NOVA SCHOOL OF SCIENCE & TECHNOLOGY
COMPUTER SCIENCE DEPARTMENT

Concurrency and Parallelism 2022-23

Lab 08: Multiple Synchronisation Strategies in a Concurrent Hash Map

Hervé Paulino & Alex Davidson

May 2023

Abstract

A continuation of last week's assignment, with an additional exercise for implementing a lock-free hash map.

1 Introduction

In this assignment, you will complete last week's assignment, which can be retrieved using the same link as before:

`https://classroom.github.com/a/BkraTcnd`.

The PDF for last week's assignment is available here.

There will be some additional exercises introduced this week, which is detailed below.

2 Lab Work

2.1 Complete last week's assignment

Firstly, complete each of the locking strategies for last week's assignment. You should also complete the *optional* exercise by creating a hash map solution that implements a **fine-grained locking strategy**, namely **hand-over-hand, optimistic, and/or lazy sychronisation**, to control the concurrent accesses to the hash map collision lists.

2.2 Challenge: Lock-free strategy

Implement a concurrent hash map using a *lock-free* design strategy. You can use Section 9.8 of [1] to learn about lock-free implementation strategies. A link to this book is available on CLIP.

3 Evaluation

You can now evaluate each of the strategies that were implemented using the same evaluation strategies that were given last week, but can also include evaluation of the fine-grained and lock-free strategies implemented this week. The evaluation can be done using the following method.

3.1 Effectiveness of each locking strategy

Run tests to evaluate the correctness of each of your solutions from above.

If and only if your implementations prove correct, then run more benchmarks and performance tests to evaluate the scalability of the application when you increase the number of threads. Run experiments with $1, 2, 4, , N$ threads, where N is double the number of *hardware threads* you have available in your computer.

NOTE: if you are using some kind of virtualised environment, make sure that you have more than one processor assigned to your virtual execution environment. Otherwise, you will have no speed-ups with more than one thread.

3.2 Acquire a deeper understand of your Java program

Install the `visualvm` (`https://visualvm.github.io/features.html`) tool with the following command (or download from the link provided):

```
apt install visualvm
```

and then use these tools to analyse the multiple versions of the concurrent hash map.

3.3 To Think More About...

The sequential (unprotected) version is much faster than any lock-based solution. Will any of your solutions ever exhibit a speed-up instead of a slowdown? How many processors would you need for that? How does the lock-free design compare with the other approaches?

References

[1] Maurice Herlihy and Nir Shavit. *The Art Of Multiprocessor Programming*. Morgan Kaufmann, 2008.