MULTITHREADING THE CALCULATION OF THE AVERAGE ANGULAR DISTANCE BETWEEN 50,000 STARS IN THE TYCHO STAR CATALOGUE

Prepared for
Trevor Bakker, Adjunct Professor
Operating Systems
The University of Texas at Arlington

Prepared by
Ethyn Nguyen, Student
Software Engineering Major
The University of Texas at Arlington

March 29, 2023

EXECUTIVE SUMMARY

This report analyzes the use of multithreading to calculate the average angular distance between 50,000 stars in the Tycho star catalogue. The goal is to increase throughput and the real-time it takes for the main program to execute and finish.

Important Libraries used and Reasons for Chosen Timing Method

For this assignment, <pthread.h>, <sys/time.h>, and <ctype.h> were added to the existing main file. I used library <pthread.h> to implement threading in the program. With <sys/time.h>, I implemented a timer to time how long it took for the program to calculate the average angular distance between the stars in the Tycho Star Catalogue. <ctype.h> was used for its isdigit() function. The timeval struct gave a start and end time. This timing method was chosen because it could provide real-time with precise accuracy.

Results of Multithreading

Multithreading the program resulted in a decrease in the amount of real-time the program needed to execute. There were rounding float errors when introducing more threads, but they are negligible, and the issue was resolved. The optimal thread count was 25. Higher thread counts resulted in slower performance.

MULTITHREADING THE CALCULATION OF THE AVERAGE ANGULAR DISTANCE BETWEEN 50,000 STARS IN THE TYCHO STAR CATALOGUE

IMPORTANT LIBRARIES USED AND REASONS FOR CHOSEN TIMING METHOD

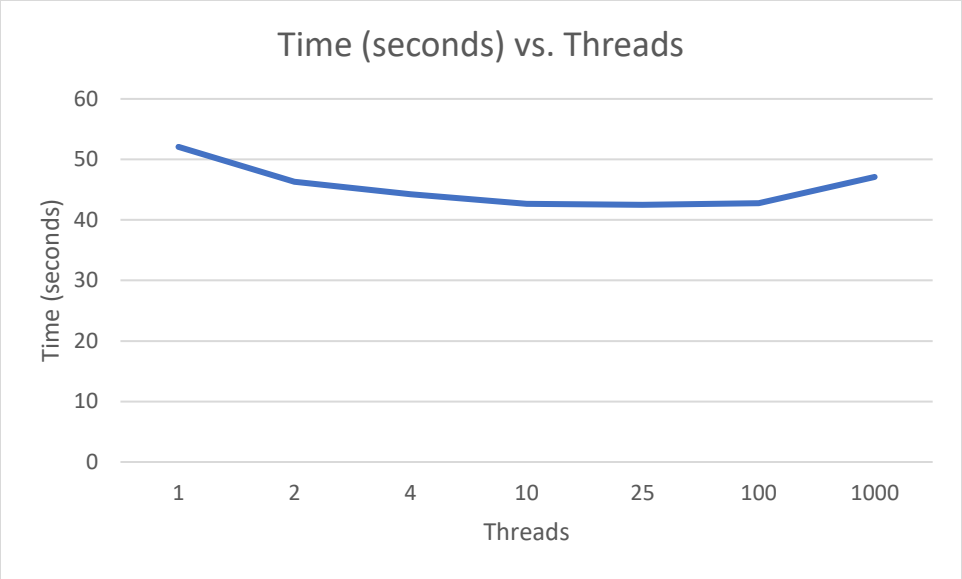
For this assignment, <pthread.h>, <sys/time.h>, and <ctype.h> were added to the existing main file. I used library <pthread.h> to implement threading in the program. Each thread would take an evenly distributed portion of stars in the star_array and calculate the distance between two stars. A mutex is needed to protect global variables and prevent race conditions when working with threads. Mean and count were two variables that required a mutex, as the calculation for the mean required both variables. Max and min, while global variables, did not exhibit race conditions and thus didn't need a mutex. With <sys/time.h>, I implemented a timer to time how long it took for the program to calculate the average angular distance between the stars in the Tycho Star Catalogue. Using the timval struct allowed for more real-time precision in determining how long the threads took to finish executing. <ctype.h> was used for its isdigit() function. The isdigit() function determines whether the -t argument was valid.

RESULT OF THREADS

Time (seconds) vs. Threads

Test	1 Thread	2 Threads	4 Threads	10 Threads	25 Threads	100 Threads	1000 Threads
1	51.987	46.261	44.37	42.242	42.37	43.031	46.736
2	52.053	46.301	43.967	42.652	42.307	42.603	47.445
3	52.488	46.23	44.473	42.56	42.933	42.688	47.227
4	51.943	46.509	44.289	43.005	42.494	42.727	47.096
5	52.021	46.328	44.413	42.442	42.815	43.117	46.799
6	52.078	46.544	44.38	42.593	42.257	42.745	47.347
7	51.93	46.461	44.249	42.341	42.409	42.893	46.959
8	52.448	46.181	43.96	42.968	42.294	42.672	46.693
9	51.832	46.232	44.204	42.666	42.635	42.49	46.979
10	51.936	46.088	44.161	42.944	42.447	42.735	47.723
Avg	52.072	46.314	44.247	42.641	42.496	42.77	47.1

Average distance found is 31.904231
Minimum distance found is 0.000225
Maximum distance found is 179.569720



ANOMALIES FOUND

One anomaly found was the resulting average distance. For lower thread counts, the average distance found was 31.904231. When using higher thread counts, the mean found sometimes was 31.904232. There is a difference of 0.000001 between the lower and higher thread counts. The most likely reason for this difference is the rounding of the float calculations. However, one-millionth of a difference in distance is negligible and can be overlooked. However, by removing a redundant check, I fixed this anomaly. Another anomaly found was that when using 1000 threads, the program executed slower than when using 100 threads. While one may assume that higher threads lead to faster runtime time, having too many threads could result in slower execution. Having more threads than your CPU supports results in serializing and not parallelizing the program.

CONCLUSIONS

The optimal amount of threads tested was 25 threads. Using more threads resulted in an execution time slower than 25 threads. When using 25 threads, the program took, on average, 42.496 seconds to execute, which was about 10 seconds faster than using one thread. Having more threads increased bottlenecks as there needed to be more CPU cores to handle the threads. Having fewer threads resulted in underutilized CPU cores. Conclusively, 25 threads are the optimal number to calculate the average distance between stars.