2013.1 Advanced Multicore Computing,

Project #2 : Performance Comparison of Concurrent Linked List Methods

(Due : May 28th 11:59pm)

Submission Rule

1. Create a directory "proj2"
2. Insert into the "proj2" directory: (i)**compilable** JAVA source code, (ii)a document that reports the parallel performance of your code, and (iii) readme.txt into the directory "proj2". The document reports the parallel performance of concurrent linked list methods (coarse grained locking, fine grained locking, and lazy locking) - See below for details of what should be contained in the document. In readme.txt file, you should briefly explain how to compile and execute the source codes you submit. You should use JAVA language. (but C language using pthread library is also allowed.)
4. zip the directory "proj2" into "proj2.zip" and submit the zip file into eClass homework board.

※ The experimentation should be done **in a quad-core CPU computer** which is available in our 4th floor laboratory and etc.

In our class, we studied three main methods for Concurrent Linked List that allows multiple threads to do **add(), remove(), and contain()** operation on the same linked list concurrently. The three methods are coarse grained locking, fine grained locking, and lazy locking. **add()**, **remove()**, and **contain()** are the operations of Set interface. The goal of this project is to implement the three methods using JAVA language, and test/compare/analyze their performance as experimental results.

For experimentation, you should do following.

(i) Implement JAVA concurrent linked list classes: **CoarseList**, **FineList**, and **LazyList** using coarse grained locking, fine grained locking, and lazy locking, respectively. You may use the JAVA source code that is availabe in our lecture note or web : "http://www.elsevierdirect.com/v2/companion.jsp?ISBN=9780123705914" (Chapter 09). Because those source codes may contain some minor errors, you should fix the errors during your implementation. (You may assume that each node of the linked list contains only an integer key value as data of the node.)

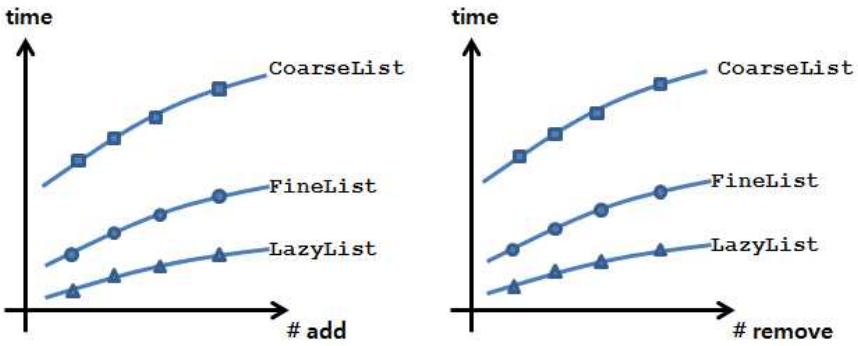
(ii) Write a main JAVA program that uses above classes (you implement) and does following.

1. generate 50,000 random numbers between 1 and 50,000 and put the numbers into an array **A[]**.
2. get user keyboard input : integer number **N** that indicates the number of threads to create.
3. create **N** threads
4. each of **N** threads repeat following work concurrently until no number is left in the array **A[]**
   (1) take out a number **x** from **A[]** one by one (done concurrently)
   (2) add(x) into **CoarseList**
   Totally, 50,000 numbers are added into **CoarseList.**
5. each of **N** threads repeat following work concurrently until **CoarseList** becomes empty
   (1) take out a number **y** from **A[]** one by one (done concurrently)
   (2) **remove(y)** from **CoarseList**
6. measure and print the time for 5,000, 10,000, 15,000, ... , 50,000 **add(x)** of the loop line 4.
   measure and print the time for 5,000, 10,000, 15,000, ... , 50,000 **remove(y)** of loop line 5.
7. Do the same process of line 1-6 for **FineList** and **LazyList**, too.
8. Test above code for thread number **N** = 1, 16, 64, 128

