CS417 Parallel Processing
Assignment 01: Parallel BST
Due : 23h59 Tuesday 3rd Octobre, 2023.

FCSE
GIKI

Fall 2023

1 Objectives

(a) Test pthreads multithreaded synchronization.

(b) Test parallel implementation of BST.

(c) Test ability to write clean readable code.

(d) Test ability do error handling in code.

(e) Test ability to follow written instructions.

You will be using the C programming language and the pthread library to do this assignment.

2 Parallel Binary Search Tree (BST)

Write a program that implements a BST which can be accessed by multiple threads simultaneously for searching, insertion, and deletion.

You will proceed in 3 steps:

2.1 Q1. Write Binary Search Tree (BST) with one global lock.

Here, you'll have one global lock that'll protect your tree access. Any thread that wants to access the tree, whether for reading or writing, will first acquire this lock, proceed with its operation (reading or writing) and, one done, will release this lock.

Only one thread will be accessing the tree in this scheme.

2.2 Q2. Write Binary Search Tree (BST) with one lock per node.

Here, you'll have one lock for each node of the tree (make the lock a part of your node struct). Any thread accessing the tree would start by acquiring the lock for the root node of the tree, then as it moves downward in the tree hierarchy, it will acquire the locks for the children nodes that it is interested in and release the locks for the parent nodes. Those nodes whose lock have been released can now be accessed by other threads.

In this scheme, multiple threads can access different parts of the tree at the same time. A thread will hold locks for only those nodes which it is susceptible to modify or which it wants to protect from modification by other threads.

Since multiple threads can access the tree simultaneously, albeit at different branches, this program ideally should have more operations in parallel compared to the first. See if there's any performance improvement.

Hint: you might need to keep track of parent and stuff when doing operations. I'll let you figure out how best to implement it.

2.3 Q3. Write Binary Search Tree (BST) with reader-writer locks.

In this implementation we'll distinguish our threads operations into read and write operations:

- search is a read operation as it does not need to modify the tree,
- insert and delete are write operations.

Read operations will dominate our program (more than 99%).

Your job is to provide an implementation which maximizes the parallelism by allowing multiple readers to be able to read parts of the tree simultaneously:

- If a read operation is happening in a part of the tree, other read operations are allowed in that tree, but no write operation can proceed there.
- if a write operation is happening in a part of the tree, no other operation can happen there whether read or write.

Try to maximize the parallelism in your implementation.

2.4 Testing.

Once you've done an implementation, you can test it by running multiple threads that access that BST concurrently.

I've provided a sample implementation that works with 1 thread. To test with multiple threads with large numeber operations you can comment the line 147 in `bst.c`. Currently it crashes . Your job is to make it work with multiple threads.

You can run 1, 2, 4, and 8 threads to test. They should do 800,000 operations altogether. For example, with 8 threads that would make it 100,000 operations per thread.

2.5 Help.

Section 4.9 of your course book [Pacheco] does exactly these same implementations for a linked list. You can start with that as an inspiration.

If you feel you are missing on the background on threads and synchronization, you can read up either the chapter 4 of course book [Pacheco] or for a gentler introduction, read the chapters 26-32 of your OS book (https://pages.cs.wisc.edu/ remzi/OSTEP/).

3 Error Checking and Clean Code instructions

When working with Linux System calls, it is extremely important that you always check the return values of the system calls to verify whether the call was successful or not. The function documentation would usually contain all the information you will need. You should always read the function documentation.

Your programs should guard against invalid user input and in case of an invalid input, it should print a helpful error message.

You would lose marks if your program crashes during use.

It is the programmer's responsibility to free any system resources i.e. memory , file descriptors, etc., they have acquired from the system. Your program should always free system resources once it is done using them.

Code should be properly indented, readable and commented.

You should always your compile your code using the `-Wall` option to check for warnings.

When displaying their valid output on the screen your programs should write to `stdout` and when displaying error messages, they should write to `stderr`.

4 Submission Instructions

1. Submission will be on MS Teams.
2. You submission should consist of two .c files only. Do not compress.
3. You can name your submission as u2020xxx_a1q1.c, u2020xxx_a1q2.c and u2020xxx_a1q3.c where u2020xxx is your registration number.
4. Missing submission deadline on MS Teams will cost you 40 marks. Submissions received more than 24 hours after submission deadline will get a 0.

5 Rubric

This is an individual assignment. Any form of collaboration, cheating, plagiarism will get you a 0. Giving your code to somebody else, even if it is for their understanding only, is not allowed.

You may be called for a viva; if you are unable to explain **any line** of your submitted code, you'll get a 0 even if it's working perfectly (we live in the age of chatgpt!).

Category	Marks
Q1 working properly	20 marks
Q2 working properly	40 marks
Q3 working properly	40 marks
Missing deadline	-40 marks
Not following instructions	upto -100 marks
Max marks	100 marks