Due: October 17th, 2021

Braelyn Rotman
1006740

Assignment 1 Report – Data and Task Parallelism Using Pthreads
Braelyn Rotman - 1006740

Raw Data

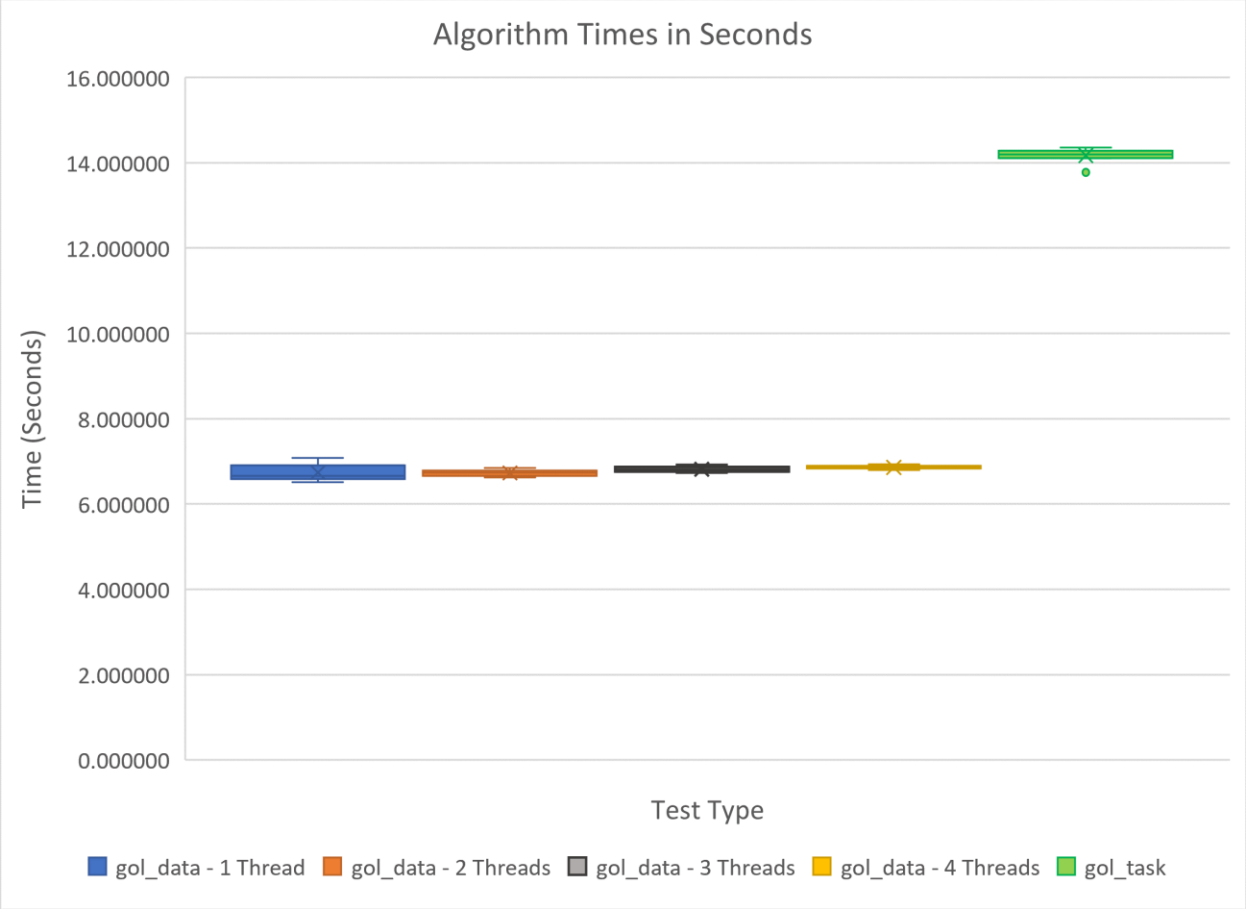
Algorithm: gol_data

Threads	RUN 1	RUN 2	RUN 3	RUN 4	RUN 5	RUN 6	RUN 7	RUN 8	RUN 9	RUN 10
1	6.506710	6.623932	6.532672	6.633690	6.688895	6.968935	6.890827	6.865845	7.074208	6.609400
2	6.739063	6.621268	6.843019	6.737664	6.782770	6.739989	6.787502	6.736747	6.625220	6.671135
3	6.885695	6.864745	6.922000	6.806835	6.761238	6.736344	6.817269	6.739717	6.765942	6.864656
4	6.890055	6.930432	6.795135	6.859955	6.853151	6.881297	6.833457	6.836223	6.820510	6.887870

