THE MANDLEBROT FUNCTION

Jack Sime

The Purpose & Problem

- The purpose of the application is too produce a display of a Mandelbrot set and allow the user to select both the colour choice and level of zoom that is used in the actual display.
- Parallel programming aims to solve the problem of the time actually being taken by the system to run the application and produce the image of the Mandelbrot set.
- Through the usage of threads the aim was to reduce the time taken by the system to produce these images.

Parameters of testing

- When testing the impacts of threads each thread number was tested 25 times.
- The median value was used as the timing result as it provides the most true idea of timing.
- The selection menus were bypassed during testing as the selection has no impact of the actual creation and only relates to the display and introduced an uncontrollable variable of time spent in the menu.
- Output to the console such as the progress was enabled during testing as it's a true part of the application and does impact timings.
- All timings were recorded through code and were exported to a csv file to be as accurate as possible.
- Background performance within the system was limited while testing to reduce external impacts and all testing was performed within a single session.

Structure

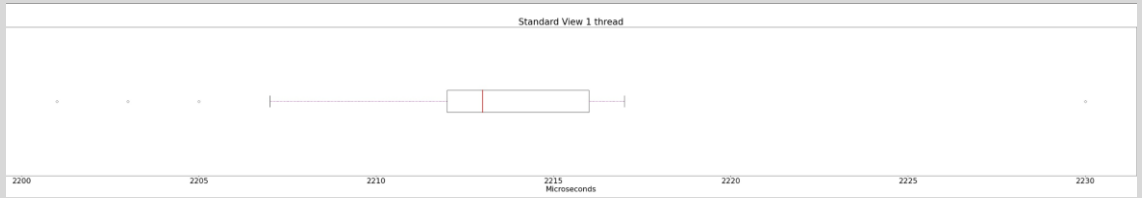
- The application follows a fork join pattern where the task is separated up into section and then joined once every section is complete.
- The structure allows the Mandelbrot set image to be split into a number of sections and then each thread is assigned a section to create.
- After all the threads have completed their section, the sections are joined again and this is repeated for the zoomed image as well.
- Condition variables and mutexes are used to allow a progress statement to be shown to the user, after a thread has completed its section the usage of the mutex and condition variable allows it to only display that its finished when it has truly completed its section.

Structure

- The interactions between threads are managed using the condition variables and the mutexes as it prevents more than a single thread being able to signal its complete at the same time, this prevents any errors that could occur.
- A second thread function is used to handle the displaying of the images to the user and manages the users choice for zooming in and out and for exiting the program.
- This display function runs separate from the initial threads and only runs once both sets of threads have completed their tasks e.g. compiling the Mandelbrot set image

Single thread

Standard



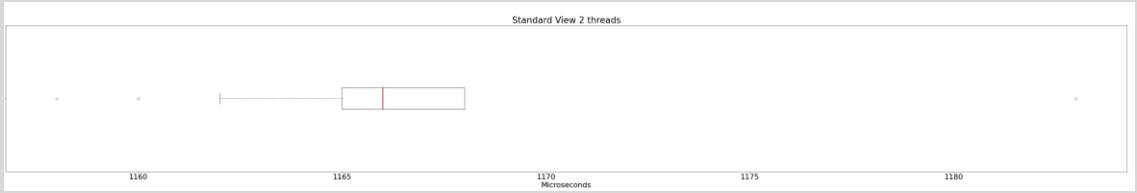
- Minimum- 2201
- Maximum- 2230
- Q1-2212
- Q2(Median)-2213
- Q3-2216
- IQR-4
- Range-29

Zoomed

- Minimum- 29064
- Maximum- 29167
- Q1-29086.5
- Q2(Median)-29100
- Q3-29114
- IQR-27.5
- Range-103

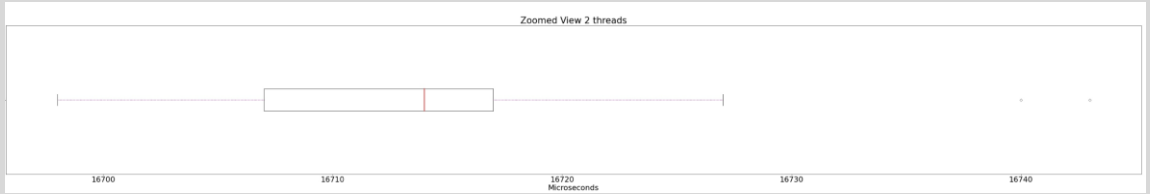
Two threads

Standard



- Minimum- 1158
- Maximum- 1183
- Q1-1164.5
- Q2(Median)-1166
- Q3-1168
- IQR-3.5
- Range-25

