Jeremy Holloway
CPSC-3220-001
5/29/2019

Pre-class Assignment #9

1. How does the experiment described in section 6.1 show the effect of false sharing?

It showed false sharing would cause a significant slowdown during critical section execution time. This can be measured by having two threads iterate over the same array getting the even or odd elements, ideally there should be no effect.

2. What are the three methods identified in section 6.2.1 when a program must obtain multiple locks at the same time in order to restructure a shared object that uses fine-grained locks?

1. Introduce a readers/writers lock
2. Acquire every lock
3. Divide the hash key space

3. What is the basis for the performance advantage of per-processor data structures?

Data could be stored on separate processors, which would then need to be communicated between the two and cause a significant slow down during the execution of the critical sections.

4. Does a thread in the ownership design pattern hold a lock while it processes an object? Explain why or why not.

No, the thread removes the object from the container its stored in and then can access it without holding the lock.

5. What is the key idea in the MCS lock as compared to a regular spin lock?

The lock passes a counter variable using a compare and swap technique from one lock holder to the next, so only one thread at a a time can execute the critical section. This will allow the threads to queue up waiting for a lock variable without having to use a spin lock.

6. What is the key idea in RCU as compared to a RWLock?

RCU optimizes the read path to have extremely low synchronization costs even with a large number of concurrent readers.

7. What does the compare_and_swap instruction do?

An atomic read-modify-write instruction that first test the value of a memory location, and if the value has not changed, sets it to a new value.

8. What is the grace period in RCU?

For a shared object protected by a read-copy-update lock, the time from when a new version of a shared object ius published until the last reader of the old object is published until the last reader of the old version is guaranteed to be finished.

9. What is a lock-free data structure?

Concurrent data structure that guarantees progress for some thread: some thread will finish in a finite number of steps, regardless of the state of other threads executing in the data structure.

10. Can a thread starve when using optimistic concurreny control?

No, because optimistic concurrency control is lock-free so it would never get dead locked and starve.