Brian Quinn

Multicore Programming

Professor Mitchell

5 April 2017

Homework 3

1. A) The concurrency bug prevalent in this code is an atomicity violation. This is due to the assumption that the second thread will not run in between the lines of code in the first thread. An easy solution to solve this would be to lock both threads before that critical section using a mutex and unlocking after. Another simple solution would be to simply put the line `thr_glob->proc_info = nullptr;` after the fputs in thread one. Without more context there is really no reason why that line of code would need to be executed in a separate thread.

2. The sentinel simplifies a concurrent queue because it allows the programmer to compartmentalize the queue into the tail part and the head part. By doing so, the queue can now use CAS (compare and swap) using this sentinel value. This allows a check to see if the expected tail has been corrupted before performing an operation.

3. The ABA problem arises within memory management because pointers are spatially unique but not necessarily temporally unique. As such, we want pointer comparisons in the case of CAS to return false if it is a different pointer, even if it points to the same memory. In order to achieve this, versioned pointers can be used that use half the bits to point to an address with the other half of bits storing a counter. Reused

pointers increment the counter, thus giving a difference in bits that can be realized in a comparison.

4. Linearizability, as compared to serializability, is a guarantee that within certain critical segments, no values can be altered or otherwise corrupted and is thus isolated from concurrent processes. Serializability on the other hand is a guarantee that the resulting outcome will remain the same despite the order of a subset of linearizable parts. An example would be exemplified in our ThreadSafeListenerQueue, where the pop() method exemplifies linearizability because each operation is atomic and thread-safe, however, depending on the order of the enqueues/pops there are multiple states the remaining data can be in. However, the listen() method guarantees the outcome state will be the same because it blocks until an element is enqueued (assuming the enque runs sequentially).
5. A) One of the strengths of CAS is without locking there should be no chance of a deadlock. Alternatively, it can also be considered harder to understand and more complicated. Additionally, CAS is more vulnerable to starvation whereas locks can assure starvation does not occur. B) The issue with the tail pointer only pointing to the last element is given the case that thread 1 first enters CAS1 then thread 2 enters both CAS1 and CAS2 before thread 1 moves on to CAS2. Because of this, thread 1 has lost connection to the head. By having two caches of the tail and current tail, the program and differentiate between whether the tail had been modified since the first CAS and if so, it can respond accordingly.