Introduction for the codes

I wrote the serial version (you can find the single_proc.cpp in the serial folder), the striped decomposition solver and grid decomposition solver (they are all in the "DataType.h", "DataType.cpp", "tidy.cpp"). In addition, I also uploaded the version for high performance calculation called "game_class.cpp", and you can see the Makefile and pbs file on github. In the import data folder, there is a "ipynb" file called "video_produce. ipynb" (this file generates the figure 1), this file can merge the output files generated by each processor and make video and gif pictures of the whole iteration process. If you want to test the solution of this game, just change the bool variable verify into true, after finishing running the tidy.cpp the screen will show you whether the result is correct.

My codes can generate arbitrary size cells and the status of cells is random, and in the "test_proportion" folder you can see another version 'game_class.cpp' file, this file can generate fix initial proportion of live cells when you change the distribution(probability) function. In this folder, "get_picture. ipynb " and "different_init.ipynb" can generate the Figure 3 below. In the subordinate directory "import data" folder the "video_produce. ipynb" can generate the figure 2.

In folder "performance_analysis" I store the high-performance data (including grid domain and striped domain for different matrix size and different core number, in addition, I add the serial related data). There is a "performance_analysis. ipynb" file handling the data in this folder and producing the performance analysis figures. (Fig5,6,7)

I upload a video called "import_data1582366604.mp4" which can show you the evolution of this game, and you can also see the gif form in the import data folder called animation.gif

In the codes, when the spited columns or rows are equal to 1, it will use striped solver. (you can see this in "tidy.cpp")

Evolution pattern

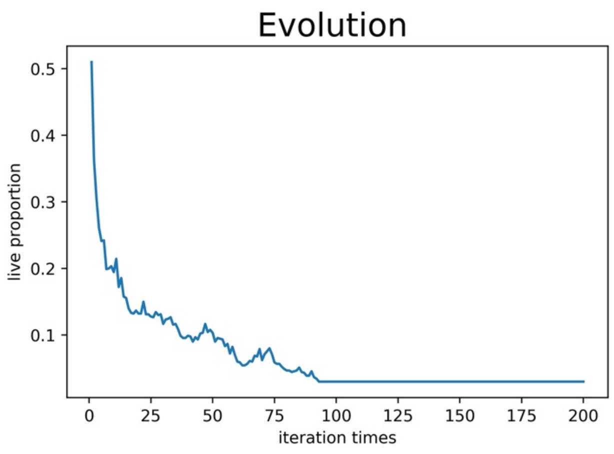

Fig 1: the relationship between live proportion and iteration times

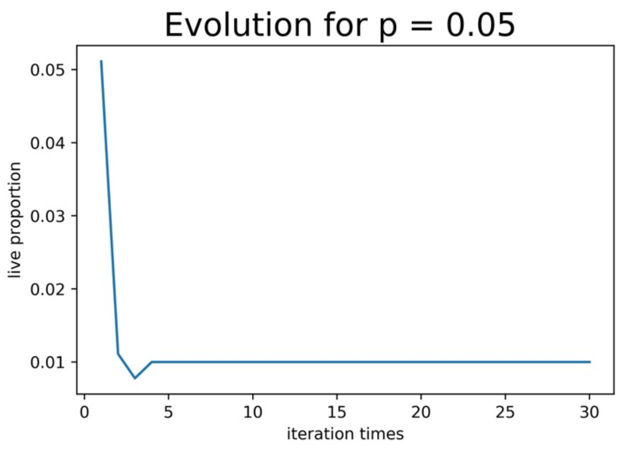

Fig 2: the relationship between live proportion and iteration times for initial live proportion = 0.05

