CS 6346 P1 Report
JAYANTH AVINASH POTNURU, JXP220032

I have implemented FilterLock, FilterLock with Generalized Peterson Algorithm, Bakery Algorithm lock, and a variant of TournamentTree algorithm.

I used Java programming language for the implementation and testing compared to the initially planned C. This is because Java is platform-independent, and C, Arm64, and x86 architectures have different memory consistency models, making it difficult to reproduce the results. In the future, I plan to release an Arm64v8 version for the same in C in my Git Hub.

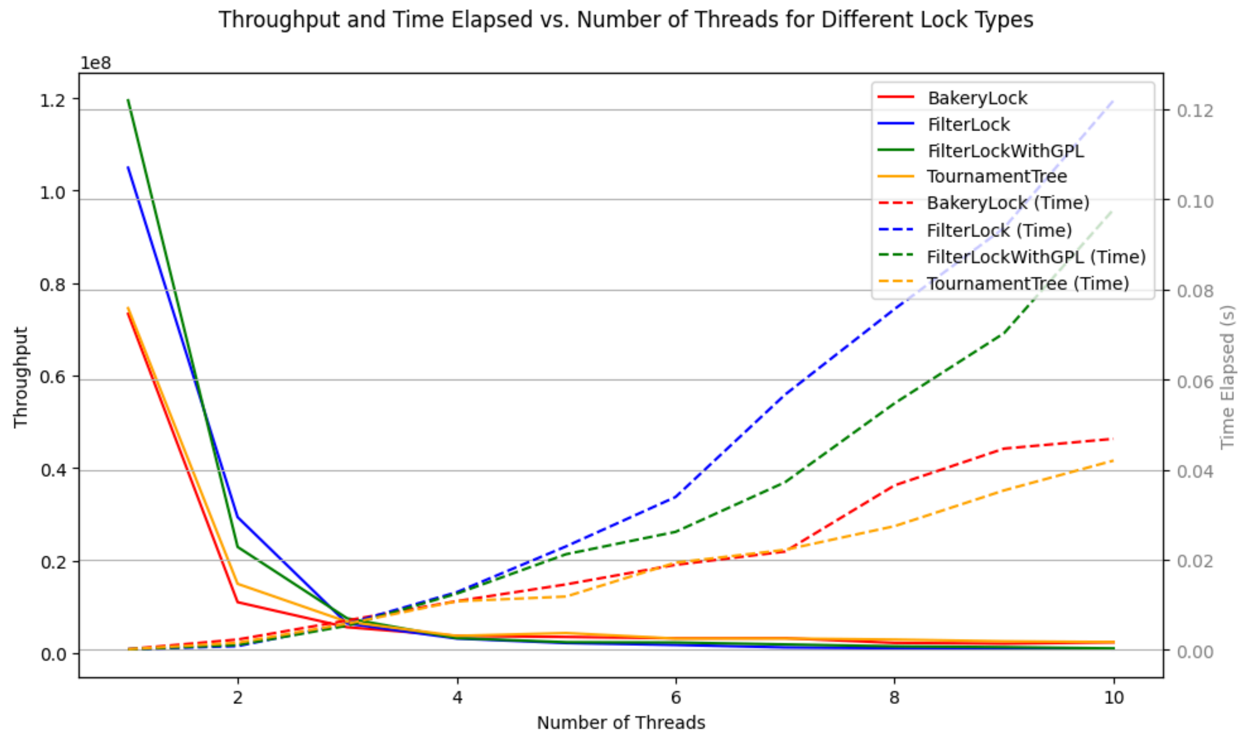
Coming to our experiments,
we define

Throughput = number of times a critical section is entered per unit of time.

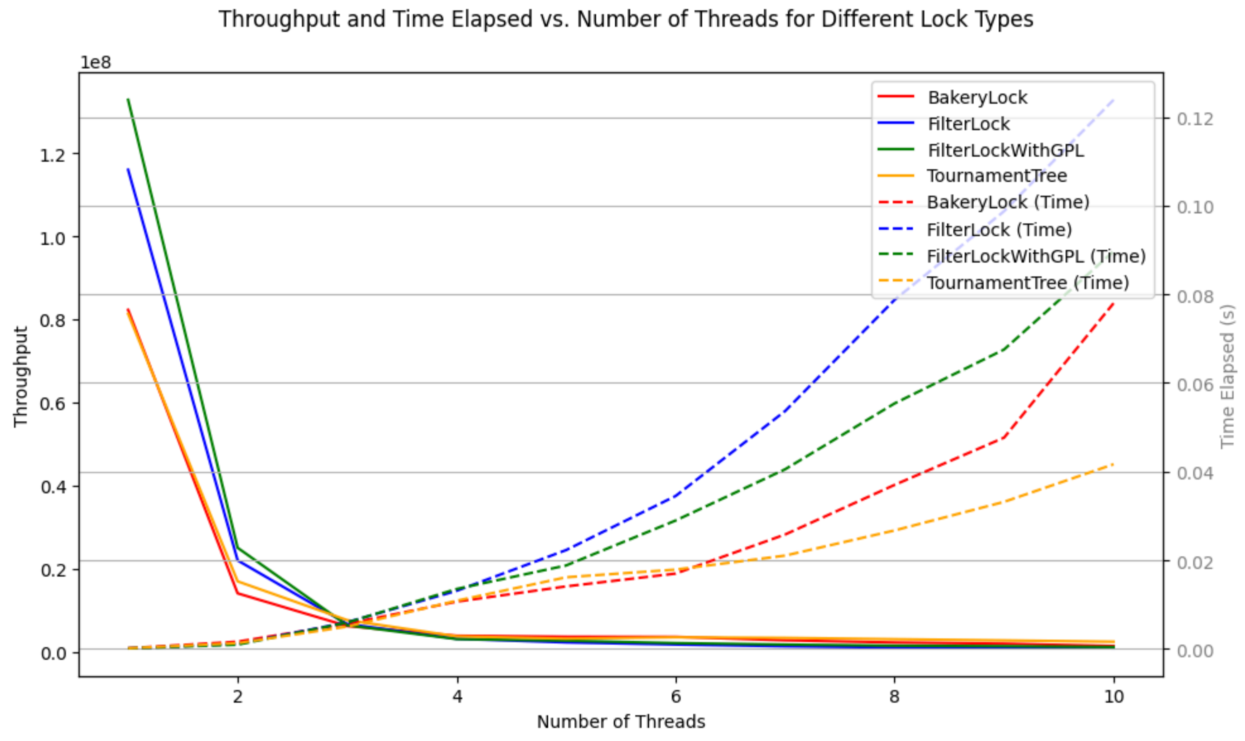
```
double iterationsPerSecond = (numThreads * COUNTER_LIMIT) /elapsedSeconds;
```

Turnaround time/ elapsed seconds is the amount of time taken for all threads to access the critical section and perform their job of incrementing the counter by 10000. This is averaged over 100 to 200 iterations.

I used Matplotlib to plot the data, and the plots look as below for 100 iterations



For 200 iterations



The submission contains Main.java, CustomLock.java (interface), Data.csv, GraphPlot.py and Readme.md file

Important note:
The tournament tree algorithm is implement by using BakeryLocks for every node as it provides First-Come First Serve Property. Also the implementation is bit simplified.

From the performance charts , we can see
Tournament Tree with Bakery Lock > Bakery Lock > Filter With Generalized Peterson Lock and > FilterLock