Part1: Write Up

My implementation for the synchronous circular queue is simple. With two private atomic variables, I use a Boolean as a flag and an int for the two threads to communicate with each other by store values in the box. I also have two public functions called enqueue and dequeue to push items into the box and pull items from the box, respectively. While one thread is trying to push items to the box, the other thread must spin until the enqueue thread is complete. While one thread is trying to pull values from the box, the other thread must spin until the dequeue thread is complete.

The asynchronous circular queue was more complicated than the synchronous queue because of accounting for the head and the tail wrapping back around each time the queue is full or empty. Modular arithmetic must also be implemented to make sure that the head and tail are incremented correctly. We also include two additional functions called enq_8 and deq_8, which enqueues/dequeues eight elements from the queue at a time, respectively.

The results of part1 were interesting because of the performance difference between the three different Queue implementations. Based on the Part1 graph that I made, the synchronous queue outperforms all the other queues. This is probably because the implementation of the asynchronous queue and queue8 API are not optimized compared to the queues in Part2.

This shows that throwing more threads at a problem does not necessarily give better performance. The problem must be suited for threads and optimized properly to see adequate performance gains.