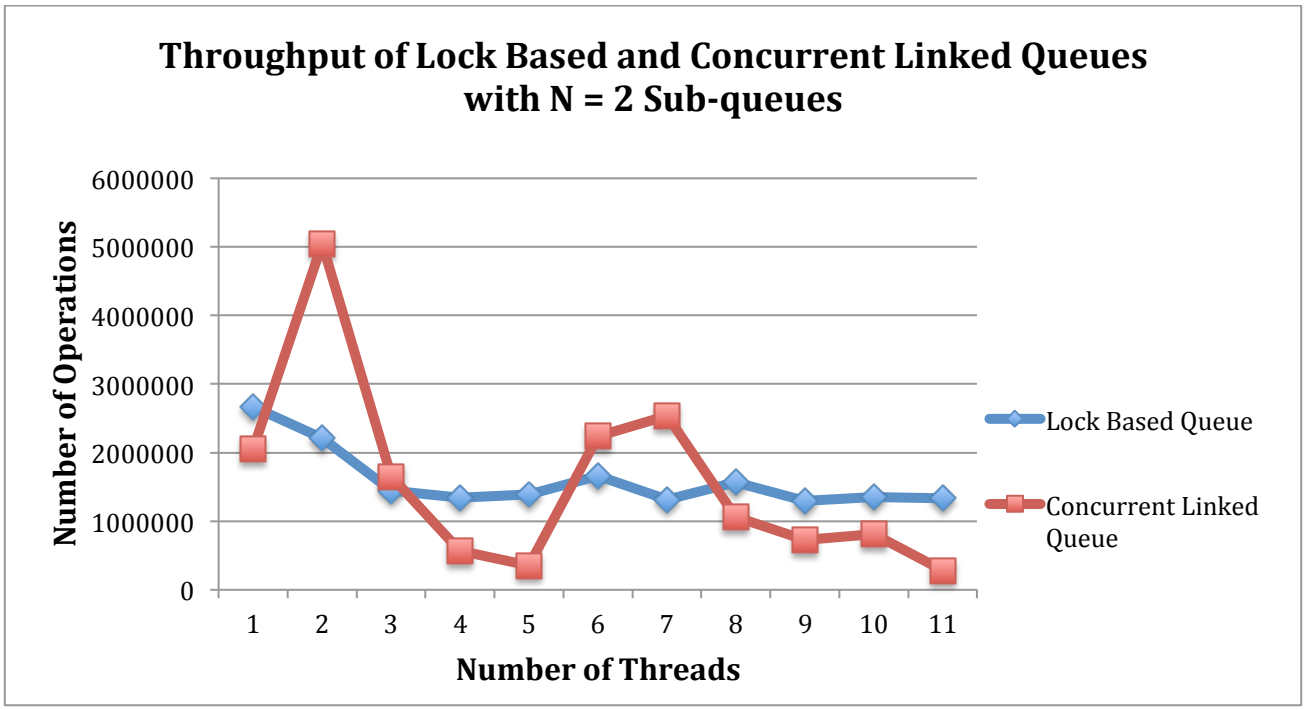
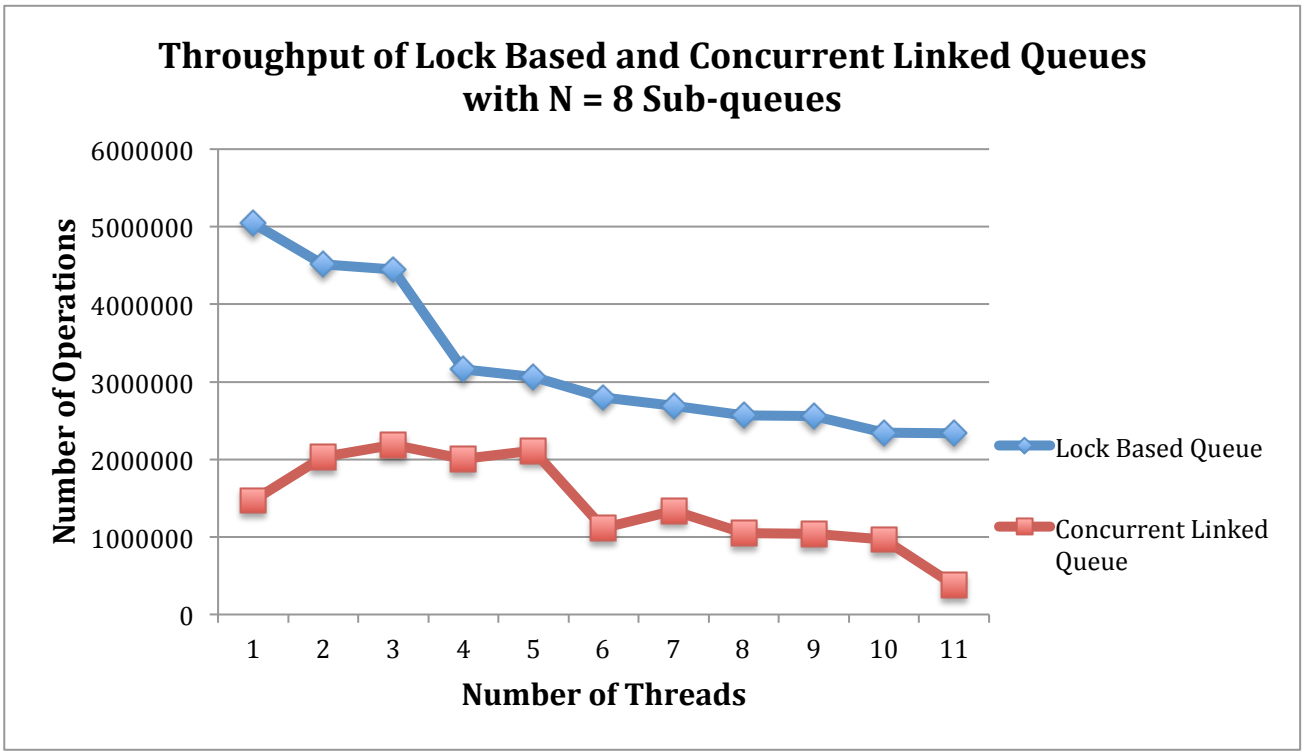
What are the interesting trends and behaviors you observed?

