COP 4520 Spring 2020

Project Assignment 2

Notes:

Please, submit your work via Webcourses.

Submissions by e-mail will not be accepted.

Due date: Wednesday, April 1st by 11:59 PM

Late submissions are not accepted.

This is a team assignment.

You can use a programming language of your choice for this assignment.

If you do not have a preference for a programming language, I would recommend C++.

Grading policy:

General program design and correctness: 50%

Efficiency: 30%

Documentation including statements and proof of correctness, efficiency, and experimental evaluation: 20%

Additional Instructions:

Cheating in any form will not be tolerated.

In addition to being parallel, your design should also be efficient in terms of execution time and memory use.

The goal of this assignment is to re-implement the core functionality of the data structure described in your main topic paper with its key properties (progress – lock-free, wait-free and correctness – linearizability). This re-implementation will be a reflection of your interpretation of the described algorithms and does not have to be a perfect match of these algorithms. Feel free to use alternative synchronization techniques that you have learnt in our lectures such as descriptor objects.

Here are the required steps for Project Assignment 2:

1. Read the main topic paper again and this time and focus on the key synchronization techniques used and the concurrency properties of the data structure (progress and correctness).

2. Re-implement the data structure with its key multi-threading properties in terms of progress, correctness, efficiency, and thread-safety. The goal is to implement your version of the algorithms described in the main paper. You are allowed to modify the described algorithms as you see fit as long as your solution is multi-threaded and provides the same progress and correctness guarantees as the original approach.

Do not worry about garbage collection and make sure to pre-allocate all elements that you might need in your tests.

3. Run performance tests and compare the execution time of this version of the data structure with the MRLOCK-based version from Project Assignment 1.

In your benchmark tests, vary the number of threads from 1 to 32 and produce graphs where you map the total execution time on the y-axis and the number of threads on the x-axis. Produce at least 3 different graphs representing different operation mixes. In your benchmark tests, the total execution time should begin prior to spawning threads and end after all threads complete. All nodes and random bits should be generated before the total execution time begins (if necessary). Provide a ReadMe file with instructions explaining how to compile and run your program.

4. Write a brief report that will include: a) a quick summary of the data structure and its key methods and properties, b) a discussion on the technical details of your implementation including the use of non-blocking synchronization techniques and the key algorithms of your implementation, c) a section on the performance analysis of the results from step 3 above, and d) a list of references including a citation to the MRLOCK algorithm, your main topic paper, and other relevant approaches.

5. Publish your report according to the publication instructions and submit your work via Webcourses.