CMPT 431: Distributed Systems (Fall 2019)

Assignment 3 - Report

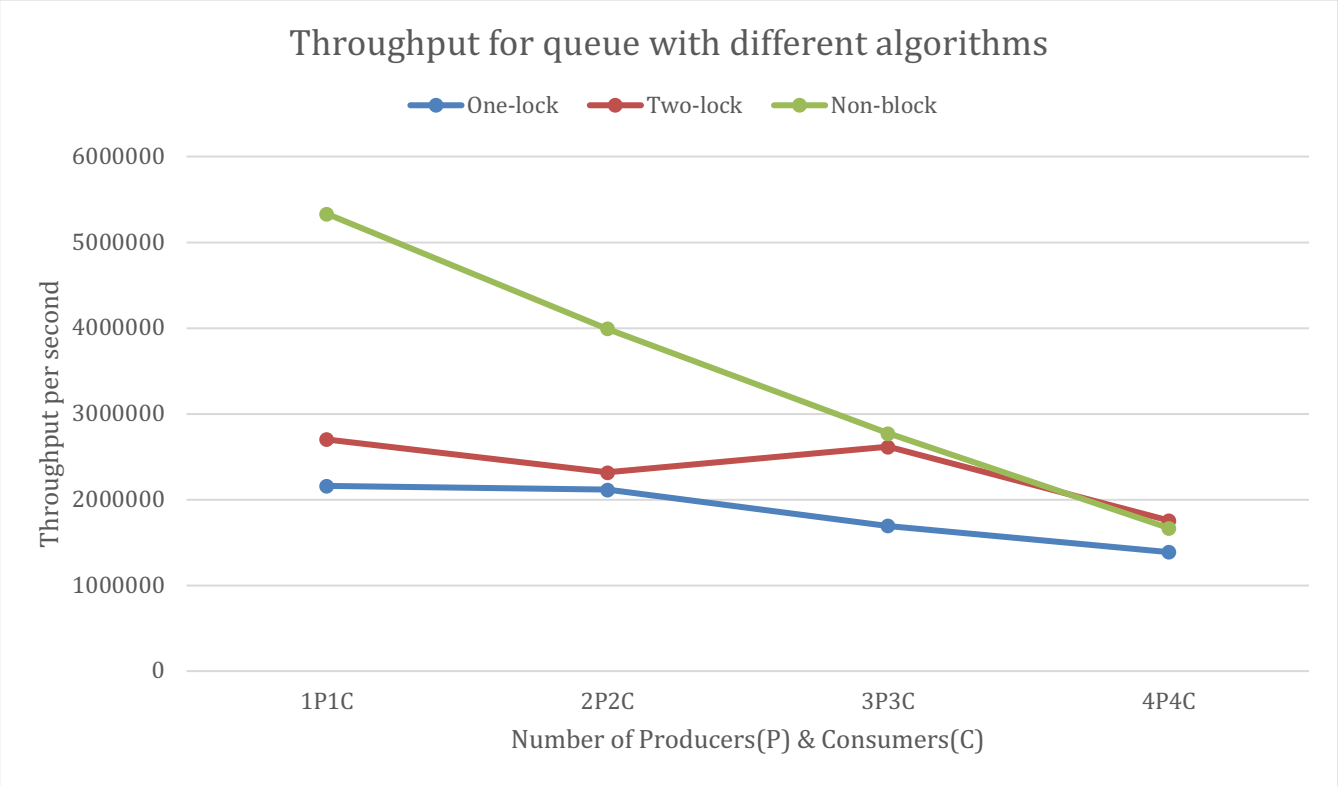
Name	Raghav Mittal
SFU ID	rmittal (301307947)

Instructions:

- This report is worth 20 points.
- Answer in the space provided.
  Answers spanning beyond provided lines (11pt font) will lose points.

---

1. [5 Points] Plot the throughput per thread for OneLock Queue, TwoLock Queue and Non-Blocking queue. The structure of your plot should be similar to ones shown in class where x-axis has number of producers (P) and consumers (C) between 1 and 4 (i.e., 1P1C, 2P2C, 3P3C, 4P4C), and y-axis has throughput in terms of number of operations ("`Total throughput`" from output) per second. Your plot should have three lines (one for each queue).



2. [4 Points] Based on your plot from question 1, why does the throughput for each queue implementation vary as it does when producers and consumers increase? Why does the throughput vary as it does across different implementations?
   Throughput for 1-lock & 2-lock stays about same on increasing producers (PR) & consumers (CS) as they only allow 1 & 2 operations concurrently respectively. For non-blocking, contention for head & tail pointers results in increased cache misses & in higher completion time of an operation on increasing PRs and CSs and enqueuers & dequeuers need to re-traverse repeatedly. Order of performance: one-lock (allows 1 operation (ops)) < two-lock (allows 2 ops) < non-blocking (allows more than 2 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.).
   Dequeuers becomes blocking on removing DLabel as they will depend on enqueuers to update tail. On removing ELabel, subsequent enqueuers become blocking on the first enqueuer who was able to add logically to finish physical addition as well. Both enqueuers and dequeuers will become blocking on the enqueuer mentioned above if both are removed. Algorithm stays correct in all scenarios.

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 co-located to eliminate ABA problem. If removed, CAS operation of updating head while dequeuing can succeed even if head pointer has been manipulated since we checked it and can end up pointing to a removed node. Ex: (Queue before starting deque): Head->A->B (A-sentinel). While dequeuing, thread1 goes to sleep after reading A & B who are removed after that (head updated). A & B are with allocator in free nodes list and A is enqueued back and becomes sentinel again. Thread1 wakes up, CAS succeeds and now head points to B which is in free nodes list.