CS425 Fall 2025 – Homework 4

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

Out: Nov 3, 2025. Due: Dec 4, 2025 (2 pm US Central time.)

Topics: Lecture 19 and onwards (RPCs, Concurrency Control, Replication Control, and the rest of the topics)

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 (regardless of how many credits you're taking the course for). If you attempt more, we will grade only the first 8 solutions that appear in your homework (and ignore the rest). Choose wisely!
3. Please hand in **solutions that are typed** (you may use your favorite word processor. We will not accept handwritten solutions. Figures (e.g., timeline questions) and equations (if any) may be drawn by hand (and scanned).
4. **All students (On-campus and Online/Coursera) –** Please submit PDF only! Please submit on Gradescope. [https://www.gradescope.com/]
5. Please **start each problem on a fresh page**, and **type your name at the top of each page**. **And on Gradescope please tag each page with the problem number!**
6. Homeworks will be **due at time and date 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**. All non-DRES students must submit by the deadline time+date.**
7. Each problem has the same grade value as the others (10 points each).
8. Unless otherwise specified in the question, the only resources you can avail of in your HWs are the provided course materials (slides, textbooks, etc.), and communication with instructor/TA via discussion forum and e-mail.
9. You can discuss lecture concepts and the questions on Piazza and with your friends, but you cannot discuss solutions or ideas on Piazza.
10. Like for any assignment, the instructions for this homework are a union (not an intersection) of the instructions included in this PDF and the instructions (includes clarifications) on any pinned Piazza post related to this homework. Please follow instructions on Piazza carefully.

Prologue: After the success of your epic Saturn mission and your celebrated return to Earth, you receive many offers of employment at top companies and universities. But before you get hired, you have to go through interviews at these places. Your goal is to attempt 8 interviews and ace them so that you have the maximum choice of where you want to (After all, if you're attached, the choice will likely be made by your significant other, so you want to give her/him the most choice! If you're not attached, you want to have the most choice anyway, right? ;).

The storylines, statements, events, things, and games in this homework are purely fictitious. Any resemblance to persons, places, or events, living or dead, past, present, or future, is purely coincidental. All actions, words, and thoughts, ascribed to real entities like people and companies, are purely fictitious.

Problems:

1. During an interview at IBM Research Almaden, they tell you that the relational database model was invented by E. F. Codd, so they love transactions. They give you the following transactions
   - T1: write(a, caz, T1); read(b, T1); write(c, foo, T1);
   - T2: write(a, bar, T2); read(b, T2); write(c, baz, T2);

   They ask you
   a. If one interleaves the transaction's statements alternately, starting with T1's first statement, then T2's first statement, then T1's second statement, and so on – is this interleaving serially equivalent? Say 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 statement of each of T1 and T2 (accesses to c) to derive T1' and T2'. How many of *all possible* interleavings of T1' and T2' are serially equivalent and how many are not? (there should be 6 total interleavings)

2. Your next interviewer, Montgomery Burns comes up with an "excellent" scheme. He tells you that they use a transaction management system that uses 2 phase locking where all necessary locks are acquired serially at the beginning of the transaction (if a lock attempt blocks, the transaction blocks and does not abort), and all locks are released at the commit point. He tells you that the system restricts transactions to acquire locks only in lexicographically *increasing* order of an object's "special field". Burns claims that this system will not deadlock. In each of the following cases, say whether the above system will deadlock or not (give a proof or a counter-example).

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 unique ID (UTC time, with a random salt/nonce appended) of the object's first creation time.

3. During one of your interviews you meet Gordon Gekko, who tells you that he has a new transaction system where he has each transaction acquiring a lock for an object just in time, right before the transaction's first access to that object, and then the transaction releases the lock object right after the last access of that transaction to that object.
    a. Will this system satisfy serial equivalence? Give a proof/counterexample (as appropriate).
    b. Can this scheme deadlock? Give a proof/counterexample (as appropriate).

4. Your next interview is with Montgomery Burns, who thinks he can one-up Gekko Gordon and so he comes up with an "Excellent!" scheme. Burns wants to build a server that provides linearizability for client operations (assume there is just a single server with no replication). Burns argues that since writing to a variable is an idempotent operation, on a failure, a client can safely re-transmit the failed write, and the server can safely re-execute the function (to write to the variable) and the system will provide 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. You are given the following sequence of operations at two clients C1 and C2.

    C1: (i) W(x=2)

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

    Suppose the server executes C1's write W(x=2) but the acknowledgment message from server is dropped. So, C1 considers the request failed and then retransmits the write. The server then re-executes the write.
    a. Can you come up with an interleaving of operations from C1 and C2 (including the re-execution of C1's write) that violates linearizability?
    (Hint: C1's sequence with re-execution would look 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?

