CS425 Fall 2026 – Homework 4

(a.k.a. "The Interview")

Out: Nov 3, 2026. Due: Dec 3, 2026 ***THURSDAY*** *(11:59 pm US Central time.)*

Topics: Lecture 19 onwards

Instructions:

1. Please double-check that the top of this page mentions the correct semester you are taking the course in (otherwise, you are looking at an old version and we will not accept your submitted solutions).
2. Attempt any **8 out of the 10 problems** in this homework. If you attempt more, we will grade only the first 8 solutions that appear in your homework. Each problem has the same grade value (10 points each).
3. Your **solutions must be typed and not handwritten**. Figures (e.g., timelines) and equations (if any) may be hand-drawn (and scanned).
4. Submit PDF only. **Start each problem on a fresh page**.
5. Please submit on Gradescope. [https://www.gradescope.com/] Self-signup using the code on the website.
6. **Important: On Gradescope, please tag each page with the problem number!**
7. Homework is **due on the date and time noted above. No extensions.**
   For DRES students only: once the solutions are posted (typically a few hours after the HW is due), subsequent submissions will get a zero**.**
8. Always justify all your answers, and show your work.
9. Unless otherwise specified in the question, the only resources you can avail of in your HW are the provided course materials (slides, textbooks, etc.) and communication with instructors/TA via the discussion forum.
10. You can discuss lecture concepts and the questions on Piazza and with your friends, but do not discuss solutions or ideas on Piazza

Prologue: *After the success of your epic Saturn mission and your celebrated return to Earth, many top companies and universities wish to interview and hire you. Your goal is to attempt and ace 8 interviews to maximize your employment choices. All characters and storylines are fictitious, and purely intended to keep the reading entertaining; these are not intended to be educational. Any resemblance to persons, places, animals, things, or events, living or dead, past, present, or future, is purely coincidental.*

Problems:

1. Your interviewer from Google's Chubby team suggests a modification to Maekawa's algorithm to make it deadlock free. Inside enter( ) each process P_i sets its state to Wanted, but then instead of multicasting the request to all its voting set (V_i) members, P_i sorts the members of V_i lexicographically (lowest to highest id), and for each member in that order, it sends a request and then waits for a Reply before moving on to the next member. Answer the following questions:
    a. Is this algorithm safe? i.e., does it ensure mutual exclusion?
    b. Is this algorithm deadlock-free?
    c. Does this algorithm avoid starvation? Starvation occurs when there is no deadlock, but some process stays stuck inside enter( ) forever.

2. During your interview at IBM Research Almaden, they tell you that the relational database model was invented there by E. F. Codd, so they love transactions. They give you these two transactions and ask you some questions:
    • T1: write(a, caz); read(b); write(c, foo);
    • T2: write(a, bar); read(b); write(c, baz);
    a. Say they interleave these 6 operations in an alternating fashion. In other words, they start with T1's first operation, then T2's first operation, then T1's second operation, and so on. Is this interleaving serially equivalent? Explain why.
    b. Your interviewer claims *any* interleaving of T1 and T2 is serially equivalent. If you agree, prove it. If not, give a counterexample.
    c. Remove the last operation of each of T1 and T2 (accesses to c) to derive T1' and T2', respectively. How many of *all possible* interleavings of T1' and T2' are serially equivalent and how many are not? (there should be 6 total interleavings)

3. Your next interviewer, Montgomery Burns, proposes an "excellent" transaction management system that uses 2 phase locking in which all necessary locks are acquired serially at the start of the transaction, and are released at the commit point (if a transaction blocks while acquiring a lock, it remains blocked and does not abort). He tells you that the system acquires locks only in lexicographically *increasing* order of the corresponding object's "special field". Burns claims that this system will never deadlock. In each of the following cases, state whether the above system will deadlock (with a counter-example) or not (with a proof).

a. Each object's "special field" is set to access order of the transaction, i.e., a transaction T locks objects in the order that T will first access these objects (note that all locks are acquired at T's start point).
b. Each object's "special field" is the object's ID, which is globally unique.
c. Each object's "special field" is the object's first creation time—UTC time with a random salt/nonce appended. The random salt/nonce is guaranteed to be unique.

4. You then meet Gordon Gekko, who has designed a new greed-is-good transaction system, in which each transaction acquires the lock for any object just in time, immediately before the transaction's first access of that object, and then releases the lock immediately after the transaction's last access of that object.
    a. Does this system satisfy serial equivalence? Provide a proof or counterexample, as appropriate.
    b. Is this scheme deadlock-free? Provide a proof or counterexample, as appropriate.

5. Your next interviewer is the infamous Lex Luthor. To create kryptonite, he needs to build a server with linearizability for client operations (assume a single server with no replication). Lex argues that since writing to a variable is an idempotent operation, on a failure, a client can safely retransmit a failed write operation, and the server can safely re-execute the operation (to write to the variable), and that such a system provides linearizability. Assume you have a server S that stores a single variable x; W(x=a) sets x at the server to value a; R(x) returns the **current** value of x at the server. The following sequence of operations occur at two clients C1 and C2.

    C1: (i) W(x=2)
    C2: (i) R(x); (ii) W(x=3); (iii) R(x)

    Let us say S successfully executes C1's operation W(x=2). However, the acknowledgement message from S to C1 is lost. So now, C1 retransmits that W(x=2) operation, and S re-executes it.
    a. Can you provide an interleaving of operations from C1 and C2 (including the re-execution of C1's write) that violates linearizability?
    (Hint: C1's sequence of operations now looks like this:
    (i) W(x=2); (ii) re-execute W(x=2))
    b. Are all interleaving of operations from C1 and C2 (including the re-execution of C1's write) sequentially consistent? If not, can you provide a sequence that violates sequential consistency?

