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 5 Specifications - Practical Locks

Release date: 30-09-2024 at 06:00
Due date: 04-10-2024 at 23:59
(Submission will remain open until 06-10-2024 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:

- Practical Locks: TAS, TTAS and Exponential Backoff

4 Introduction

4.1 Practical Locks

In concurrent programming, locks such as Test-And-Set (TAS), Test-And-Test-And-Set (TTAS), and exponential backoff are essential for managing access to shared resources among multiple threads. The Test-And-Set lock is straightforward: a thread repeatedly tries to set a shared lock variable until it acquires the lock. This simplicity makes TAS easy to implement, but it can result in high cache coherence traffic since each thread continually modifies the shared variable. The Test-And-Test-And-Set (TTAS) lock aims to alleviate this issue by first checking if the lock is free before attempting to set it. This extra check reduces unnecessary updates, potentially improving performance in some scenarios. However, both TAS and TTAS can still struggle when multiple threads contend for the lock, leading to excessive spinning and increased overhead.

Exponential backoff introduces a different approach by incorporating a waiting period before retrying to acquire a lock. When a thread fails to obtain the lock, it waits for a random interval and doubles this waiting time on subsequent failures. This method helps reduce contention by spacing out lock attempts, allowing threads to back off and prevent a collapse in performance during high contention. While exponential backoff can adaptively reduce the burden on shared resources, it also comes with its own trade-offs. The effectiveness of these locks depends on factors such as the number of threads, contention level, and specific workload patterns. Each method has its strengths and weaknesses, which become apparent under different conditions in real-world applications.

5 Tasks

You are tasked with implementing the TAS, TTAS and exponential backoff locks. After implementing them, you must then compare the locks in various scenarios (at least 2). These scenarios must be of your own construction. For example, dequeueing from a shared queue, adding to a shared counter, etc (Be creative!). Your comparison should include a graph of time taken vs number of threads (similar to figure 7.4 in the textbook) and an explanation of the trends observed.

5.1 Task 1 - Lock implementation

Implement the `TAS`, `TTAS` and `exponential backoff` locks.

5.2 Task 2 - Scenarios

Create at least 2 scenarios to test your locks. These scenarios should show the strengths and weaknesses of the locks. Test the locks with different numbers of threads and record the total time taken to complete the scenario. Hint: A sample size of 1 is not recommended.

5.3 Task 3 - Report

Write a (very) short report which including:

- Your name and student number
- A poorly worded introduction (have fun)
- Graphs showing your findings of time vs number of threads (and any others you can think of)
- Explanations of the trends in your findings
- References to any resources you used
- Your favourite programming joke (tutors can award bonus marks if it's good)

6 Mark Breakdown

- TAS Lock - 5 marks
- TTAS Lock - 5 marks
- Exponential Backoff Lock - 5 marks
- Scenarios - 5 marks
- Report - 10 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!**