CS5300 Assignment 3

Gautam Singh
CS21BTECH11018

Contents

***Abstract*—This report analyzes C++ implementations of Filter and Bakery Locks. In particular, we compare the throughput, average and worst case critical section entry time for each implementation. The rest of the report is organized as follows. In section 1, we describe the low-level program design on which experiments are run. In section 2, we compare the performance of both locks with varying parameters. Finally, we conclude the report in section 3.**

1 Program Design

We give here an overview of the implementation details of the two programs, one for the Filter Lock and one for the Bakery Lock.

1.1 Threads and Runner Functions

Threads are created using the C++ `std::thread` class. This makes it convenient to pass thread IDs and references to other objects using `std::ref` to the thread runner functions. In particular, we pass the following arguments to the runner function.

1) Thread ID from 0 to $N-1$, where N is the number of threads. This is because the `std::thread` class creates threads with IDs that may not be in this range, as they are only meant to be unique.
2) Reference to the lock object.
3) Logging output stream (a reference to a C++ `std::stringstream` object) to write timing information for further analysis and output.

1.2 Timing

The timestamps are reported using the `std::chrono` library. To prevent zero values, we report times in nanoseconds, which is the smallest available unit of time. However, results in the analysis are suitably scaled and reported in milliseconds where needed.

1.3 Random Number Generation

Random delays for simulating critical section and critical section request are generated using exponential distributions with mean λ_1 and λ_2. These distributions are instantiated as C++ `std::exponential_distribution` objects. The randomness is generated using a Mersenne Twister, instantiated using an `std::mt19937` object. This random number generator is seeded using the current time since epoch.

2 Results and Analysis

In this section, we analyze the performance of the locks on metrics such as throughput and entry time when varying the total number of threads and the total number of critical section requests made by each thread. This application was run on an Intel i9-11900H processor.

2.1 Throughput Analysis

We define the throughput of our application to be the number of tasks completed per unit time. In this case, accessing the critical section once is considered as one task. Thus, the total throughput is $\frac{NK}{T}$, where N is the total number of threads, K is the number of critical section requests made by each thread and T is the total time from creation to tear down of threads.

Figure 1 depicts the variation of throughput with N and Figure 2 depicts the variation of throughput with K. We observe the following.

1) Throughput for both locks increases approximately linearly with N.
2) Throughput for both locks increases with K.
3) As K increases, the competition to enter the critical section levels off as the time in critical section tends to the mean λ_1 of the exponential distribution, thus the rate of increase of throughput at a fixed N decreases for both locks.

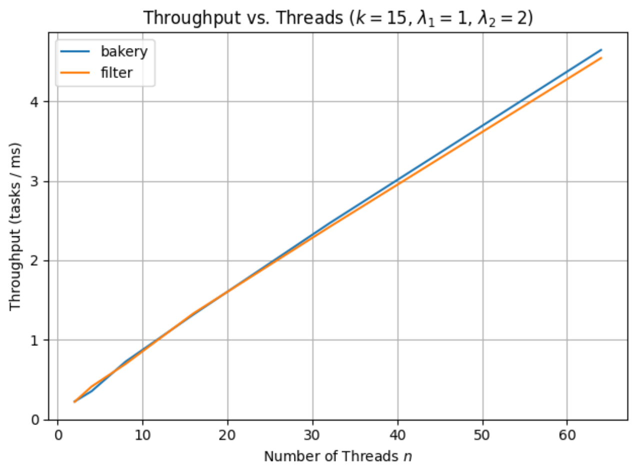

Figure 1. Throughput with varying number of threads.

