UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Department of Computer Science
Faculty of Engineering, Built Environment & IT
University of Pretoria

COS226 - Concurrent systems

Practical 7 Specifications -

Release date: 21-10-2024 at 06:00

Due date: 25-10-2024 at 23:59

(Submission will remain open until 27-10-2024 23:59 for technical difficulties)

Total marks: 30

Contents

1 General Instructions

- *Read the entire assignment thoroughly before you start coding.*
- This assignment should be completed individually; no group effort is allowed.
- **To prevent plagiarism, every submission will be inspected with the help of dedicated software.**
- Be ready to upload your assignment well before the deadline, as no extension will be granted.
- If your code does not compile, you will be awarded a mark of zero. Only the output of your program will be considered for marks, but your code may be inspected for the presence or absence of certain prescribed features.
- If your code experiences a runtime error, you will be awarded a zero mark. Runtime errors are considered unsafe programming.
- Read the entire specification before you start coding.
- **Ensure your code compiles with Java 8**
- The usage of ChatGPT and other AI-Related software is strictly forbidden and will be considered as plagiarism.

2 Plagiarism

The Department of Computer Science considers plagiarism a serious offence. Disciplinary action will be taken against students who commit plagiarism. Plagiarism includes copying someone else's work without consent, copying a friend's work (even with consent), and copying material (such as text or program code) from the Internet. Copying will not be tolerated in this course. For a formal definition of plagiarism, the student is referred to http://www.library.up.ac.za/plagiarism/index.htm (from the main page of the University of Pretoria site, follow the Library quick link, and then choose the Plagiarism option under the Services menu). **If you have any form of question regarding this, please ask one of the lecturers to avoid any misunderstanding.** Also note that the OOP principle of code reuse does not mean that you should copy and adapt code to suit your solution.

3 Outcomes

On completion of this practical, you will have gained experience with the following:

- Concurrent Stacks: Lock Free, Elimination Backoff.

4 Introduction

4.1 Introduction to Concurrent Stacks

In concurrent programming, the design of thread-safe data structures is challenging due to the need to coordinate multiple threads to access shared memory. Stacks often require concurrent versions. Three common approaches are **sequential**, **lock-free**, and **elimination backoff stacks**.

4.1.1 Sequential/Blocking Stacks

A sequential stack operates in a single-threaded environment where only one thread manipulates the stack at a time. This approach is simple but inefficient in a concurrent setting, as threads must take turns, often causing delays and bottlenecks. Synchronization mechanisms such as locks are commonly used to enforce sequential access, but they introduce significant overhead, particularly under heavy contention.

4.1.2 Lock-Free Stacks

Lock-free stacks aim to eliminate the need for mutual exclusion, allowing multiple threads to perform operations without acquiring locks. These stacks ensure that some thread will always make progress even in the presence of contention. Lock-free stacks typically use atomic primitives such to update shared data safely, reducing latency but adding complexity in managing race conditions.

4.1.3 Elimination Backoff Stacks

Elimination backoff stacks enhance performance by allowing threads to "pair up" and exchange values directly, bypassing the shared stack entirely. When contention is high, threads attempt to eliminate each other's operations, reducing contention on the central stack. This technique utilizes both lock-free mechanisms and backoff strategies, achieving scalability and improved throughput under high loads.

5 Tasks

You are tasked with implementing the three stack classes explained above. Using your implementations, you should evaluate how true the following statement is in practice:

> At low levels of contention, the performance of EliminationBackoffStack is comparable to LockFreeStack. However, at high contention the number of successful eliminations will increase, allowing more operations to be completed in parallel

5.1 Task 1 - Stack implementation

Implement the above-mentioned "Stack" classes:

1. Sequential/Blocking Stack: Implement a basic stack where all operations (push and pop) are synchronized using locks to ensure thread safety.

2. Lock-Free Stack: Implement a stack that uses atomic operations (e.g., Compare-And-Swap) instead of locks to achieve thread-safe concurrency without blocking.
3. Elimination Backoff Stack: Enhance the lock-free stack by implementing an elimination backoff mechanism to reduce contention during high concurrency. This allows threads to bypass the stack and directly exchange values when conflicts occur.

5.2 Task 2 - Experiment

Use your implementations to measure and record any relevant metrics over any relevant parameters (e.g. time taken vs number of threads; average time spent waiting vs number of method calls; etc.) Use the `Sequential/Blocking` stack for reference metrics. Hint: A sample size of 1 is not recommended.

5.3 Task 3 - Report

Write a short report which includes the following:

- Your name and student number
- An introduction
- Graphs showing your findings
- Explanations of the trends in your findings
- Reasons for the trends. (E.g. X is better than Y because of Z)
- A conclusions as to how true the statement above is.
- References to any resources you used

6 Mark Breakdown

- Sequential/Blocking Stack - 5 marks
- Lock Free Stack - 5 marks
- Elimination Backoff Stack - 5 marks
- Experiments & Report- 15 marks
- **Total - 30 marks**

7 Upload checklist

Upload all the files, including your report, that you need for your demo in a single archive.
NB: Submit to the ClickUp Module, there will be no FF submission.

8 Allowed libraries

These libraries will be allowed but you must still do the given tasks with your own code (i.e you may not use the libraries to complete the task for you)

- `import java.util.*`

9 Submission

You need to submit your source files on the ClickUp module website. Place all the necessary files in a zip named uXXXXXXXX.zip where XXXXXXXX is your student number. Your code must be able to be compiled with the Java 8 standard.

Upload your archive to the appropriate practical on the ClickUp website well before the deadline. **No late submissions will be accepted!**