Algorithm: gol_task

RUN 1	RUN 2	RUN 3	RUN 4	RUN 5	RUN 6	RUN 7	RUN 8	RUN 9	RUN 10
13.772186	14.107539	14.353595	14.113728	14.203556	14.184849	14.205971	14.264942	14.166591	14.315273

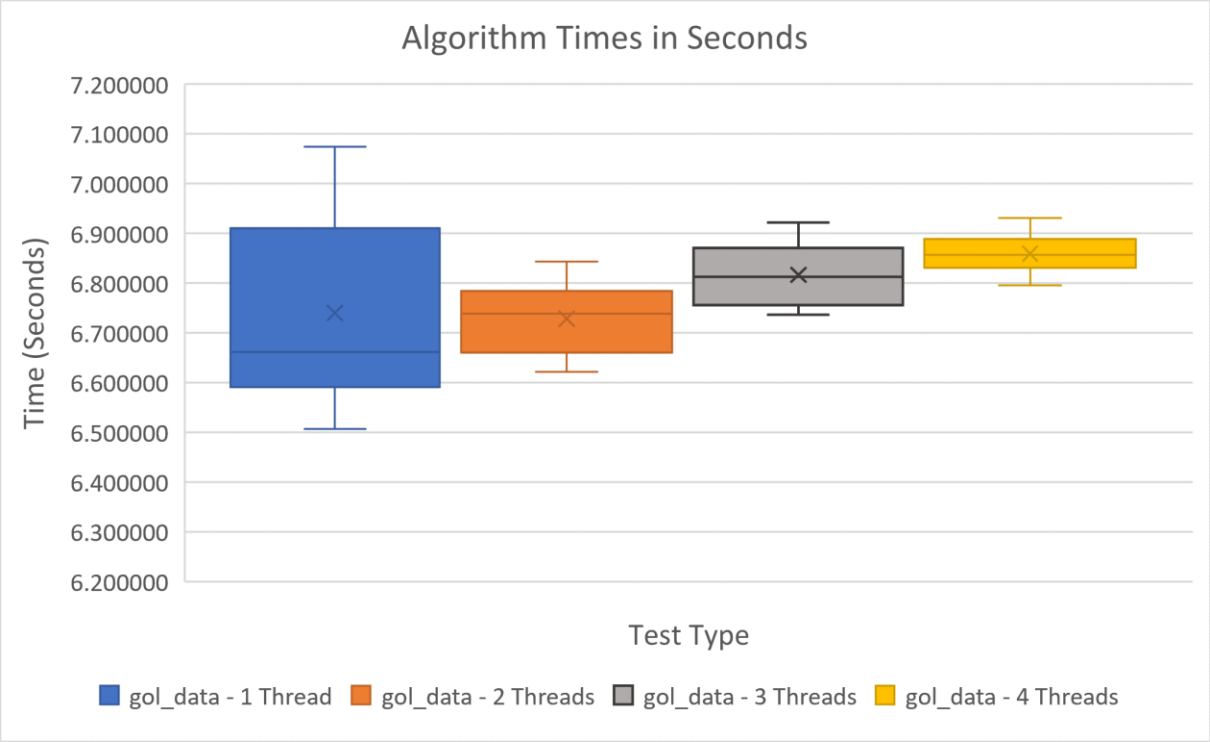
Box and Whisker Plot



Due: October 17th, 2021

Braelyn Rotman
1006740

Additional Box and Whisker Plot (Only gol_data Algorithm)



Analysis and Thoughts

Looking at the gol_data plots alone I was surprised to see an overall small increase in runtime as the number of threads increased. I thought having multiple threads would be faster as the work is divided up. My guess of what happened would be that my computer doesn't handle threads very well and throwing around the cpu takes longer than if it were a serial program. This wouldn't be surprising as my computer often struggles to run multiple applications at the same time, not overly powerful. These tests were also done on a virtual machine which could have affected runtimes.

It's difficult to compare gol_data and gol_task. The task parallel algorithm took roughly twice the amount of time as the data parallel solution however after looking through my code I believe this is due to the use of queues in gol_task.c. The queue functions take up time and act as a middleman in this scenario, not exactly a fair comparison. Aside from the queues, I thought gol_task would be more efficient than gol_data. Since all the threads in gol_data are completing the same job there is a lot more waiting for access to shared data where as in gol_task one thread determines the placement of all the cells while the other two threads fill in the grid simultaneously. The queues are shared between the queue filler and readers but there can only ever be 2 threads trying to access data rather than the multiple threads in gol_data.