Programming Assignment 2 - Better Locking for Fun and Profit

SOIC B524 - Parallelism in Programming Languages and Systems
Indiana University

Spring 2017

1 Introduction

In this assignment you will add the `try_lock` method to the locks from PA1, implement five new locking strategies, use the `try_lock` and `yield` functions to create a cooperatively scheduled version of the test program, and benchmark the results.

To help you in this assignment we have provided `C++` versions of the *compare-and-swap*, *test-and-set*, and *unset* functions. These definitions, which make use of standard built-in functions provided by the compiler, can be found in the `pa2_atomics.h` files.

2 Setup

Begin by unpacking the archive `pa2-src.tar.xz` and extracting the code handout directory using the command `tar -xJf pa2-src.tar.xz`. This will create the directory `pa2-src`. Once you have entered the directory you will find several `C/C++` source files, a file called `CMakeLists.txt`, and the shell script `setup.sh`.

Next, copy your code from the first Programming Assignment into the appropriate files. Your implementations of the Peterson's Filter lock and the Bakery lock should go into the `locks.h` and `locks.cpp` files. Please note the small change to the lock API: a lock's functions now takes the ID of the calling thread. You should also copy the synchronization code you added to `pa1.cpp` to `pa2-busy.cpp`.

Now you may run the `setup.sh` script as you did for the previous assignment. The build system will generate an executable for each combination of lock type (e.g. Peterson's, Test-and-Set...) and synchronization style (busy-waiting or cooperative scheduling), for a total of 18 programs.

The programs take the same three arguments as last time: the number of threads to spawn, the number of rounds to perform, and the size of the data vector. The default values for these parameters are 4, $4 * \text{num\_threads}$, and 1024 respectively. To run the `pa2-busy-std` program with 4 threads, for 100 rounds, with a data size of 2048 you would use the command '`./pa2-busy-std 4 100 2048`'.

3 Instructions

1. Implement `try_lock` for Peterson's Filter Lock and the Bakery Lock.
2. Implement the Compare-and-Swap (`https://en.wikipedia.org/wiki/Compare-and-swap`), Test-and-Set (pg. 144), Test-and-Test-and-Set (pg. 145), Backoff (pg. 147), and MCSLock (pg. 154). You should use the functions in `pa2_atomics.h` to implement the first three locks.
3. Use the `try_lock`, `unlock`, and `yield` functions to synchronize `pa2-coop.cpp`.
4. Enter the `build/release` directory and compile the executables. Next, benchmark each of the programs (except the `pa2_*_dummy` programs) with the following parameters:
   (a) A fixed round count of 100
   (b) Data sizes 2^{10} (1024), 2^{18} (262144), and 2^{20} (1048576)
   (c) Number of threads ranging from 2 to 128, by powers of two

   When benchmarking, run each program 6 times, drop the first time, and average the remaining times. Once you have collected your data create three line graphs for each synchronization style, one for each data size, comparing the total elapsed time versus the number of threads. Lastly, create a third set of three graphs using the data for the top three busy-locking strategies and top three cooperative-locking strategies.

4 Submitting the Assignment

To submit the assignment rename the directory `pa2-src` to `<username>-pa2-submission`, remove the `<username>-pa2-submission/build` directory, and compress the directory (with your graph files included) into an archive. Next, upload the archive file as your submission on the assignment's page on Canvas.