CS 6346: Introduction to Multicore Programming
Section 001
Programming Assignment 2

Instructor: Neeraj Mittal

Assigned on: Monday, March 4 2024
Due date: Monday, April 1, 2024 (at midnight)

This is a group assignment. A group can consist of at most two students. Code sharing among groups is strictly prohibited and will result in disciplinary action being taken. You can write the program in either C, C++ or Java.

1 Project Description

To the best of your ability, implement concurrent versions of the following fundamental data structures:

(a) Linked List

(b) Stack

(c) (Unbalanced) Binary Search Tree

Linked List: A linked list stores a set of *ordered* keys. Each key in the list is *unique* (no duplicates allowed). For convenience, you can assume that a key is an integer. You can also assume that keys are stored in the sorted (increasing) order in the linked list. A linked list supports the following three operations, all of which take a key k as an argument:

(1) search(k), which returns true if k is present in the list and false otherwise.

(2) add(k), which inserts the key in the list if not already present, and returns true if it modified the list and false otherwise.

(3) remove(k), which deletes the key from the list if present, and returns true if it modified the list and false otherwise.

Stack: A stack stores a set of (possibly unordered) items. For convenience, you can assume that an item is an integer. A stack supports the following two operations:

(1) push(x), which inserts the item x into the stack (and has no return value).

(2) pop( ), which removes the *last* item inserted into the queue and returns it. If the stack is empty, it returns the special value $\perp$.

Use linked list based implementation of a stack.

Binary Search Tree (BST): A BST stores a set of *ordered* keys using a binary search tree. As in linked list, each key in the tree is unique. It supports the same set of operations as a linked list.

Experimental Evaluation: Conduct experiments to evaluate the performance (system throughput) of your implementations when thread count is equal to the number of logical cores in the machine. In addition, for linked list and binary search tree, choose two different key space sizes of 10^2 and 10^4 keys and two different workloads:

(i) *read-dominated:* 90% search, 5% add and 5% remove

(ii) *write-dominated:* 0% search, 50% add and 50% remove

Measure the throughput of your system after all threads have collectively performed at least 100,000 (or more) operations to get reliable performance data.

Remarks: You can use locks (*e.g.*, mutex, read-write lock and/or try lock) as well as read-modify-write instructions (*e.g.*, test-and-set, fetch-and-add, compare-and-swap, swap, *etc.*) to design a concurrent variant of a data structure.

2 Submission Information

You will have to submit your project using eLearning. Submit all the source files necessary to compile the program and run it. Also, submit a README file that contains instructions to compile and run your program. Finally, submit a report describing your experimental results with properly labeled graphs.