CS 758: Programming Multicore Processors (Fall 2016)

Homework 6

You should do this assignment alone. No late assignments.

Filelist for the assignment:

- Yoo, et al. SC13
- TSX Web Resources
- C++ Reference for TM -- DRAFT
- Transactional Memory in C++ Slides
- Performance Pathologies in HTM

The purpose of this assignment is for you to familiarize yourself with Transactional Memory extensions to C++ and to apply them to the concurrent queues of Homework 5.

The hardware

You will be using the machine named adsl-sl01.cs.wisc.edu. This is a single socket Skylake system with four cores and 8 threads. Usage procedure is detailed on piazza!

The software

You should use GCC 6.1 for this assignment.

Step 2: Concurrent queues with Transactional Memory

For this problem, you will use C++ transactional memory support to implement a circular fixed-sized queue. As we did last time, do this in stages. You should make sure your implementation works with `synchronized` blocks before trying `atomic` blocks.

- Develop a transactional memory circular queue implementation which works with a single producer and a single consumer (TM1).
- Develop a transactional memory circular queue implementation which works with a single producer and multiple consumers (TM2).
- Develop a transactional memory circular queue implementation which works with multiple producers and a single consumer (TM3).
- Develop a transactional memory circular queue implementation which works with multiple producers and multiple consumers (TM4).

TM1-4 could be the same implementation. As you are working, try to understand how the constraints on the queue---fixed size and circular---help your transactional memory implementations. Also, think about how the transactional memory hardware works when trying to understand how it will perform.

For simplicity, you may assume that the number of producer or consumer threads are always in powers of two, and that the total number of operations is always divisible exactly by the number of producers/consumers.

1. Describe your implementations for each of the above queues.
2. Describe any tricky scenarios you encountered while writing the transactional implementations.

3. Provide a couple of sentences in the final writeup describing how the transactional implementation was harder or easier than the lock-free implementation.

Problem 3: Analysis of Fixed Size Queues For this analysis, set the number of operations as 1000000 (1 zero less than HW5), queue size as 1000, and run each experiment on adsl-sl01 nodes for 3 iterations. For each graph, make sure the axes are named correctly, have title, captions and legends as needed, and in general, are self-sufficient. Use only std::memory_order_seq_cst for this portion.

- Compare the throughput of all 5 queue implementations (MX - provided, FG, LF1, LF2, LF3, TM1, TM2, TM3) for a single producer, single consumer scenario.
- Compare the throughput of MX, FG, LF2, TM2 for a single producer, multiple consumer case with number of consumer threads = [1, 2, 4, 8]
- Compare the throughput of MX, FG, LF3, TM3 for a multiple producer, single consumer case with number of producer threads = [1, 2, 4, 8]
- Compare the throughput of MX, FG, TM4 for a multiple producer, multiple consumer case with number of producers = number of consumers = [1, 2, 4, 8]

Remember to rerun all of your HW5 implementations on the new machine for accurate comparison!

Tips and Tricks

- **Start early.**

What to Hand In

Please turn this homework in on **paper** at the beginning of lecture. You must include:

- A brief writeup of what you tried, what worked and what didn't
- Answers and supporting graphs for the questions above that you were able to address.

Important: Include your name on EVERY page.