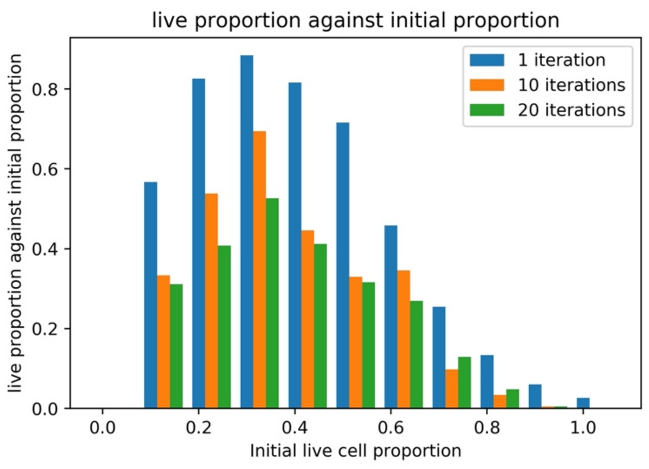

Fig3: the relationship between live proportion against initial proportion and the initial live cells proportion for three different iterations

From the Fig 1, we can see that the live proportion decreases to a steady state as the iteration times increase up to 100, and this trend is identical to the experiment of F Bagnoli et al. [1]. In Fig 2, we can discover that there is a point lower than the steady state when the initial live cells proportion is 0.05. For Fig3, we can see that when the initial proportion in the scale of 20%~40% the live proportion is higher than other situation, and this phenomenon can be explained by the game rules that if 2 or 3 of the 8 neighbors' cells alive, the cell survive. So the live proportion can be around 20% ~ 37.5%.

Performance:

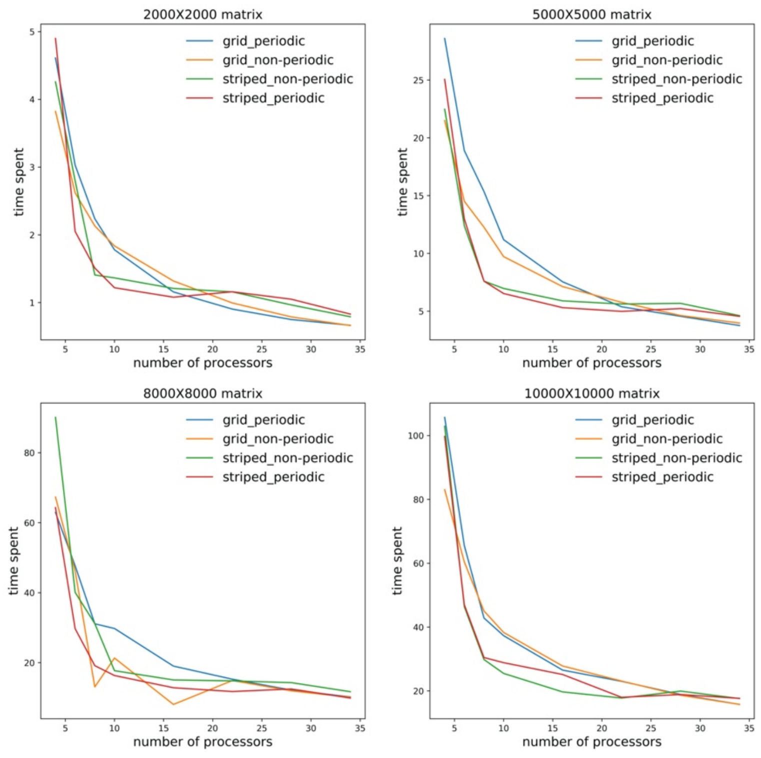

Fig4: the relationship between time spend for 100 iteration times and number of processors are used to calculation for different decomposition method and state(grid

or striped, periodic or non-periodic).