The Lock based Queue with two sub-queues starts at 26,000 then descends quickly into the fifteen thousand ranges as the number of threads increases. The Concurrent Linked Queue with two sub-queues starts at 20,000 then spikes to 50,000 before fluctuating into the 5,000 to 25,000 range. The Lock Based Queue performs the best with one thread since there are no waiting threads and the add/remove methods are synchronized, so access can only happen one at a time. The Concurrent Linked Queue must have a way to handle two queues effectively and some optimization greater than two threads that increases throughput sometimes. I noticed that the Concurrent Linked Queue gave varying results for the number of threads added if the program was executed multiple times.

The throughput of the Lock Based Queue with 8 sub-queues starts at 50,000 and descends to the 20,000 to 30,00 ranges. The Lock Based Queue now has more sub-queues so the throughput is higher and there are less chances of hitting exceptions. The Lock Based Queue still had the highest throughput for one thread because add/remove are linearized by the synchronized keyword. The throughput of the Concurrent Linked Queue with 8 sub-queues starts at 15,000 and ascends to the 20,000 ranges before descending to the 10,000 ranges. The Concurrent Linked Queue has more sub-queues, so fewer exceptions are thrown and most have an optimized way of handling greater than one thread such that the throughput increases. The throughput for the Concurrent Linked Queue varied for the number of threads added if the program was executed multiple times.




Throughput of Lock Based and Concurrent Linked Queues with N = 8 Sub-queues
Number of Operations
6000000
5000000
4000000
3000000
2000000
1000000
0
1
2
3
4
5
6
7
8
9
10
11
Number of Threads
Lock Based Queue
Concurrent Linked Queue