, although there are some fluctuations in problem in shape of 312500 x 80. While in problem of 80 x 312500, the efficiency declines with fluctuation and finally increases to about 70% suddenly when I use 80 number of processors.

C. Analysis and Explanation

For domain with 500 x 500 in size, the problem size is quite small and it will not take to much time to simulate by using serial code (less than 5 seconds). If we use parallel code with large number of processors to solve such small-size problems, it will not be efficient enough because each processor may consume time to communicate with their counterparts. Although it will not be a long time waiting for MPI send and receive, the time spent on communication still accounts for a large proportion of the total time to solve the problem. Acceleration of parallel computing can hardly offset the time spent on communication. That is why comparing with domain in 3000 x 3000 and 5000 x 5000, problem in 500 x 500 has quite low parallel efficiency and insignificant acceleration effect.

For domains with 3000 x 3000 and 5000 x 5000 in size, it will take about hundreds of seconds to solve the problem when using serial code. Although MPI communication will take some time to complete, comparing with the total time for simulation, it is trivial enough. Owing to the time spent on MPI communication, the parallel efficiency can never reach 1. Logically speaking, the efficiency of size in

5000 x 5000 problem should higher than that of size in 3000 x 3000, but they can hardly find any difference. I assume that both 3000 x 3000 and 5000 x 5000 sizes are large enough, and time consumed by MPI can be omit when compared with their problem solving time.

For problems in different shapes, in running time figures, we can hardly find any difference between our test samples. But that does not mean they are similar in performance, because when number of processors is large, the running time is extremely short and trivial deviation can make great difference. In running time figures difference is not obvious, so we should focus on charts for speedup and efficiency. Admittedly, in this test, shape in 5000 x 5000 and shape in 312500 x 80 performed similar, although some fluctuation can be found in shape of 312500. Fluctuation can generate from several reasons which I will analyze later. Apparently, shape in 80 x 312500 behaved quite differently with its counterparts. It is because I decompose the total domain by horizontal stripes, and when domain becomes extremely flat and wide, in each communication, there are big amount of data to be sent and received. In this example (80x312500), there are 625000 numbers to be sent and 625000 numbers to be received for each processor in every single step. While only 160 numbers to be sent or received in shape of 312500 x 80 problem. Furthermore, for problem in shape of 80x312500, the soar in the final can also be explained. When I use 40, 50 or 60 as number of processors, always some processors assigned for 2 rows and some only assigned for 1 row. Despite the processors with 1 finish their jobs, they also need to wait for receiving the data from those processors who were assigned for 2 rows to complete their job and receive data from them. But things changed when I call for 80 processors, every processor only responsible for 1 row, and there is no need to wait for long time after finish their own job because of evenly divided workload. So that is why both speedup ratio and efficiency soar suddenly when I got 80 processors used.

In the processing of my test by using HPC system, I also find that HPC system may work unstable. It may depend on current workload. When workload is heavy or have lots of users in the same time, the running time for same program with same configuration will longer and in this situation, fluctuation showed in above graphs can generate. Admittedly, sometimes such fluctuation can be quite significant. Therefore, we I collect running time from HPC system, I always discarded the data with obvious deviation and measured for several time and calculated the mean of data for identical test.

D. Conclusion

Parallel version code for Conway's Game of Life is suit for the simulation with large size game domain (Ideally more than 3000 x 3000). Horizontal striped decomposition is suit for most of situation of Conway's Game of Life and have great efficiency, except for those problem with extremely flat and wide domain. I assume that vertical striped decomposition can hardly perform well in solving problems with extremely tall and thin domain. Logically speaking, grid decomposition can solve the problem in any shape but the threshold of minimal size of problem may higher than striped decomposition because in grid decomposition, each processor should communicate with its 8 neighbors and it will definitely take more time on communication. All in all horizontal striped decomposition can solve most of large size problem of the Game of Life and it performs quite well. Shape of problem can change speedup ratio and parallel efficiency, but what kind of change it may lead depends on what kind of domain decomposition ways parallel code use.

Running time decreases with number of processors used increase, while increase with problem size rise. Speedup ratio increases with growing number of processors used and the increase rate will be more and more moderate. In addition, large problem size may enhance the increasing rate of speedup. Parallel efficiency decrease with more and more processors used, and small size problems can make it drop much more sharply than large size problems.