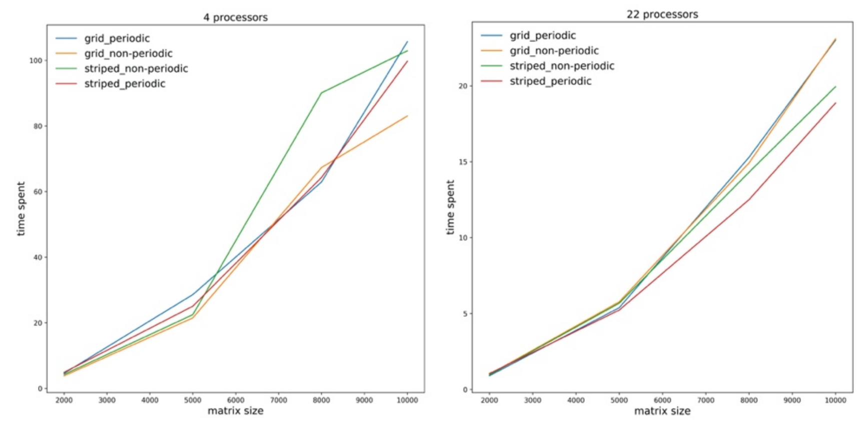

Fig5: for 4 and 22 cores, the relationship between time consumption and matrix size

From Fig4, we can see that the time consumption decreases as the number of processors increase. However, after the number of processors increases to around 10, the time decreases is no longer as obvious as before. As the increase of matrix size, the time consumption increases, and from the Fig5, we can discover there is a linear relationship between matrix size and time consumption. Comparing the ratio $\frac{time\ spent}{matrix\ size}$ (actually the approximate slope)between 4 and 22 cores(0.1, 0.002 separately, while $\frac{22(22\ processors)}{4(4\ processors)} = 5$), we can see that the relationship between cores and time spent is nonlinear(this can also be confirmed by Fig4).

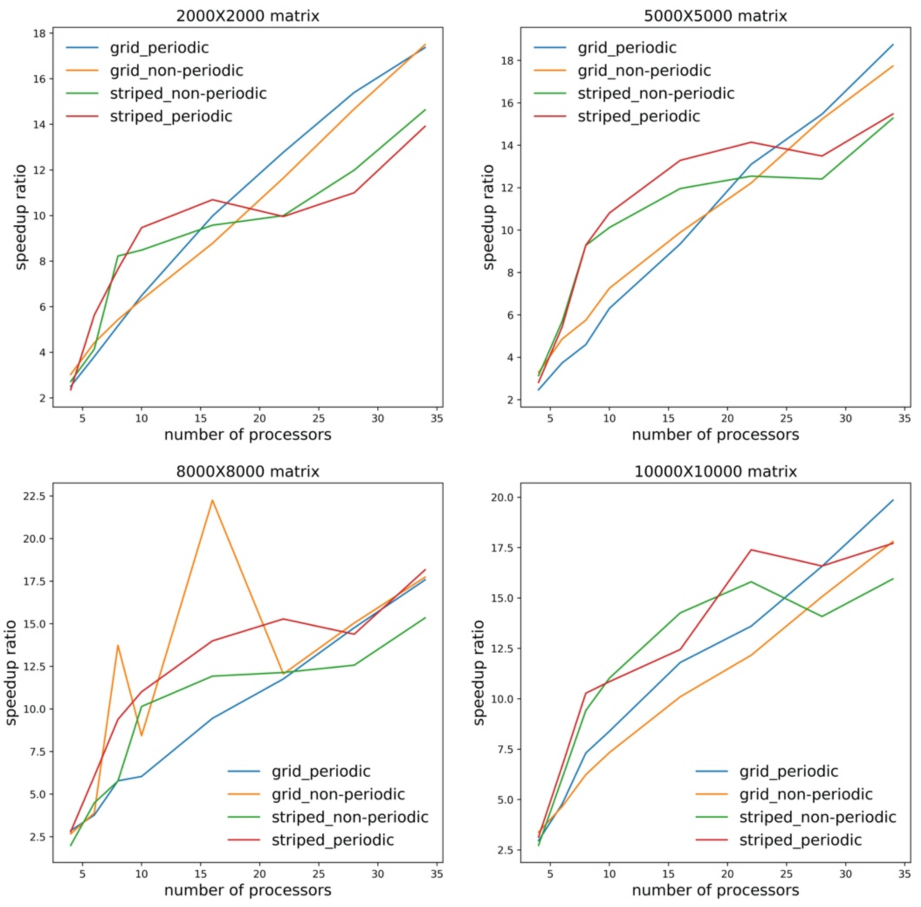

Fig6: the relationship between speedup ratio and number of processors are used to

calculation for different decomposition method and state (grid or striped, periodic or non-periodic).

