Homework 1 Write Up

Shannon Ernst, Keith Stirby, Tanner Cecchetti
21 April 2017
CS 444 Spring 2017
Group 10-01

April 21, 2017

Abstract

This is the abstract

1 Log of Commands

2 qemu Flag Explanation

3 Concurrency Write Up

3.1 Main Point of the Assignment

The main point of the assignment was to introduce the idea of concurrency and refresh on threads. The basics of the assignment was to spawn a provided number of threads which would arbitrarily be assigned producer or consumer status. These threads would then operate on the same buffer space adding and removing jobs. This simple simulation allowed the student to become comfortable with pthreads.

3.2 Design Decisions

There were very few concrete requirements for this assignment. The few we did have were:

- Producers create an object containing a random number and a random wait time ranging from 2-9 seconds.
- Producers add their object to a buffer so long as there are fewer than 32 items in the buffer after a random wait time of 3-7 seconds.
- Consumers remove jobs from the buffer so long as there are more than zero in the buffer
- Consumers read the wait time on a given item and wait for that amount of time before printing the item's random value.
- Randomization technique is determined by the system and will either use rdrand or the Mersenne Twister.

Beyond these requirements, we were left to design the rest. We decided to begin with the synchronization technique. This assignment needs to have all of the threads running at the same time on the same buffer. We decided to use mutexes to control the locking and unlocking of the buffer. A producer will take a lock on the buffer to check if it is full. If it is the lock is released immediately. If the buffer is not full, the producer maintains the lock and will create an object. When an object is created the program checks if it can use rdrand and then makes a selection on which randomization technique to use. The item is then added to the current counter position in the buffer and then the counter is incremented. The buffer is an array which acts as a stack: the first item will be the last item removed. We made this decision for simplicity's sake and recognize that a better scheduler would be a first in first out structure which would have just required a rotation of adding to the array, keeping two index counters or pointers to keep track of where to consume from and where to produce to. Similar to producers, consumers take a lock on the buffer and check if it is empty. If it is empty, the consumer releases the lock. If there is an item in the buffer the consumer removes it, decrementing the counter, and reads the wait time. The consumer calls the sleep command for that amount of time. When the consumer wakes up it prints the value. The program continues until a kill command is received from the keyboard. The number of threads is received from the command line.

3.3 Testing

We did incremental testing as we developed. The trickiest thing to test was the randomization techniques. We began by testing the rdrand locally on a new MAC laptop which successfully invoked the randomization. Having confirmed that our program would run the rdrand function, we moved onto the os-class server and implemented the Mersenne Twister. We tested with a variety of threads on the os-class maxing out at 4, printing out traces of the functions being invoked.

3.4 Take Aways

Coming into this assignment, there were some reservations about implementing pthreads. This assignment has shown us that pthreads is not as difficult as we feared. We also learned how to correctly lock and unlock implicitly shared data.

4 Version Control Log

5 Work Log

Date	Time	Work performed
Tuesday, 11 April 2017	10:00 am	We met our group for the first time in recitation, we attempted to set up the vm and kernel boot in recitation unsuccessfully
Wednesday, 12 April 2017	6:00 pm	We met to successfully boot the vm and the kernel. We were able to get the vm to work but could not get the kernel to boot. We were going to ask questions later about it.
Wednesday, 19 April 2017	6:00 pm	We met and coded the concurrency assignment.
Thursday, 20 April 2017	10:00 am	Tanner continued to work on the concurrency implementation. Switched buffer implementation from stack to queue, as well as abstracted the external random number generator library. Additionally, restructured the project Git repository.
Thursday, 20 April 2017	9:00 pm	Shannon created the latex file, the make file and wrote the concurrency write up section.