**Paper Review Writeup – Lock Free Queues**

**CS 554: Data Intensive Computing**

**Reviewer Name**: Nilesh Balu Jorwar (A20405042)

**Paper Title**: Simple, Fast, and Practical Non-Blocking and Blocking Concurrent Queue Algorithms

**Paper Authors**: Maged M. Michael, Michael L. Scott

**Publication Venue**: Proceeding PODC '96 Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing. Pages 267-275. Philadelphia, Pennsylvania, USA

**Year of Publication**: May 23 - 26, 1996

**Summary of Paper**

In a multithreaded environment, threads communicate through a data structures such as queues, stacks, and so on. To facilitate correct and simultaneous access to data structures over multiple threads, data structures must be protected by a concurrent algorithm. If the algorithm that protects a concurrent data structure is blocking (using a thread suspension), then it is a blocking algorithm. If the algorithm that protects a concurrent data structure is non-blocking, then it is a non-blocking algorithm and the data structure is a non-blocking data structure.

Authors proposed the non-blocking concurrent queue and two lock-queue.

Authors claimed that the properties of linked-list are held in implementing these data structures as linked-list is always connected, followed by the insertion and deletion operations onto the list as per the queue properties.

They claimed to achieve the linearizability while claiming that lock-free algorithm implemented is non-blocking. They also presented two-lock algorithm which is Livelock-free and suggested this data structure to be used for machines that are not multi-programmed and does lack in universal atomic primitive such as compare-and-swap.

The performance of these algorithms is tested onto the 12 processor Silicon Graphics Challenge multi-processor and compared with Single Lock algorithm presented in related work. Results of the evaluation has shown the effective use of two lock algorithm in multi-processor system not multi-programmed ones.

**Contributions**

* Introduced non-blocking data structure that implemented the queue as singly-linked list with head and tail pointers where head points to dummy node (first node in list) and tail points to last or second to last node in list. The algorithm used compare-and-swap and modification pointers to avoid ABA problem.
* Reusability of dequeued nodes is achieved through dequeuing process. Treiber’s non-blocking stack algorithm is used to implement non-blocking free list.
* Authors also presented two-lock queue data structure that has separate head and tail locks that employ concurrency between enqueues and dequeues. In non-blocking queue, we have dummy node at beginning of the list because of which enqueue never have to access head and dequeues never have to access tail, thus avoiding potential deadlock.

**Difference with related work**

* Authors listed and discussed various earlier proposed solutions for lock free algorithms for concurrent FIFO queues. They started by mentioning that Hwang and Briggs presented lock free algorithms based on compare-and-swap but failed to handle empty and single queues, concurrent enqueues and dequeues.
* Lamport came up with wait-free algorithm that restricts concurrency while Gottlieb et al. and Mellor-Crummey presented lock-free but non-blocking algorithms. Treiber’s non-blocking algorithm were inefficient while Herlihy and Prakash, Lee proposed non-blocking version of sequential and concurrent lock-based algorithms.
* Authors also listed the work of other authors such as Massalin and Pu, Stone, Prakash, Lee and Johnson in implementing non-blocking concurrent queue. However, these algorithms were based on compare-and-swap and should have the ability to handle the ABA problem at the same time that made these algorithms blocking. Hence, authors of this paper introduced concurrent FIFO queue as non-blocking and queue that uses pair of locks.
* Similar to Valoi’s algorithm, non-blocking queue is implemented as singly-linked-list with Head and Tail pointers where head points to dummy node and tail points to last or second to last node in list. This algorithm used compare-and-swap with modifications counter to avoid ABA problem. Consistent values of pointers are obtained using sequence of reads which are similar to Prakash et al. algorithm. Treiber’s non-blocking stack algorithm is used to implement non-blocking free list.

**Pros**

* Linearizability is achieved at algorithm assures single set of operations on single object.
* ABA problem is handled in non-blocking queue.
* The non-blocking concurrent queue data structure can be used in transaction safe versions of lock-free queue.
* Non-blocking queue algorithm is a choice for multiprocessors that support universal atomic primitives.
* Non-blocking atomic update algorithms outperforms all alternatives, not only on multi-programmed systems, but on dedicated machines as well.
* Two-lock queue algorithm allows one enqueue and dequeue to run concurrently. This algorithm should be used for heavily-utilized queues on multiprocessors that has non-universal atomic primitives such as test-and-set.

**Cons**

* A queue which is accessed by the single or two processors, single lock would work better than two lock-algorithm.

**Improvisation for Author**

**Future Work**