Programming Assignment 5

Comparison of Optimistic and Lazy Synchronization in Sets implemented using Linked Lists

Submission Date: 25th December 2020, 9:00 pm

Goal: The goal of this assignment is to implement Concurrent Sets based on Linked Lists using Optimistic and Lazy Synchronization as discussed in class. Then compare the performance of these two parallel implementation of Linked List.

Details: As explained above you need to implement the two variants of sets based on linked list (a) Optimistic Synchronization and (b) Lazy Synchronization.You have to implement them in C++. Each algorithm implements three methods: add, remove and contains. To test the performance of the algorithm you should create n threads once the program starts. Each thread will perform k operation concurrently with other threads. For every operation each thread randomly select from one of the three functions and perform the function randomly.

Listing 1: main thread

```
1   void main()
2   {
3       ...
4       Initialize the linked list.
5       ...
6       create n test threads
7   }
```

Listing 2: testCS thread

```
1   void test()
2   {
3       id = thread.getid();
4       for(i=0;i<k;i++)
5       {
6           op=rand(add,remove,contains); \\ choose an operation randomly.
7           number = rand(k*N); \\ choose a number randomly to perform the operation.
8           starttime = getsystime()
9           result = op(number);
10          endtime = getsystime()
11          cout<<``thread"<< id << ``performs" << op << ``with number" << number <<
12          ``in round "<< k << ``with results" << result << ``at" <<endtime <<endl;
13          sleep(t1);
14      }
15  }
```

Description of the Test Function: As can be seen each thread invokes the test function. Where a thread randomly selects one of the operation from add, remove, contains. Once the thread chooses the operation

it records the time as starttime. After recording the time thread performs the operation. After performing the operation thread records the time as endtime.

After executing this operation, the thread sleeps for a time $t1$ that is exponentially distributed with an average of $\lambda 1$. Each thread repeats thse steps for k times.

Input: The input to the program will be a file, named inpparams.txt, consisting of all the parameters described above: $n, k, \lambda 1$. A sample input file is: 100 100 5.

Output: Your program should output to a file in the format given in the pseudocode for each algorithm.

Thread 1 performs add with number 10 with result True at 10:00.
Thread 2 performs remove with number 5 with result False at 10:01.
Thread 2 performs contains with number 3 with result False at 10:02.
...
...
...

Report: You have to submit a report for this assignment. This report should contain the comparison of both the algorithms. You must run both these algorithms multiple times to compare the performances and display the result in form of a graph. You must average each point in the graph plot by 5 times. You run both these algorithms varying the number of threads from 2 to 64 in the powers of 2 while keeping other parameters same. Please have k, the number of operation requests by each thread, fixed to 10 in all these experiments.
The graph in the report will be as follows for each algorithm:

- the x-axis will vary the number of threads from 2 to 64 in the powers of 2 (as explained above).
- Y-axis will show the average time taken to finish an operation for each algorithm.

There will be two curves corresponding to each algorithm. Finally, you must also give an analysis of the results while explaining any anomalies observed in the report.

Deliverables: You have to submit the following:

- The source files containing the actual program to execute. Please name them as: ProgAssn5-<rollno >.cpp.
- A readme.txt that explains how to execute the program.
- The report as explained above.

Zip the two files and name it as ProgAssn5-<rollno>.zip. Then upload it on the google classroom page of this course. Submit it by the above mentioned deadline.