Programming Paradigms for Concurrency

Homework 4

December 6, 2010; Due Anytime before the course grading

Homework submissions: by email to the instructor.

The goal of this homework is to train you in the various possibilities of implementing a concurrent data structure, and understand the trade-off in each case. You shall implement a hash-table that contains integers, and supports the following operations: add, remove, find. You are supposed to execute the following steps one by one, and record the times taken for writing the code and debugging for steps 1, 3, 4, 6. A rough guideline is that the time for writing the code is up to the time when your code compiles. The time from that point on is debugging time. I would suggest you do this in Java for the ease of spawning threads and good support for STM.

- **Step 1.** Get the basic sequential hash-table ready and test it.

- **Step 2.** Build the infrastructure to allow more threads to access the hashtable. This step is separated just because it is useful for all following steps.

- **Step 3.** Use a coarse-grained locking policy (at the level of the whole hash-table) to parallelize the sequential code.

- **Step 4.** Use a finer-grained locking policy (possibly at bucket-level) to improve the efficiency of the above code.

- **Step 5.** Now, you shall use transactions instead of explicit locks. I suggest you do this using DeuceSTM (available at http://www.deucestm.org). Learn how to run sample transactional programs using DeuceSTM. You can look at the BankAccount code online available in Lecture 1.

- **Step 6.** Use transactions for managing the concurrency.

- **Step 7.** Create a table that compares the throughput for coarse-grained, fine-grained, and transactional hash tables for 4 and 8 threads. Also, put the respective coding and debugging times (steps 3, 4, 6) in the table.

The grade of this course depends only on the level of completion of the homework, and not on any other factors.