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 26, 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. The optimal thread count was 100. 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

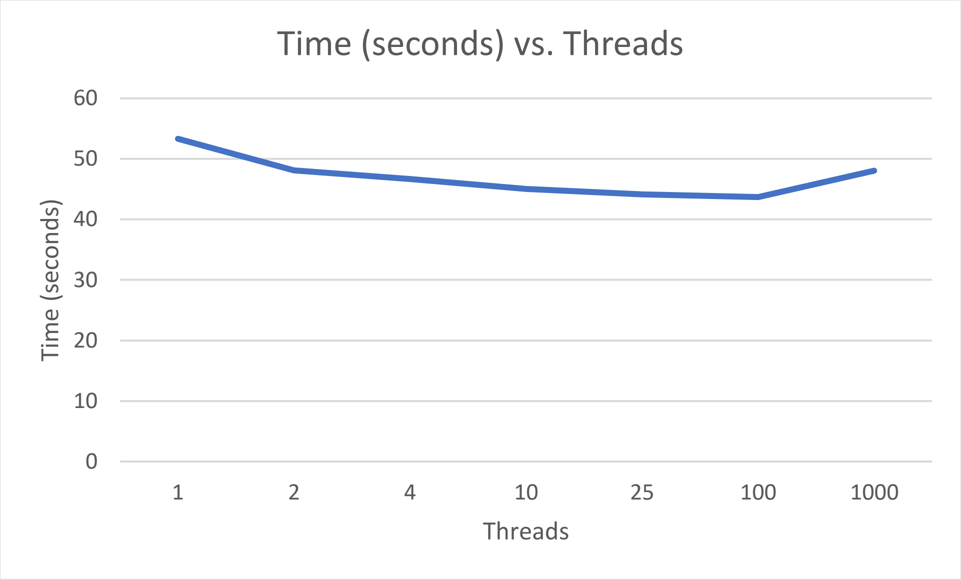
Time (seconds) vs. Threads

Test	1 Thread	2 Threads	4 Threads	10 Threads	25 Threads	100 Threads	1000 Threads
1	53.937	48.217	46.405	45.271	44.152	43.819	47.534
2	53.542	47.984	46.815	44.947	43.776	43.34	48.163
3	53.646	47.997	46.189	44.523	44.093	43.757	48.343
4	52.898	48.102	46.097	44.733	44.348	43.522	48.373
5	53.132	48.249	47.277	44.598	43.992	43.903	47.869
6	52.976	47.92	46.44	44.89	43.786	43.721	47.979
7	52.82	47.851	46.771	45.661	44.617	43.683	48.46
8	53.562	48.219	47.077	44.912	44.271	43.676	48.186
9	53.415	48.283	47.352	45.412	44.185	43.615	47.426
10	53.347	48.072	46.153	45.421	44.217	43.938	48.437
Avg	53.328	48.089	46.658	45.037	44.144	43.697	48.077

Average distance found is 31.904232

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 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. 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 100 threads. Using more threads resulted in an execution time slower than 100 threads. When using 100 threads, the program took, on average, 43.697 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, 100 threads are the optimal number to calculate the average distance between stars.