**CARDIFF SCHOOL OF TECHNOLOGIES: ASSIGNMENT FEEDBACK PROFORMA**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| STUDENT NAME: Hui Yin Ting | | | | | PROGRAMME: Computing | | | |
| STUDENT NUMBER: St20104639 | | | | | YEAR: 2018-19 | | GROUP: Comp & IS | |
| Module Number: CIS6007 | Term: 1 | | | | Module Title: Parallel and Distributed Systems | | | |
| Tutor Responsible For Marking This Assignment: Paul Angel | | | | | | | | |
| Module Leader: Paul Angel | | | | | | | | |
| Assignment Due Date: 30th November, 2018 | | | Hand In Date: 30th November, 2018 | | | | | |
| ASSIGNMENT TITLE: Creating the Julia Set using Parallel Programming Techniques | | | | | | | | |
| **SECTION A: SELF ASSESSMENT (TO BE COMPLETED BY THE STUDENT)** | | | | | | | | |
| **In relation to each of the set assessment criteria, please identify the areas in which you feel you have strengths and those in which you need to improve. Provide evidence to support your self-assessment with reference to the content of your assignment.** | | | | | | | | |
| STRENGTHS  Working sequential and Parallel code  Decent testing choices | | | | AREAS FOR IMPROVEMENT  NO OPENCL  NO testing with OPENCL | | | | |
| I certify that this assignment is a result of my own work and that all sources have been acknowledged:  Signed:\_Hui Yin Ting\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date\_\_08/12/18\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | | | | | |
| SECTION B: TUTOR FEEDBACK **(based on assignment criteria, key skills and where appropriate, reference to professional standards)** | | | | | | | | |
| STRENGTHS | | | | AREAS FOR IMPROVEMENT AND TARGETS FOR FUTURE ASSIGNMENTS | | | | |
| MARK/GRADE AWARDED | | DATE: | | | | SIGNED | | |
| ASSIGNMENT MODERATED BY: | | | | | | | | DATE |
| MODERATOR’S COMMENTS: | | | | | | | | |

Parallel Report Documentation

Hui Yin Ting

St20104639

Table of Content

Test Plan ………………………………………………………………………………………………………... Page 5

Test Result ……………………………………………………………………………………………………… Page 5

Test Evaluation ………………………………………………………………………………………………. Page 7

Evaluation Result …………………………………………………………………………………………… Page 8

Extra Testing ………………………………………………………………………………………………….. Page 8

Bibliography …………………………………………………………………………………………………… Page 9

Test Plan

For the Julia Set Code that I have created, it creates a Julia set image with mostly black and white. The variables that could be altered in this code that I created are: ITERATIONS, width and height. However, from a coding error even though the code works the width and height is considered not to be changed to variables that are different from each other. As in my code the width and height may be stated as two different variables but, in the loops, if they are put into different variable from each other a proper Julia set image may not be created in the process. For this testing I will be changing the variables of Iteration, width and height. From the changing of these variables to see how they affect the processing speed of the code and time differences between the different ways of processing. The codes will be the normal one which uses sequential and the other being parallel processing which will be using the threading building block for it. Both codes will be using freeimage for the image building. Another processing method was the use of GPU however, I was not able to get the code working properly so for this test I will only be looking into the sequential and parallel code that are functioning.

Test Result

Test 1

|  |  |  |  |
| --- | --- | --- | --- |
|  | Julia Set | Julia Set Parallel | Time Difference |
| Width | 1000 | 1000 |  |
| Height | 1000 | 1000 |  |
| Iterations | 500 | 500 |  |
| Time Taken | 2.14 | 0.59 | 1.55 |

From the chart above we can see that with the width and height of 1000 and 500 iteration sequential Julia Set takes 2.14 seconds to complete the task while parallel takes 0.59 seconds to complete. The time difference is 1.55

Test 2

|  |  |  |  |
| --- | --- | --- | --- |
|  | Julia Set | Julia Set Parallel | Time Difference |
| Width | 2000 | 2000 |  |
| Height | 2000 | 2000 |  |
| Iterations | 1000 | 1000 |  |
| Time Taken | 8.23 | 2.17 | 6.06 |

The second test I increased all the changeable variables and doubled all of them. For Sequential it took 8.23 seconds to complete while parallel took 2.17. The time difference is 6.06 seconds.

Test 3

|  |  |  |  |
| --- | --- | --- | --- |
|  | Julia Set | Julia Set Parallel | Time Difference |
| Width | 5000 | 5000 |  |
| Height | 5000 | 5000 |  |
| Iterations | 5000 | 5000 |  |
| Time Taken | 50.67 | 12.03 | 38.64 |

For the next test I tried changing all the variables into huge numbers. So, I changed all the variables to 5000. It took sequential 50.67 seconds to complete the task and Parallel 12.03 seconds to complete the task. Time difference is 38.64 seconds.

