ITCS 6144/8144 Assignment - Pthread Lock

Metadata

Key	Value
type	coding assignment
topic	Concurrency and Lock
release date	Sept. 10
deadline	Sept. 24
hand-in	a zip/rar file, including README, Makefile, Codes, Report
platform	Linux Only (Ubuntu)
language	C Only

Assignment Description

In this assignment, you will develop several multi-threaded pthread programs using different types of locks and compare their performance on different workloads. Detailed requirements on each of these programs are listed below.

Needed Programs

- **Program 1: Mutex Lock and Spin Lock**
  - develop a program using both pthread **Mutex Lock** and **Spin Lock**
  - the program takes n as the input parameter to denote how many threads will be created. n $\in$ [2, 4, 8].
  - each thread runs 100,000 times in a loop
  - in each loop, each thread adds 1 to a shared variable (k) [1, 1,000, 1,000,000] times (three run cases) with lock acquired
  - for each n, run the program three times [1, 1,000, 1,000,000].
  - compare the execution time of using different locks for different cases and different numbers of threads.
- **Program 2: Condition Variable**
  - develop a program using pthread **Condition Variable**
  - the program takes n as the input parameter to denote how many threads will be created. n $\in$ [1, 2, 4, 8].
  - $n - 1$ threads randomly add 1 to a shared variable (k) in an interval of 100 ms
  - one thread wait until that shared variable reaches 100 and print something
  - no performance comparison; just show the execution time of your program
- **Program 3: Reader/Writer Lock**
  - develop a program using both pthread **Reader/Writer Lock** and **Mutex Lock** to implement the same logic
  - the program takes n as the input parameter to denote how many readers(n) will be created. n $\in$ [2, 4, 8] and **one** writer
  - each of the n readers reads a shared variable (k) 10,000 times
  - the writer adds 1 to the shared variable (k) 10,000 times
  - compare the execution time of using different locks
  - identify and show when `write` happens

If you like, you can use more advanced locks (**MCS** and **RCU**) to implement the same **Program 1** and compare the performance with mutex lock and spin lock. You will get extra points.

Submissions

The submitted package should include:

1. a `README` file describing how to compile and run your code;
2. a `Makefile` to help me easily repeat your program
    - since we do not have TA, I will not be responsible for learning how to run your program;
3. your `Source Code`;
4. a `Report` describing
    1. your test platform
    2. your results (better with plots)
    3. explain why you get the results

Hints

Pthread

- A standardized multi-threaded programming interface.
- A very good pthread tutorial: https://computing.llnl.gov/tutorials/pthreads/

```c
/*
 * Please compile with gcc -Wall helloworld_pthreads.c -lpthread
 */

#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

#define NUM_THREADS 10

void *print_hello_world(void *tid)
{
   long threadid;
   threadid = (long)tid;
   printf("Hello World! Greetings from thread #%ld!\n", threadid);
```

```
   pthread_exit(NULL);
}

int main(int argc, char *argv[])
{
   pthread_t threads[NUM_THREADS];
   int status;
   long i;

   for(i=0; i<NUM_THREADS; i++){
     printf("Main here. Creating thread %ld\n", i);
     status = pthread_create(&threads[i], NULL, print_hello_world, (void *)i);
     if (status != 0){
       printf("ERROR; return code from pthread_create() is %d\n", status);
       exit(-1);
       }
     }
   pthread_exit(NULL);
}
```

Pthread Locks

- Mutex Lock
  - `pthread_mutex_init`
  - `pthread_mutex_destroy`
  - `pthread_mutex_lock`
  - `pthread_mutex_unlock`
  - `pthread_mutex_trylock`
- Condition Variable
  - `pthread_cond_init`
  - `pthread_cond_destroy`
  - `pthread_cond_signal`
  - `pthread_cond_broadcast`

    - `pthread_cond_wait`
    - `pthread_cond_timedwait`
- Reader/Writer Lock
    - `pthread_rwlock_init`
    - `pthread_rwlock_destroy`
    - `pthread_rwlock_rdlock` and `pthread_rwlock_tryrdlock`
    - `pthread_rwlock_wrlock` and `pthread_rwlock_trywrlock`
    - `pthread_rwlock_unlock`
- Spin Lock
    - `pthread_spin_init`
    - `pthread_spin_destroy`
    - `pthread_spin_lock` and `pthread_spin_trylock`
    - `pthread_spin_unlock`

Optional

- Userspace RCU: https://liburcu.org
- MCS Lock: Implement your own.