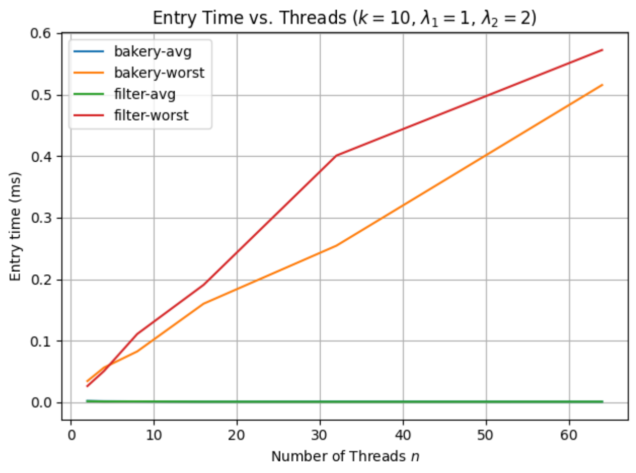

Figure 3. Entry time with varying number of threads.

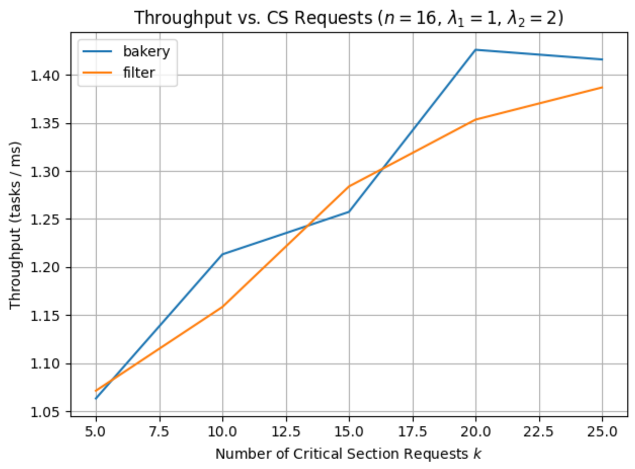

Figure 2. Throughput with varying number of critical section requests per thread.

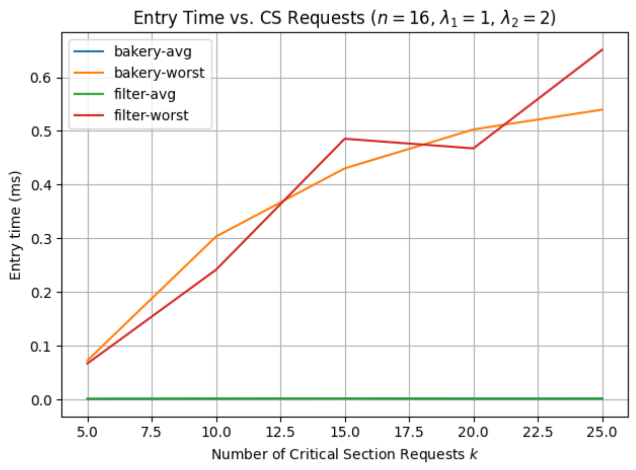

Figure 4. Entry time with varying number of critical section requests per thread.

4) The bakery lock shows better throughput with increasing number of threads and critical section requests. This is because acquiring the filter lock involves more read/write operations as compared to the bakery lock. In other words, the lock method is much slower for the filter lock.

2.2 Entry Time Analysis

The entry time is defined as the time taken to acquire the lock and enter the critical section. We perform an average entry time analysis and a worst-case entry time analysis for each lock implementation.

Figure 3 depicts the variation of average and worst-case entry time with N and Figure 4 depicts the variation of average and worst-case entry time with K for both locks. We observe the following.

1) The average entry time for both locks is comparable across various values of N and K. Thus, on an average, there is no noticable difference in the entry time of a thread.
2) The worst-case entry time of each lock has an approximately linear relation with N, since there is increased competition to acquire the lock and enter the critical section.
3) The worst-case entry time increases with K as well, since threads are making more requests to enter the critical section.
4) The bakery lock has lower worst-case entry time than the filter lock, which could be due to the slow lock method in the filter lock and more chances of

a thread waiting in trying to acquire the filter lock.

3 Conclusion

From the analysis, we conclude that the bakery lock demonstrates marginally better throughput and entry time performance than the filter lock, mainly due to the slow lock method in the filter lock.