6. Norman Osborn, the founder of Oscorp, shows you a database system he's designed to track Spiderman's movements. The database system uses OCC with Timestamp Ordering. Three transactions (shown below) arrive and are assigned start timestamps in order, T1 is assigned 1, T2 is assigned 2, and T3 is assigned 3.

Time	**T1**	**T2**	**T3**
1	Start & Read(A)		
2	Read(B)	Start & Read(B)	
3	Write(A = A+B)	Read(C)	Start & Read(A)
4	Enter Validation		
5	Validation succeeds	Write(B = B + C)	Read(D)
6	Writes Committed		Write(D = D - A)
7		Enter Validation	
8		**Result?**	
9			Enter Validation
10			**Result?**

   (a) Does T2 pass or fail validation at time=7 against T1? Explain your answer.
   (b) Does T3 pass or fail validation at time=9 against T1 (and T2, if it successfully committed)? Explain your answer.

7. Interviewers at Berkeley and Stanford are surprised that you know about their invention DRF. In our lecture, we discussed Dominant Resource Fairness (DRF), but not the equations to derive "fair" allocations. Look at the equations in the original paper, especially Section 4 and 4.1 (only). You can read only this paper for this question: https://courses.engr.illinois.edu/cs425/fa2026/nsdi_drf.pdf. Then, calculate fair allocations for each of the following cases (Cloud has 40 CPUs, 80 GB RAM). How many tasks, CPUs, and how much memory are allocated to each job? Assume that only an integral value of CPUs can be allocated to a task, but memory allocations can be fractional (GB).
   a. Job 1's tasks: 2 CPUs, 2 GB. Job 2's tasks: 4 CPUs, 4 GB.
   b. Job 1's tasks: 4 CPUs, 8 GB. Job 2's tasks: 4 CPUs, 8 GB.
   c. Job 1's tasks: 4 CPUs, 2 GB. Job 2's tasks: 8 CPUs, 8 GB.

8. Your interviewer at MIT asks you this question involving 5 processes P1-P5 in a distributed shared memory system using invalidate. Process P3 wants to write a page X. In each of the following cases, list the series of operations that need to happen for P3 to be able to write. If the conditions in a case appear wrong, you must point out ALL errors in it.
    a. P3 holds the page X in Read mode, P4 holds it in Write mode, and P4 is the owner.
    b. P4 is the owner and holds page X in Write mode.
    c. P1 and P2 each hold page X in Write mode, and P4 is the owner.
    d. P4 and P5 each hold page X in Read mode, and P4 is the owner.

9. (You can use the internet as a resource for this question.) Your interviewer at a high-profile startup in the Bay Area asks you to discuss data processing systems/concepts. Be sure to include URLs/citations you used for answering this question. Please keep this answer to <= 1 page.
    a. List 2 similarities between Deep Neural Nets (DNNs) as in Apache TensorFlow or PyTorch and stream processing DAGs as in Apache Storm.
    b. What is pipeline parallelism? How does it differ from other parallelism strategies, such as model and data parallelism in distributed machine learning?
    c. In large-scale distributed training, failures are the norm. How do practical systems deal with failures that occur during long-running training jobs? Pick any system and discuss its fault-tolerance mechanism.

10. Your interviewer at Meta, asks you to explain client and server caching behavior with NFS. Two NFSv3 clients C1 and C2 access a shared file /mnt/foo/bar from a remote file system exported by an NFS server S. At time t=0, the payload of that file is "ORIGINAL". Here is a timeline of events on C1, C2, and S:

Time(s)	**Actor**	**Action/Event**
t=0	C1, C2	open /mnt/foo/bar. File descriptors on C1 & C2 are fd1 & fd2, respectively.
t=1	C1, C2	read(fd1) & read(fd2) => Each of them reads the payload of "ORIGINAL"
t=2	C1	seeks to offset 0 on fd1 write "CLIENT1_DATA" to fd1

t=4	C1	close(fd1)
t=5	C2	Without closing/reopening fd2, seeks to offset 0, and reads payload of fd2 again.
t=6	C2	close(fd2)
t=37	C2	open /mnt/foo/bar, read payload, & close the file.
t=39	Server	Power failure; S reboots in 10s.
t=70	C2	After S reboots, open /mnt/foo/bar and read its payload.

In each of the 3 following config settings (A), (B), and (C), tell us what C2 reads at times: (1) t=5s, (2) t=37s, and (3) t=70?

(A) Client caching on C2 is enabled with freshness TTL=30s, but server caching on S is disabled (i.e., write-through mode).
(B) Client caching is disabled, but server caching on S is enabled (i.e., delayed writes mode). And, S did not manage to flush its page cache to persistent storage before t=39.
(C) Client caching on C2 is enabled with freshness TTL=30s, and server caching on S is enabled (i.e., delayed writes mode). And, S did not manage to flush its page cache to persistent storage before t=39.

P.S.: *If you're wondering why UIUC is not listed in the above places, it's because we encourage cross-pollination and would like our alumni to spread their knowledge everywhere around the world (true fact!).*

====== END OF HOMEWORK 4 ======