Test 4

|  |  |  |  |
| --- | --- | --- | --- |
|  | Julia Set | Julia Set Parallel | Time Difference |
| Width | 2000 | 2000 |  |
| Height | 2000 | 2000 |  |
| Iterations | 5000 | 5000 |  |
| Time Taken | 8.43 | 2.27 | 6.16 |

For the next test I went back to 2000 for both width and height but kept iterations at 5000. For sequential it took 8.43 seconds to complete the task and 2.27 seconds for parallel. The time difference is 6.16.

Test 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Julia Set | Julia Set Parallel | Time Difference | OpenCL |
| Width | 5000 | 5000 |  | 5000 |
| Height | 5000 | 5000 |  | 5000 |
| Iterations | 1000 | 1000 |  | 1000 |
| Time Taken | 50.6 | 11.95 | 38.65 | 60.3 |

For this test I went the opposite to the previous test, I changed width and height to 5000 and iteration back down to 1000. The time taken is 50.6 seconds for sequential and 11.95 seconds for parallel. I have a OpenCL code in my project folder that can generate a Julia set. Although it does not function properly as it should be much faster than the result I got here with the same variables but the code I got here has took 60.3 seconds to complete task and has generated a Julia set image with some different colour displayed.

Test Evaluation

From all the testing conducted it is obvious to see that the larger the variable the longer it will take for all the code to complete their task. At the same time the time gap taken between sequential and parallel increases as the variable increases. From test 1 the time gap between sequential and parallel is 1.55 seconds, while in test 3 where the variable has been substantially increased the time difference is 38.64. From this we can see that sequential and parallel time difference may not be very obvious when processing small amount of data but when it comes to large amount parallel can complete the test much fast then sequential can.

I conducted test 4 and test 5 to see how much the variable iteration can affect the actual time it takes to complete the task. For test 4 I kept iteration at 5000 while returning width and height variables back to test 2. The result shows that iteration may not have a very big impact on the time it took to complete the task. The result from test 4 are 8.43 and 2.27. Test 2 has the result of 8.23 and 2.17. With only 0.2 and 0.1 second difference. For test 5 I did the opposite and changed the iteration variable down while maintaining the same variables for width and height from test 3. It took 50.6 and 11.95 seconds to complete their task in test 5. While in test 3 they took 50.67 and 12.03 to complete their task. There were no significant changes as the time difference is only 0.06 and 0.08. The result also show that iteration has no big impact on the time it takes to complete the task.

From doing multiple testing on my code with the exact variable I was able to get different time taken on them, there are many factors that can affect the speed it took for the code to complete their task. Since in testing for test 4 I did its multiple times to since I had one time that the sequential result was the exact same time it took as test 2 however, when I run the code again it gave me the result that I got right now. So, the results I have gotten from my testing may not be completely accurate and can be biased.

Evaluation Results

From the testing we can see that sequential is not very effective way to process large amount of data compared to parallel. However, if the data is not that big then there is not a very big difference between parallel and sequential as a 1.55 seconds difference is not that big of a deal. The variables that effects the processing the most from my code will be width and height, while iteration will only affect a little bit. If OpenCL was function properly I could get more interesting results.

Extra Test

Another interesting variable I tested around is: “outputBuffer [(y \* width + x) \* 4] = colors [i % 5];”. This part of the code is used to input the Julia set image into the canvas of the “outputBuffer” with colors. Originally for this part of the code I was considering using GetPixelColor, However I was not able to make that work so I went with another way to get the Julia Set image on the canvas. The formula for this code can be altered at the (y\* width + x) \* 4 area. By changing the number of 4 and adding addition to the function it can change the way the picture is presented on the image. Form doing many testing multiplications does not work with decimals and will not go any higher then 4, if the equation was altered to \* 5 the code will not run and give an error. If the number was changed to 3 the image that will be displayed changes from 1 to 4 multiple same images. If then add 9 to the equation the 4 images are then smeared together to make something completely different.

Bibliography

YouTube. (2018). *Coding Challenge #22: Julia Set in Processing*. [online] Available at: https://www.youtube.com/watch?v=fAsaSkmbF5s&t=192s [Accessed 2 Dec. 2018].

YouTube. (2018). *Coding the Mandelbrot - C++ Tutorial*. [online] Available at: https://www.youtube.com/watch?v=entjSp3LIfQ&t=1s [Accessed 2 Dec. 2018].

Joubert, N. (2008). *Using FreeImage 3.11 for Image Output*. [ebook] UC Berkeley: CS184 TA, pp.1-5. Available at: <http://graphics.stanford.edu/courses/cs148-10-summer/docs/UsingFreeImage.pdf> [Accessed 3 Dec. 2018].

En.wikipedia.org. (2018). *Threading Building Blocks*. [online] Available at: https://en.wikipedia.org/wiki/Threading\_Building\_Blocks [Accessed 4 Dec. 2018].

C++, G. (2018). *Getting pixel color in C++*. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/4839623/getting-pixel-color-in-c [Accessed 5 Dec. 2018].