Project Proposal

Concurrent Non-Blocking Priority Queue Implementation

CS-550 Advanced Operating System

September 5th, 2017

Pradyot Mayank (A20405826)

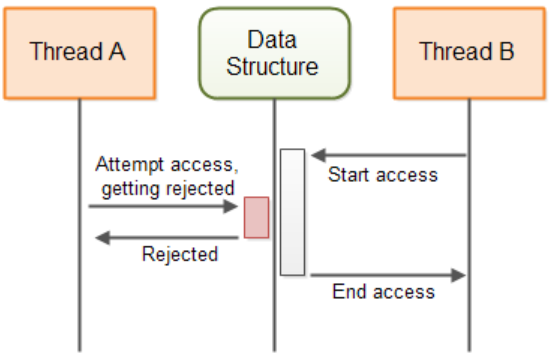
Nilesh Jorwar (A20405042)

1. Introduction

Writing a thread-safe code is, at its core, about managing access to shared, mutable state. Making data structure thread-safe requires using synchronization to coordinate access to its mutable state; failing to do so could result in data corruption and other undesirable consequences [1]. A priority queue is a data structure that is very often needed at places such as the process managers of operating systems, distributed event simulations, network bandwidth managers and sorting algorithms and thus may face severe contention.

2. Background Information

With non-blocking and lock free implementation of priority queue we can achieve scalability and real-time response. Each process does not need to wait for its turn if there are multiple threads requesting the same resource. Unlike blocking concurrency algorithm, non-blocking concurrency algorithm either performs the operation or notifies the thread that operation cannot be performed at that state of time.



Non-blocking concurrency algorithm

3. Problem Statement

With the fast pacing world and emerging technologies it becomes human and technology necessity to have access and availability to those with the higher priority. Though the currently available applications are capable enough to provide the services to those which are based on First Come First Serve (FCFS) policy but are subsequently failing to mitigate the exigencies needed for higher preference demanded. Widely available parallel applications based on the thought of Concurrent FIFO queues failing to serve clients/applications that need an immediate attention. This need leads to increase in indefinite idle time for waiting clients and makes the system slower due to flooding their requests.

4. Related Work

There has been development in the past regarding implementation of wait free and lock free algorithms on concurrent priority queue. The algorithm presented by Hunt et al [7] is based on heap structures, it accesses the heap by scatter technique while locking each node separately,

thus reducing the contention. This algorithm has been tested on multi-processor systems. The wait free algorithm for a concurrent priority queue [8] was presented by Israeli and Rappoport. This algorithm uses strong atomic synchronization primitives.

5. Proposed Solution

Currently the programming languages have implementations of priority queues and a thread-safe priority blocking queues. Even though the latter is thread safe, it is not efficient as it blocks the concurrent threads for synchronization which may lead to starvation of some threads indefinitely. . In this project we will try to achieve a non-blocking synchronization for priority queues. This implementation will be based on an algorithm as described in 'Simple, Fast, and Practical Non-Blocking and Blocking Concurrent Queue Algorithms' by Maged M. Michael and Michael L. Scott

6. Evaluation

The evaluation of this project is comprised by the following:

- Concurrent and non-blocking implementation of Priority Queue data structure.
- Performing all the operations of the queue without affecting the existing complexities.
- Performance of the algorithm will be calculated based on complexities and big O notation.

7. Conclusion

Our programming solution will implement non-blocking concurrency for priority queues. This priority based non-blocking data structure will work more efficiently for scheduling and sharing of the resources over the distributed network. We are going to check how high priority applications, resources get executed with minimal or no idle time. It is also going to check the performance improvement as compared to available non-blocking FIFO algorithms.

8. Additional Resources

8.1 Timeline

Sr. No	Task	Week
1	Requirement Gathering 1.1 Analysis of Existing system 1.2 Literature Survey	1
2	System Design 2.1 Architecture Diagram 2.2 Flow Chart	2
3	Algorithm Designs	3
4	Mid Term Report Preparation	4
5	Implementation of proposed solution	5
6	Testing and evaluation of the proposed solution	6
7	Validation and Modification to	7

	the proposed solution	
8	IEEE Paper and final report preparation	8
9	Project Presentation	9

8.2 Deliverables

- Progress/Mid term report
- IEEE Paper
- PPT Presentation
- Source Code

9. References

[1]http://tutorials.jenkov.com/java-concurrency/non-blocking-algorithms.html#non-blocking-concurrent-data-structures

[2] https://www.cs.rochester.edu/research/synchronization/pseudocode/queues.html

[3] http://blog.shealevy.com/2015/04/23/use-after-free-bug-in-maged-m-michael-and-michael-l-scotts-non-blocking-concurrent-queue-algorithm/

[4]https://secweb.cs.odu.edu/~zeil/cs361/web/website/Lectures/priorityQueues/pages/ar01s02.html

[5] http://pages.cs.wisc.edu/~siff/CS367/Notes/pqueues.html