## Declaration on Plagiarism

*This form must be filled in and completed by the student(s) submitting an assignment*

|  |  |
| --- | --- |
| **Name:** | Xiaofo Geng |
| **Student Number:** | 19210733 |
| **Programme:** | MCM |
| **Module Code:** | CA670 |
| **Assignment Title:** | Concurrent Programming |
| **Submission Date:** | 19th April 2020 |
| **Module Coordinator:** | Dr. David Sinclair |

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. I have read and understood the Assignment Regulations. I have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the source cited are identified in the assignment references. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

I have read and understood the referencing guidelines found at <http://www.dcu.ie/info/regulations/plagiarism.shtml>, <https://www4.dcu.ie/students/az/plagiarism> and/or recommended in the assignment guidelines

Name:\_\_\_\_Xiaofo Geng\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_ April 2020\_\_\_\_\_\_\_\_\_\_\_\_

## Project Description

1. Project overview

Using OpenMP to complete the efficient multiplication of a large matrix. OpenMP provides a fork-and-join execution model in which a program begins execution as a single thread. This thread executes sequentially until a parallelization directive for a parallel region is found (Alpatov et al, 1997; Anderson et al, 1987). In this assignment, the large matrix is partitioned and use multi-threaded parallel computing to improve the efficiency.

1. The Purpose of the Project

Using OpenMP technology to develop multithreaded parallel computing to improve the efficiency. Proving that the efficiency of parallel computing is better than that of serial computing.

1. Design ideas

initialized two matrices A and B such that the number of columns of A and the number of rows of B are equal(Variable M in code). all the values in A and B are random numbers between 0~1. The results are stored in the matrix result. When carried in sequential, Matrices multiplication takes a time O()( M. Khaled et al, 2014 ). In parallel computing, the large matrix is partitioned into many small matrices by four tasks to improve the computing efficiency. Every partition tasks are parallel execution. Master thread forks the outer loop between the slave threads, thus each of these tasks implements matrix multiplication using a half of rows or column from the first matrix. when the threads multiplication are done the master thread will join the total result of matrix multiplication. Finally, output the running time. As a control group, the running time of serial calculation is also output. Mainly functions are following:

* void init\_matrix()：initialize matrix。
* void smallMatrixMult（）：Matrix multiplication
* void matrixMulti（）：Partition Large matrix
* void serial\_matrixMulti()：serial matrix multiplication

## Result

1. Running environment

CPU: i7-8550U(4 core 8 thread) 1.8HGz

RAM: 8GB

Operating System: Window 10

IDE: CodeBlocks

1. Running time

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 500\*500  (time in seconds) | 2000\*2000  (time in seconds) | 6000\*6000  (time in seconds) | 8000\*8000  (time in seconds) | 9000\*9000  (time in seconds) |
| serial | 0.3238 | 43.0722 | 2021.4100 | 7612.0200 | 8548.1700 |
| 2 threads | 0.2457 | 18.3366 | 936.9560 | 1618.1380 | 2126.6730 |
| 4 threads | 0.2036 | 13.9066 | 619.1990 | 1515.8500 | 1439.5010 |
| 8 threads | 0.1338 | 10.5043 | 545.890 | 968.1950 | 1236.130 |

## Conclusion

Based on the results obtained and shown above, the following conclusions can be drawn:

* The running time of multiplication do not have too much difference between serial and parallel for matrices with small sizes.
* OpenMP is a good method to use as a parallel computing multiplication in a large matrix. The efficient of matrix multiplication increases with the number of thread

## Reference

Alpatov, P., G. Baker, C. Edwards, J. Gunnels, and P. Geng (1997). ‘Parallel Matrix Distributions: Parallel Linear Algebra Package’, Tech. Report TR-95-40, Proceedings of the SIAM Parallel Processing Computer Sciences Conference, The University of Texas, Austin.

Anderson E., Z. Bai, C. Bischof, and J. Demmel (1987). ‘Solving Problems on Concurrent Processors’, *Proceedings of Matrix Algorithms Supercomputing '90*, IEEE 1, pp. 1-10.

M. Khaled, A. Alhasanat, H. Alashaary, Z. Alqadi (2014). ‘Analysis of Matrix Multiplication Computational Methods’, *European Journal of Scientific Research*, 121(3), pp. 258-266.