Final Project Proposal

Yutong Zhu (yutongz3)
Haoyang Wu (haoyang2)
November 3, 2021
Project URL: Lock-Free Queue Using CAS

Summary

In this project we plan to implement a concurrent lock-free queue using CAS. We will evaluate the performance of our lock-free implementation by comparing it with locked queues of different granularity, which we will also implement.

Background

There are many problems associated with locks in shared data structures, such as overheads in high-frequency synchronisations. Rather than achieving concurrent operations through mutual exclusion, lock-free data structures rely on synchronization primitives such as compare-and-swap (CAS) while ensuring that no process is forced to wait for another process to complete a queue operation.

FIFO Queue is a widely used data structure for parallel scenarios such as work assignment, which we discussed in class. A basic queue supports enqueue and dequeue. However, things can get tricky when we have multiple producers and consumers accessing shared resources concurrently.

Challenge

The challenge in implementing a lock-free queue using CAS will be to use CAS calls to test whether the update is safe and the introduction of ABA problems.

The case of single-linked queue with Head and Tail pointers presents a difficulty because to do an Enqueue operation, we need to update two shared locations: 1) make the last

node point to the new node and 2) make the Tail pointer point to the new node. We cannot perform two updates using a single CAS.

Resources

We will be starting our project code from scratch while relying on some papers as references:

- "Verifying Michael and Scott's Lock-Free Queue Algorithm using Trace Reduction" by Lindsay Groves
- Implementing Lock-Free Queues by John D. Valois

We will need machines with CPUs that support CAS operation. We will be using GHC clusters for development and performance analysis.

Goals and Deliverables

Plan to achieve

We will focus on not only implementing the lock-free queue and test its correctness but also on evaluating its performance along with other concurrent queue solutions (e.g., traditional lock techniques).

1. Implement lock-free queue with CAS.
2. Implement queues with lock of different granularity
3. Evaluate and conduct performance analysis.

Hope to achieve

If time allows, we hope to evaluate the performance of our lock-free queue more thoroughly by comparing with more other implementations, such as blocking queues with test-and-set lock and ticket lock. Also, we would be interested in exploring the performance of other lock free queue implementation.

If time allows, we are also interested in further evaluation of performance by comparing the speedup improvement of lock-free queue vs the performance improvement of a lock-free stack (which is discussed in class)

If work goes slow

If we are behind schedule, we will only compare performance with queue of coarse granularity locks.

Deliverables

We will conduct performance analysis and produce speedup graphs between the different implementations of queue data structure. We expect that the lock-free implementation will have better performances than other implementations. We will also investigate how the performance of our lock-free queue changes under different parallel scenarios.

Platform Choice

We will be using GHC cluster machines for development and performance analysis because we need CPUs that support CAS operation. Also, GHC machines provides hardware support that we can utilize for testing high contention scenarios. We will use C++ for implementing concurrent queues. As for the evaluation analysis, it's possible that we will use script language like Python because it's more efficient in plotting and analytical work.

Schedule

Nov 5 - Nov 12	Read paper and study implementation of lock free queue
Nov 13 - Nov 19	Implement lock free queue and unit tests
Nov 20 - Nov 26	Implement lock based queue with different granularity
Nov 27 - Dec 3	Evaluate and analyse performance
Dec 4 - Dec 10	Finish final report and poster