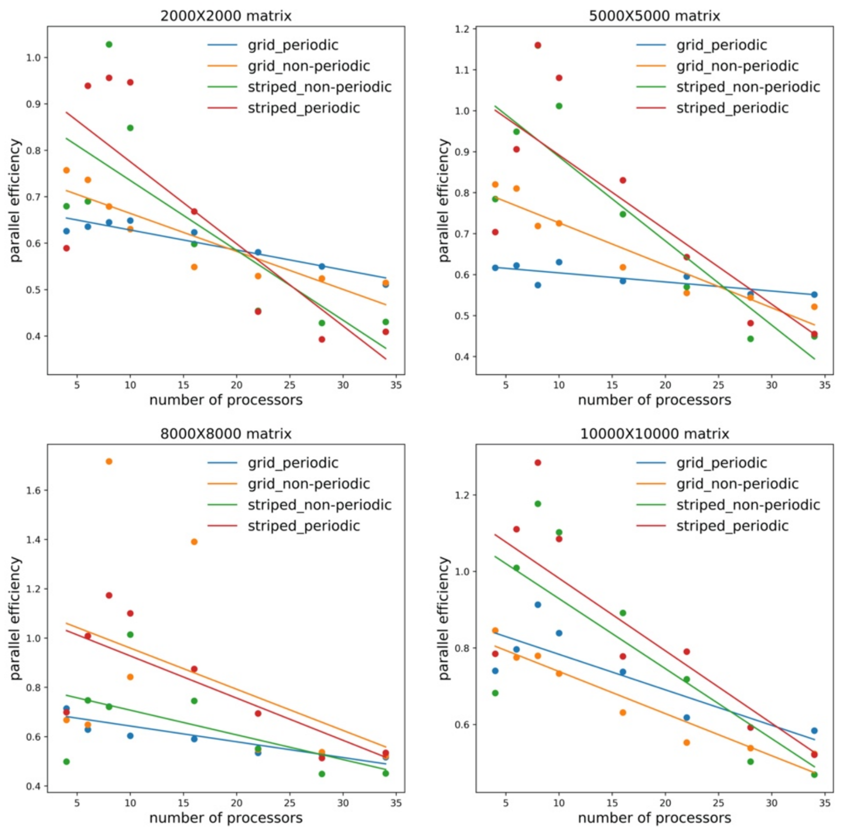

Fig7: the relationship between speedup ratio and number of processors are used to calculation for different decomposition method and state (grid or striped, periodic or non-periodic)

From Fig6 and Fig7, I discover that the relationship between the speedup ratio $S = \frac{T_1}{T_N}$ and number of cores is nearly linear, and the parallel efficiency $E = \frac{S}{N}$ drops as the number cores used increases. From the Fig7 I discovered that striped domain sometimes performs better than grid domain especially when the number of processors is small. This phenomenon can be explained like that grid domain needs more frequency communications among different processors than striped domain, and this may cause the low parallel efficiency when the number of cores is small (smaller than around 30 cores). Another reason why striped domain is more efficiency is that the test data may not large enough for the edge length is large relative to the area of the domain. Additionally, when the number of processors larger than 25, the grid domain speedup ratio is larger than striped domain.

In conclusion, both striped and grid domain can save lots of time. The decomposition strategy in parallelizing game of life is very important.

Reference:

[1] J. B. C. Garcia, M. A. F. Gomes, T. I. Jyh, T. I. Ren, and T. R. M. Sales. Nonlinear dynamics of the cellular-automaton ‘‘game of Life’’. Phys. Rev. E 48, 3345–3351 (1993)

[2]ACSE-6 lectures Power Point