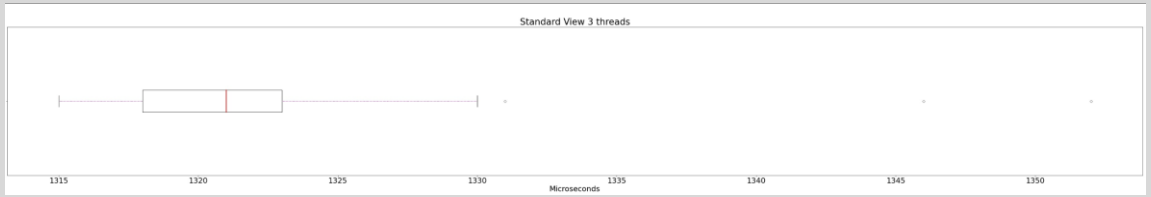
Zoomed



- Minimum- 16698
- Maximum- 16743
- Q1-16707
- Q2(Median)-16714
- Q3-16718.5
- IQR-11.5
- Range-45

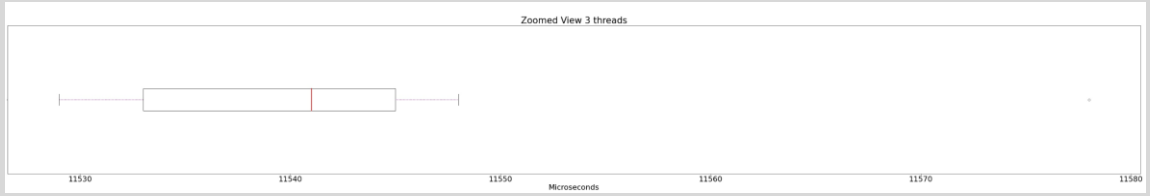
Three threads

Standard



- Minimum- 1315
- Maximum- 1352
- Q1-1318
- Q2(Median)-1321
- Q3-1323.5
- IQR-5.5
- Range-37

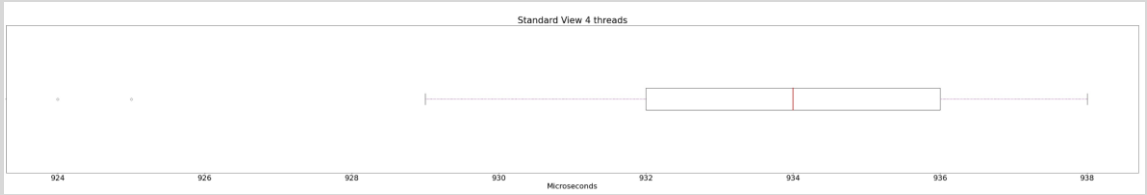
Zoomed



- Minimum- 11529
- Maximum- 11578
- Q1-11532.5
- Q2(Median)-11541
- Q3-11546
- IQR-13.5
- Range-49

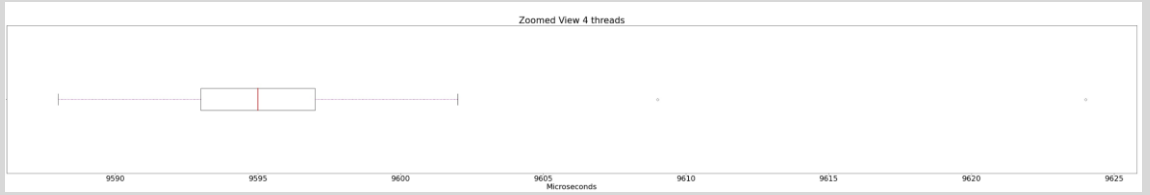
Four threads

Standard



- Minimum- 924
- Maximum- 938
- Q1-931.5
- Q2(Median)-934
- Q3-936
- IQR-4.5
- Range-14

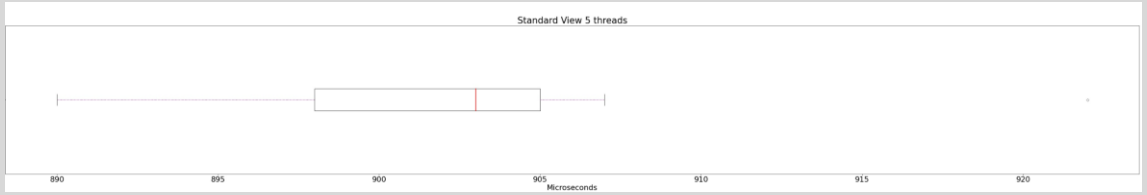
Zoomed



- Minimum- 9588
- Maximum- 9624
- Q1-9593
- Q2(Median)-9595
- Q3-9597
- IQR-4
- Range-36

