Team1998 Group Project Report

Yang, fan Liu, xiangqi Zhang, chi 2021.02.17

Version Info

Compiler: visual studio 2019, OpenMP: 3.0, Operating sys: Win 10, Num of logical processor: 8

1. Design
   a) Structure

   In general, this project mainly contains two cpp files. One called "ConwaysGame_P_new" is the parallel codes of Conway's game. The second file called "test_correct" is its test file.

   Specifically, the parallelism implementation of Conway's game was similar to the serial version. Both of the updates of the grid were realized by a 3-level nested loop. The outer loop iterated every generation, and the inner 2-level nested loop iterated every row and column to update each cell in the grid. However, in the parallelism version, the traverse will be allocated to different threads concurrently. The main part of parallelization is the iteration function and matrix initialization part in the game of life. It is in the iterative function that we use '*#pragma omp parallel for*' to parallelize the iterative for loop. And in this part, each thread modifies the same array. Therefore, shared is used by threads to share addresses. In the matrix assignment part, because we used a modified one-dimensional array to store the matrix, this part also uses '*#pragma omp parallel for*' for parallel processing. In addition, every matrix assignment operation is assigned to a single thread by '*critical*'. At the same time, parallel processing is performed in the iterative storage and initialization of the array.

   In addition, in order to solve the problem of frequent switching between parallel and serial, we have made some modifications to the program structure. There is no printing in the for loop of the main method. Instead, an '*addlist*' method is created to store the iterations of each step and print them out together at the end of the main method. This greatly improves overall performance. In the output part of the dat file, we will convert one-by-one output to line-by-line output. Convert boolean value to string output. This also improves the overall performance of the program.

   b) I/O

   The program took random Boolean values as its input. The main outputs of the program are images in jpg format that stored in a folder named "img". The image is updated based on the change of the grid's elements. We also outputs dat file to directly outputs boolean values.

   c) Test

   In order to test the correctness of the parallelism implementation, the results obtained from multiple threads are compared to the results obtained from a single thread in "test_correct" cpp. In the program, the initial states of two implementations are same. After the same number of iterations, the consistency of two girds were checked to ensure the correctness. In addition, in order to make the results visible. We set some special types of patterns including still lifes, oscillators, and spaceships. Through the transformation of these patterns, we can detect the correctness of the results from multiple threads as well.

   d) Execution

   First, you should download the openCV and add it to the visual studio. Second, the release mode should be utilized to get a relatively reasonable result. Third, a new folder called "img" should be created to store output images. Then, the values of variables "imax" and "jmax" should be modified to specify the size of grid. Then, modify the value of "max_steps" to specify the total number of generations. Finally, the main function of "ConwaysGame_P_new.cpp" should be run to start the game.

2. Performance

	100x100	1000x1000	10000x10000
serial	0.0433117s	1.64138s	158.957s
parallel	0.0317976s	0.32247s	22.8899s

Table 1. running time of serial codes and parallel codes in different grid size with 100 steps

It can be seen from the running time that the computing performance of Conway's Game of Life has been significantly improved after parallelization. And the ratio of increase in operating speed will increase as the size of the grid increases.

10 steps print inside the game	100x100	1000x1000	10000x10000
serial	0.0629763s	1.28005s	103.817s
parallel	0. 0.0691586s	1.2954s	103.9315s
10 steps print outside the game			
serial	0.145186s	2.25922s	109.388s
parallel	0.0503985s	1.26784s	91.6347s

Table 2. running time of serial codes and parallel codes in different grid size with printing

According to the table above, we can easily find out that the performance of parallel codes was significantly better than the serial codes. As we mentioned earlier, we also optimized printing by collecting grids and printing them out at one time after the game finished. This scheme incurred a slight improvement of the performance as we expected. However, we may also notice that the performances of serial and parallelism was similar when printing images inside each update when the size of the grid less than 1000. This may be caused by time-wasted of those parallel and serial switching that we mentioned earlier.

	16	8	4
100 step 1000x1000	0.342455s	0.609595s	0.931687s

Table 3. running time of parallel codes with different cores

From table 3, the running time of our parallel codes increases as the number of cores decrease.

Max_steps	10	100	1000
parallel	0.143722s	0.944632s	7.94826s
serial	0.235228s	1.82403s	14.835s

Table 4. running time of parallel codes with different max steps with the 1000*1000 grid

Based on table 4, for each max_steps value, the running time of serial code is approximately twice of parallel codes. It showed that the max_steps will not affect the performance.

3. Workload distribution

In this coursework, Mr. Liu was mainly responsible for parallelizing the updating algorithm. Mr. Yang was mainly in charge of image generation, output, and its optimization. Mr. Zhang optimized the data structure of the updating algorithm. All team members were involved in the optimization of printing, debugging, and conduction of tests.

4. Conclusion

In conclusion, this coursework has significantly and optimized the serial codes of Conway's game by both parallelizing the algorithm and improving the data structure. However, the speed of the program can be improved further by optimize the IO output method in the future.

GitHub link: *https://github.com/acse-2020/group-project-team1998*