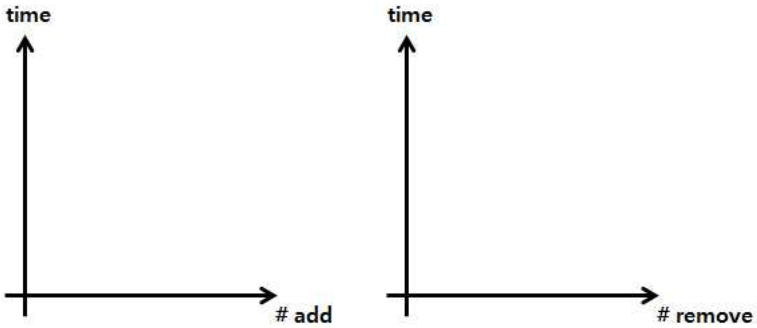
※ Your JAVA code should be compilable and executable.

(iv) draw graphs that show the results of (ii)-8. (You may use MS-Excel for drawing graphs)
Your graphs shoud look like following.

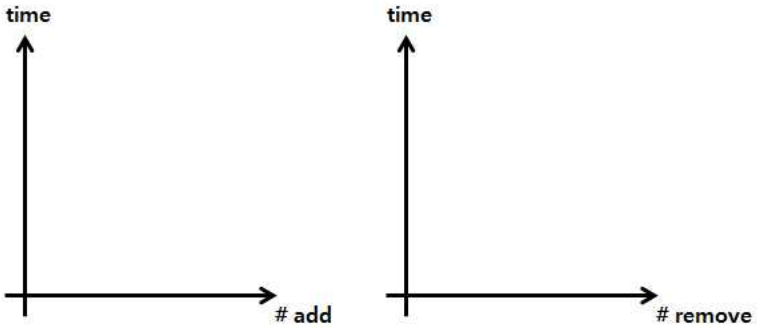
(v) You should write and submit a document that reports the parallel performance of the concurrent linked list methods (coarse grained locking, fine grained locking, and lazy locking) you implemented. The document should contain (a) execution environment, (b) timing result tables(time measurement), (c) following 8 graphs, (d) explanation/analysis on the result and the reason for such result obtained, and (e) complete JAVA source code you made.



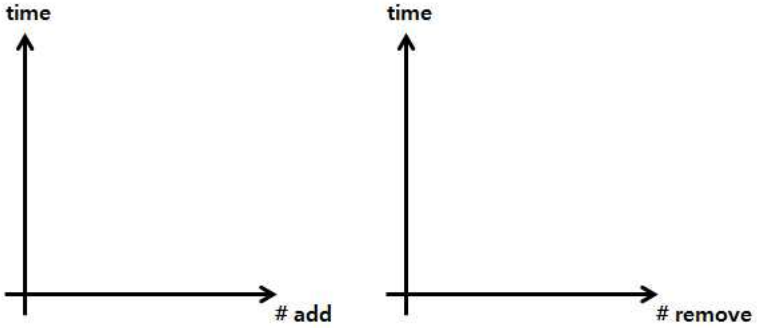
Thread number N = 16



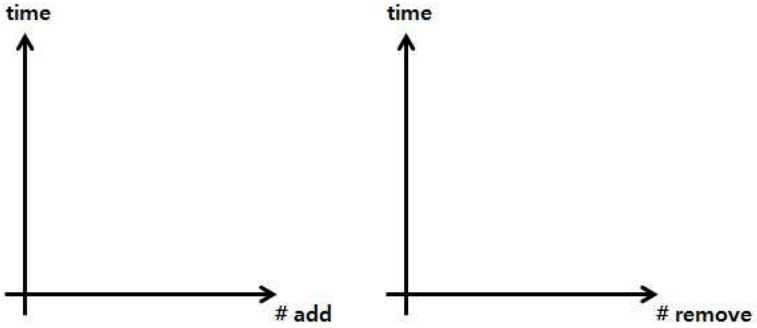
Thread number N = 1



Thread number N = 16



Thread number N = 64



Thread number N = 128