COMP409 Winter2020 A1 Writeup

Eric Shen
260798146

I set my image width as 1980, image height as 1080.
I let number of the rectangles to draw is 100.

For 1 thread:

1st trial: 2126 milliseconds.
2nd trial: 1982 milliseconds.
3rd trial: 1867 milliseconds.
4th trial: 1876milliseconds.
5th trial: 1701 milliseconds.
Average time is: 1910.4 milliseconds.

For 2 thread:

1st trial: 484 milliseconds.
2nd trial: 1034 milliseconds.
3rd trial: 918 milliseconds.
4th trial: 1031 milliseconds.
5th trial: 837 milliseconds.
Average time is: 860.8 milliseconds.

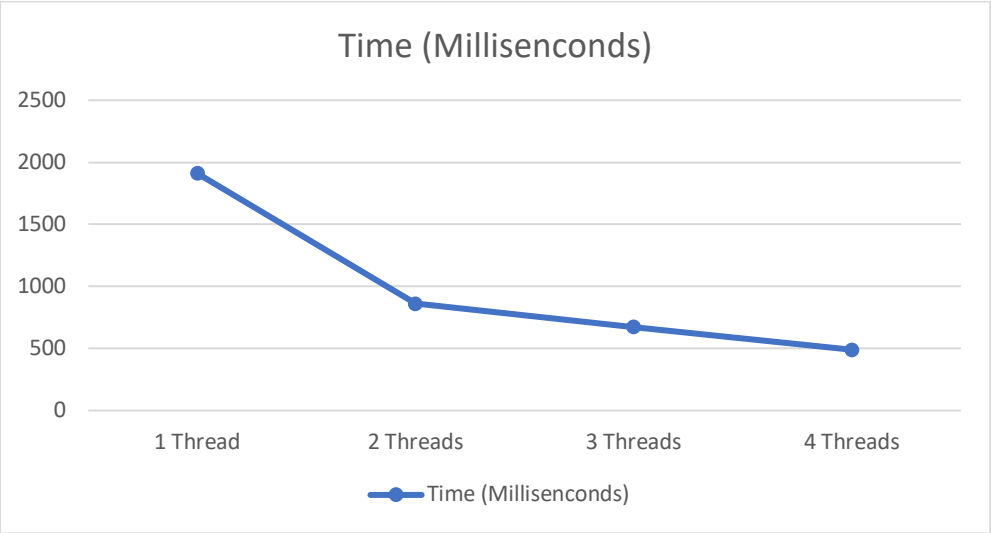
For 3 thread:

1st trial: 684 milliseconds.
2nd trial: 705 milliseconds.
3rd trial: 723 milliseconds.
4th trial: 748 milliseconds.
5th trial: 502milliseconds.
Average time is: 672.4 milliseconds.

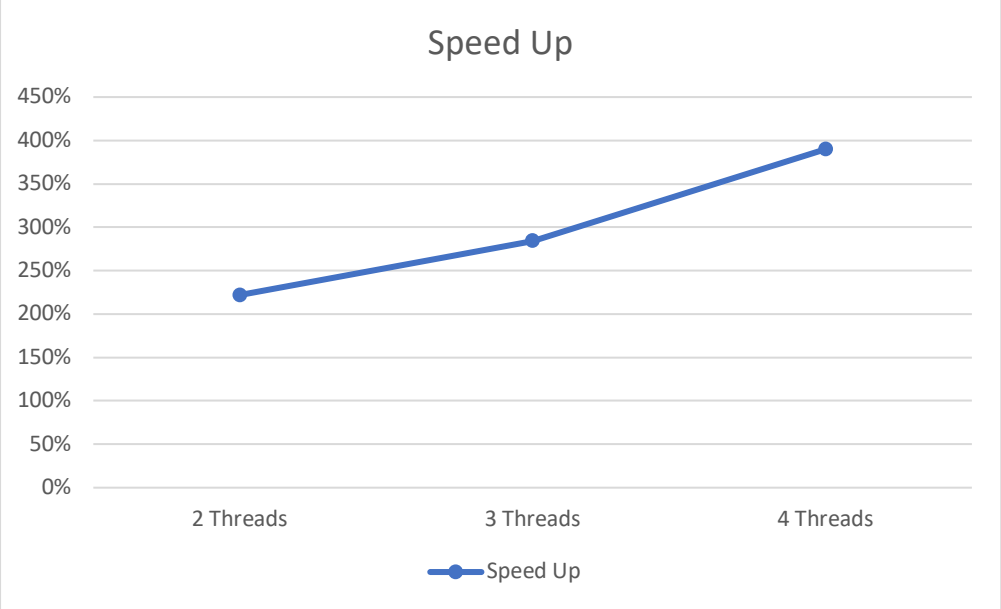
For 4 thread:

1st trial: 509 milliseconds.
2nd trial: 673 milliseconds.
3rd trial: 365 milliseconds.
4th trial: 498 milliseconds.
5th trial: 405 milliseconds.
Average time is: 490 milliseconds.

Time plot:



Performance speedup compared to single thread plot:



Thoughts:
From the data, the plot is not exactly linear but still close. I think that because when using multiple threads to draw rectangles, rectangles can be drawn concurrently. 2 threads should be twice fast as single thread, 3 threads should be third times fast as single thread, and 4 threads should be four times fast than single thread. However, due to real other elements like computer hardware or random method to get another rectangle can be overlapped by other ones thus needs to re-choose. When the number of rectangle increase, the performance should be more stable.