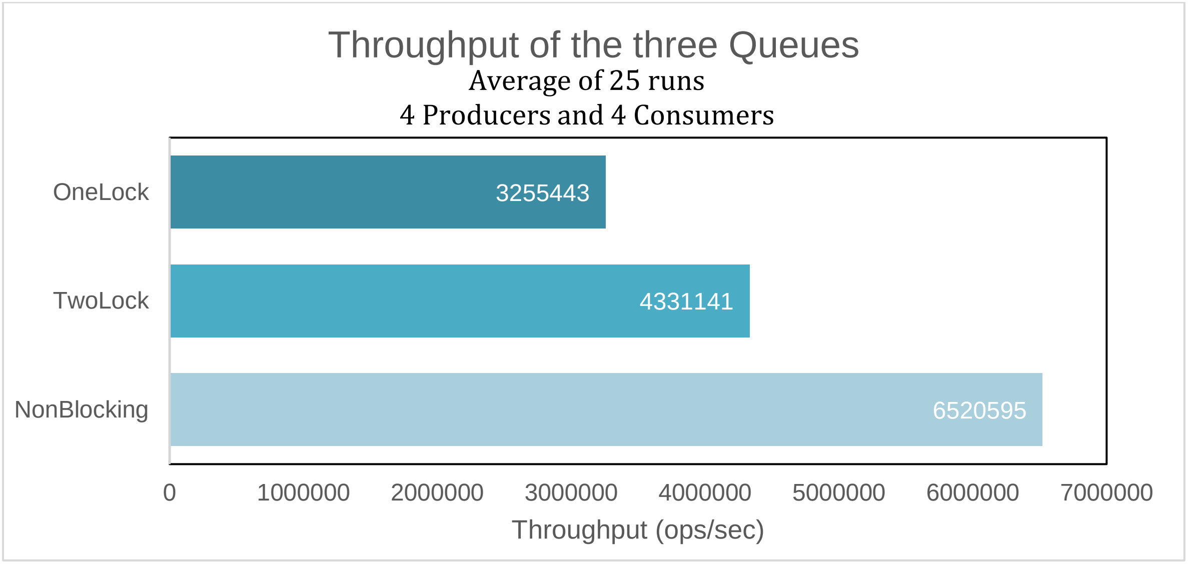
CMPT 770/479: Parallel & Distributed Computing (Fall 2021)
Assignment 5 - Report

Instructions:

- This report is worth 20 points.
- Answer in the space provided.
- Answers spanning beyond provided lines (11pt font) will lose points.
- All your answers must be based on the experiments conducted using the expected number of threads (as specified in the questions below) on fast nodes. Answers based on slow nodes and/or different numbers of workers than expected will result in 0 points.

---

1. [5 Points] Plot the total throughput for OneLock Queue, TwoLock Queue and Non-Blocking Queue with 4 producers and 4 consumers as a horizontal bar plot. The structure of your plot should be similar to the sample shown below: the x-axis has throughput in terms of number of operations ("**`Total throughput`**" from output) per second, and y-axis has the three queue implementations. The \`a', \`b', \`c', \`d', … on x-axis should be numbers (on linear scale).
   There may be some performance variation between runs, and hence you should take the average of 10 runs for each queue implementation. Remember to allocate one CPU per producer/consumer thread so that multiple threads do not have to fight for the same CPU (i.e., 4 producers + 4 consumers should be run using 8 CPUs).



2. [4 Points] Based on your plot from question 1, why does the throughput vary as it does across different queue implementations?

   The throughput decreases as we go through the tasks, this is because the amount of contention decreases in each incremental update. In the OneLockQueue Implementation both of the functions were shooting for the same "lock" (and enq() deq() both behave sequentially) which creates contention, which was mitigated to some extent when two locks (one for each function) were implemented, whereas, in the nonblocking implementation the throughput increased due to CAS ops

3. [7 Points] In Non-Blocking Queue pseudocode, there are two lines labeled as ELabel and DLabel. Why are those two lines present in the algorithm? Will the correctness and/or progress guarantees change if they are removed? If yes, which ones and why? If no, why? Support your argument using formal terms taught in class (e.g., linearizable, sequentially consistent, lock-free, wait-free, etc.).

   The Elabel and Dlabel for aiding "tail" pointer, they change the "tail" pointer to point to the "last" node. The correctness and progress guarantees will no longer be true if we remove the two lines. For enqueue(), the control gets stuck in the while loop because next.address() == NULL is always false. For dequeue(), the loop takes the control to the location where "tail" is pointing to, which is not the tail.

4. [4 Points] In Non-Blocking Queue pseudocode, what is the purpose of the counter that is co-located with the next pointer? What can happen if the counter is removed? Please provide an example to support your argument.

   The counter is there to make sure that while incrementing it gets the unique pointer location. This is the ABA problem, where the same node comes back to the queue in different context. In this problem CAS is behaving as it should be, but the meaning of what CAS operation was trying to do has changed. If these lines are removed this unlikely event will occur. If two nodes are read by a thread, and the thread gets context switched another thread comes in and performs two dequeues and first node is again brought from the free list to the queue, which is when ABA occurs if thread wakes up.