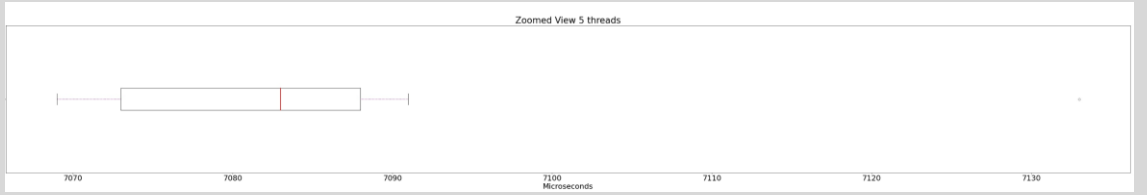
Five threads

Standard



- Minimum- 890
- Maximum- 922
- Q1-898
- Q2(Median)-903
- Q3-905.5
- IQR-7.5
- Range-32

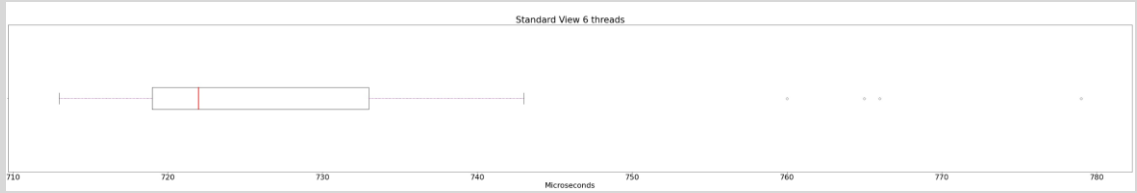
Zoomed



- Minimum- 7069
- Maximum- 7133
- Q1-7073
- Q2(Median)-7083
- Q3-7088
- IQR-15
- Range-64

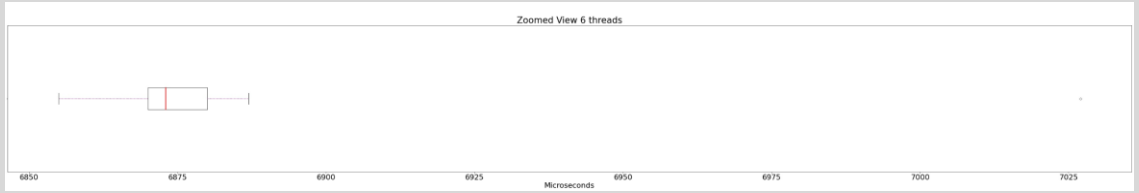
Six threads

Standard



- Minimum- 713
- Maximum- 779
- Q1-718
- Q2(Median)-722
- Q3-733.5
- IQR-15.5
- Range-66

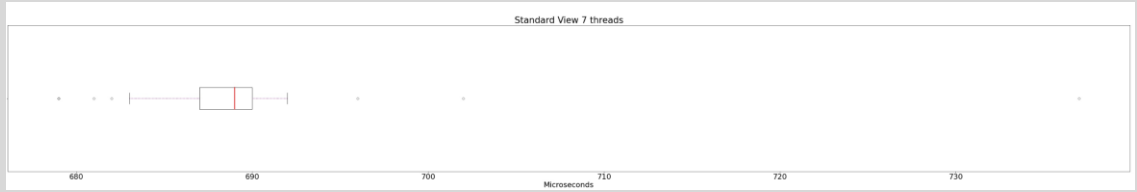
Zoomed



- Minimum- 6855
- Maximum- 7027
- Q1-6869.5
- Q2(Median)-6873
- Q3-6882.5
- IQR-13
- Range-172

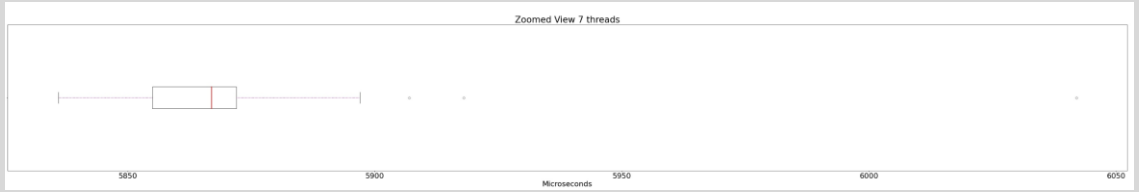
Seven threads

Standard



- Minimum- 679
- Maximum- 737
- Q1-686
- Q2(Median)-689
- Q3-690.5
- IQR-4.5
- Range-58

