Project 2

Jace Domingo & Alan Benitez

COP4634 Systems & Networks

10/17/2021

## Background

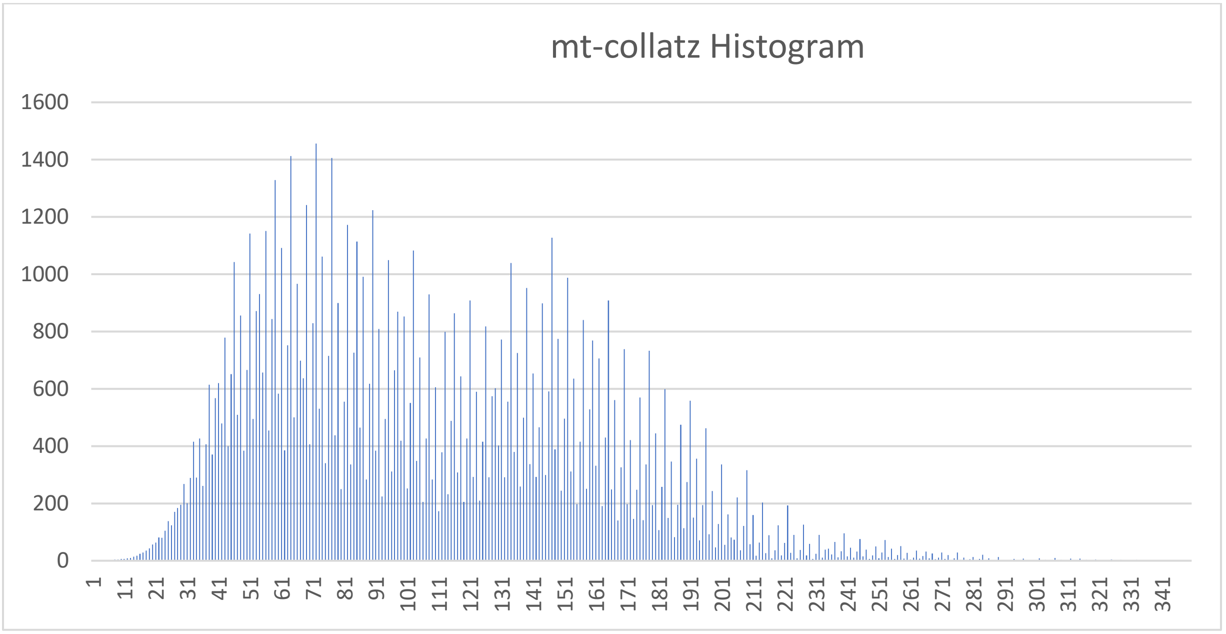
The purpose of this project was to demonstrate the strengths of multithreaded processing as well as the weaknesses. We were to develop a program that effectively utilize multithreaded processing while dealing with its shortcomings.

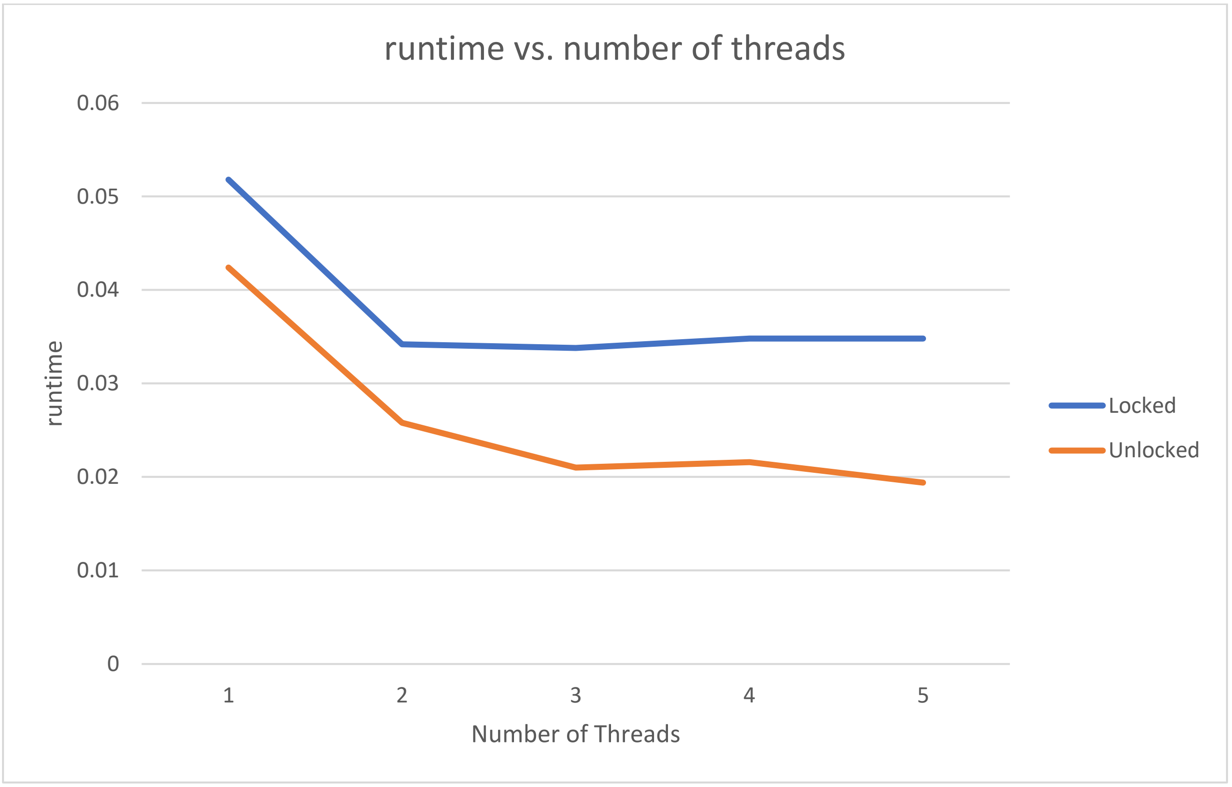
## Experiment

In this experiment we were to develop a program that would produce a list of Collatz sequences for a user specified range of numbers(N). We were to accomplish these calculations using a user specified number of threads working in parallel. The program executes successfully while sending the sequence to the standard output stream, and the process time to the standard error stream.

## Results

Shown below is a bar graph depicting the results of the Collatz sequences, and a line graph comparing the runtime vs. number of threads with lock enabled and disabled. The number range (N) used was 100000, with 1-5 threads.





## Discussion

The results of my experimenting show an initial benefit to multithreaded processing that quickly falls off as I add more threads. I assume this is because adding threads can only shorten the computation time by so much, and this was the limit of the range of N I decided to use. Adding more threads would probably be more useful if I had used a much longer range, but the loop in my program stops at around 190000 and I haven't been able to determine the cause. Even with a limited N, the program seems to show that runtime is also lower without locks. This would make sense since the lock limits one thread to be able to acquire and change shared data at a time.

## Conclusion

In conclusion, this program demonstrated that multithreaded processing can be very advantageous if used properly. Though using multiple threads can be very useful, there is a limit to how many threads you can add that benefit the program. Thought it impedes the runtime, locks are necessary to prevent racing conditions that could compromise the data.