Concurrent and Distributed Systems: Supervision 2

Lectures covered by the supervision: https://www.cl.cam.ac.uk/teaching/2526/ConcDisSys/

- Lecture 5: Liveness and priority guarantees
- Lecture 6: Concurrency without shared data, composite operations and transactions, and serialisability
- Lecture 7: Isolation vs. Strict Isolation, 2-Phase Locking (2PL), Time Stamp Ordering (TSO), and Optimistic Concurrency Control (OCC)
- Lecture 8a: Durability & crash recovery.
- Lecture 8b: Lock-free programming & transactional memory.

Previous exams.

Supervision questions:

1. Explain the four conditions required for a deadlock to occur. A sentence explaining each condition is sufficient. Explain why missing each condition would prevent a deadlock.
2. Explain the concepts of thread priority and thread affinity. How can they be used to increase performance considering the influence of cache on execution time?
3. For priority inversion: Discuss the problem with priority inversion in Mars rover, what went wrong? - https://cpen432.github.io/resources/P4-mars.pdf
4. Can you apply Banker's algorithm for deadlock avoidance in the example of Mars rover? If yes, how? If no, what else would you use to identify such bug?
5. 2016 Paper 5 Question 8
6. What are shared mutable states in software? In cases without shared mutable states, can there still have concurrency bugs?
7. Several threads split over several processes all write lines of text to one output device (there are multiple threads per process). Characters from different lines must not be interleaved. At the same time, there has to exist a thread in each process that counts the number of characters that process wrote as output). The processes have different priorities. Outline the major software components (such as processes, threads, buffers, shared variables, locks and system calls) involved in this task and how they interact. Identify the points where threads will block and examine whether priority inversion is likely to happen.
    a. Discuss serialisability in this examples.
    b. Can you use any of the below for synchronisation in this example (how, sketch pseudo code):
        i. Transactional memory
        ii. Two-Phase Locking (2PL);
        iii. Timestamp Ordering (TSO); and
        iv. Optimistic Concurrency Control (OCC)
        v. Linked list
        vi. Could enforcing isolation help here?
    c. For the example above, sketch conflicts of bad scheduling of transactions and their effects.
        i. Create examples of a log that can exist for these operations, considering write-ahead logging. How would you apply checkpoints, and Recovery and Rollback in this case?
8. Discuss terms:
    a. Trylock

    b. Cache coherence
    c. Happens-before relation
    d. Fence (barrier) instructions
    e. Reentrant Lock
9. What is the difference between strict and non-strict isolation in a transaction processing system?
10. Can language-level support for concurrent programming make code more adaptable to running on platforms with varying numbers of cores and highly non-uniform memory access time?
11. Can you sketch an algorithm that would guide a migration from a single-threaded to multithreaded software (automatic parallelisation)?
12. Choose a standard problem in software engineering that would profit from parallelisation:
    a. Explain, with pseudo code, how would you parallelise it.
    b. Explain which synchronisation mechanisms you would use and why.
    c. Explain the connection between parallelisation and testing.
    d. Would you organise your development methodology in a different way to support the parallelisation?
13. Summarize in 1-3 sentences Lesson 5.
14. Summarize in 1-3 sentences Lesson 6.
15. Summarize in 1-3 sentences Lesson 7.
16. Summarize in 1-3 sentences Lesson 8.

BONUS:

- If we want to find concurrency bugs, how do we know which lock is intended to protect which shared resource? What if two different locks protect the same shared resource?
- How is transactional memory implemented in software and how in hardware?
- Discuss atomicity violations.
- Give your opinion on how can we find concurrency bugs?
- Modify the bonus example from S1. Create 3 groups of thread priorities (ignore the fact that they are executing the same function). Threads with different priorities cannot overlap. Assign at most 1 thread to a core. Assume shared variable sum0. Each thread running on core 0 needs to update sum0 with its value. Comment on the synchronisation needs. (if you decide to go for this task and face any issues, feel free to contact me).

Save your answers into MS Teams or email them to me. Please use the following naming pattern:
CDS_Supervision_2_Answers_<last name>_<first name>_Michaelmas_2025

Send your answers as a pdf, doc, image, or any other format of a document for which there exists an easily available software to open.

Jasmin JAHIĆ
jj542@cam.ac.uk
https://www.cl.cam.ac.uk/~jj542/