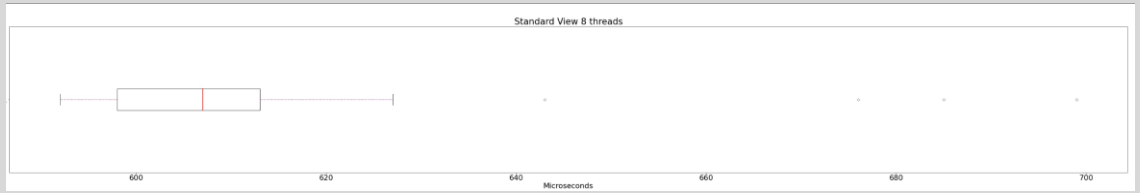
Zoomed



- Minimum- 5836
- Maximum- 6042
- Q1-5853.5
- Q2(Median)-5867
- Q3-5874
- IQR-20.5
- Range-206

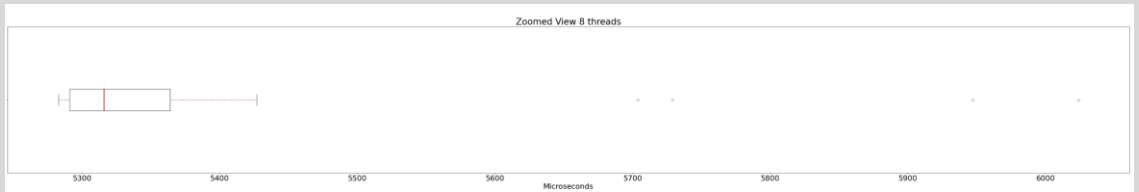
Eight threads

Standard



- Minimum- 592
- Maximum- 699
- Q1-597
- Q2(Median)-607
- Q3-614
- IQR-17
- Range-107

Zoomed



- Minimum- 5283
- Maximum- 6042
- Q1-5289
- Q2(Median)-5316
- Q3-5379
- IQR-90
- Range-741

Comparison of Standard View

	1 Thread	2 Threads	3 Threads	4 Threads	5 Threads	6 Threads	7 Threads	8 Threads
Minimum	2201	1158	1315	924	890	713	679	592
Maximum	2230	1183	1352	938	922	779	737	699
Q1	2212	1164.5	1318	931.5	898	718	686	597
Q2(Median)	2213	1166	1321	934	903	722	689	607
Q3	2216	1168	1323.5	936	905.5	733.5	690.5	614
IQR	4	3.5	5.5	4.5	7.5	15.5	4.5	17
Range	29	25	37	14	32	66	58	107

Comparison of Zoomed View

	1 Thread	2 Threads	3 Threads	4 Threads	5 Threads	6 Threads	7 Threads	8 Threads
Minimum	29064	16698	11529	9588	7069	6855	5836	5283
Maximum	29167	16743	11578	9624	7133	7027	6042	6042
Q1	29086.5	16707	11532.5	9593	7073	6869.5	5853.5	5289
Q2(Median)	29100	16714	11541	9595	7083	6873	5867	5316
Q3	29114	16718.5	11546	9597	7088	6882.5	5874	5379
IQR	27.5	11.5	13.5	4	15	13	20.5	30
Range	103	45	49	36	64	172	206	741

Results

- From testing its found that there was a 72.57% decrease in the time taken from 1 thread to 8 threads for the standard view.
- There was an increase of time taken when using 3 threads instead of 2 on the standard view, may be due to task allocation/quantity.
- From testing its found that there was also an 81.73% decrease in the time taken from 1 thread to 8 threads for the zoomed view.
- There was no increase in time from 3 threads when compared to 2 threads.
- Overall there is a clear performance improvement but could have been improved further with the usage of a different pattern in the application.
- Further interactivity would improve the functionality and would benefit the user significantly.
- A range spike was apparent in the zoomed testing of 8 threads and was not seen anywhere else in the data, this suggests its related to external factors.

Conclusion

- Major increases in performance with addition of more threads.
- Application competition time could still be improved through a different pattern that can better allocate work loads.
- External factors are inevitable in a testing environment like this, but were prevented as much as possible.
- The usage of mutexes and condition variables adds time to the process but ensures a true and reliable performance from it.
- Race conditions are avoided thanks to the usage of mutexes and the condition variables and allows the progress outputs to properly increment.

Thank You for Listening

Any Questions?