5. (For this question, you can look up the linked paper.) The folks at Berkeley and Stanford are surprised to know that you know about their invention DRF. In lecture we discussed Dominant Resource Fairness (DRF), but we did not discuss equations to derive "fair" allocations. Look at the equations in the original paper, especially Section 4 and 4.1 (only). Here is the only paper you can access for this question:
   https://courses.engr.illinois.edu/cs425/fa2022/nsdi_drf.pdf.
   Then calculate fair allocations for each of the following cases (Cloud has 40 CPUs, 80 GB RAM). Please provide answers that have only integer values for number of tasks and number of CPUs assigned to each (memory can be fractions of 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.

6. (For this question, you can look up the Web.) While you're interviewing with Cruella De Ville, she is pensive. She is wondering about two recent technologies: RDMA (Remote Direct Memory Access) and CXL 3.0 (Compute Express Link). RDMA enables a physical machine to directly access the memory of another physical machine, while bypassing the CPU on the remote machine. CXL 3.0 allows a small number of physical machines to share a common physical memory module. In less than a total of 100 words, for each of the technologies, say 1) what are the key similarities between what that technology enables and distributed shared memory (DSM) that we learned in the lecture, and 2) what are the key differences between what that technology enables and DSM.

7. (Just to be clear -- You cannot use the Web for this question.) During your exciting interview at MIT, you find that they seem to like distributed shared memory. They ask you the following question involving 5 processes P1-P5 in a distributed shared memory system using invalidate. Process P3 wants to write a page. In each of the following cases, say what is the series of operations that needs to happen for P3 to be able to write (warning: there might be tricks below!). If the setup seems wrong to you, you should point out ALL errors in it.
   a. P3 is holding the page in Read mode and P4 is holding it in Write mode and P4 is the owner
   b. P4 is the owner and is holding the page in a Write mode
   c. P1 and P2 are each holding the page in a Write mode, and P4 is the owner
   d. P4 and P5 are each holding the page in a Read mode, and P4 is the owner

8. You also interview for a research position at Berkeley. At Berkeley, where they invented the Mica Mote (true fact!), they say they are building a sensor network on an African preserve to monitor movements of lions (and ensure that none of them are killed for sport). The sensors measure sound. The deployment spreads 2000 MICA motes over several hundreds of square miles. They would like to periodically (every minute) measure the average sound across all your sensors. You have two options: either having the sensor nodes route all their sound measurements to a base station (routed via other sensor nodes) which in turn then calculates the average sound reading, or have the sensor nodes talk to each other and calculate the average amongst each other. Answer the following questions:
    a. Which of these two options would you choose? Give at least one major reason why you chose that option.
    b. To calculate the average via a spanning tree among the sensor nodes, what data would you pass along (up the tree)? Give precisely the calculation involved for aggregation.
    c. To calculate the maximum via a spanning tree among the sensor nodes, what data would you pass along (up the tree)?

9. (You can use the Web as a resource for this question.) While interviewing at a high-profile startup in the Bay Area, they ask you to tell the key differences between some data processing systems/concepts.
    a. List two similarities between Deep Neural Nets (DNNs) as in Apache TensorFlow or PyTorch and stream processing DAGs in Apache Storm.
    b. What is pipeline parallelism, and 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? You can pick any system of your choice and discuss its fault-tolerance mechanism.

    In your answer, be sure to include URLs/links pointing to specific characteristics, otherwise you may not get points (these don't count in the word limit). Don't write too long, but don't write a short answer either.

10. (Just to be clear -- You cannot use the Web for this question.) Your next interview is at Facebook/Meta. Your friend who works at Facebook, has in her free time, built a new distributed file system (FaceFS). To design FaceFS, she is exploring

some design modifications over existing systems. Can you help her by providing the advantages (if any) and disadvantages (if any) of the modified system compared to the original one?

a. She wants to design FaceFS such that it implements (at its low level) Unix file system read/write-like semantics, i.e., unlike the Vanilla DFS discussed in class, internally FaceFS maintains file descriptor data structures which contain an automated read-write pointer (at the server side).
b. She wants to design FaceFS like NFS but with a twist. FaceFS clients still cache recently accessed blocks. But every time a cache block is accessed, FaceFS only checks if $(Tm_{client} = Tm_{server})$.
c. She wants to design FaceFS like AFS but with a twist. In FaceFS, when a file is opened, instead of fetching and caching the whole file (like in AFS), only the first 4KB chunk of the file is fetched (and cached). Other chunks are fetched (in 4KB granularity) on demand whenever a block in that chunk is accessed. When a file is closed in FaceFS, clients only flush the chunks that were modified to the server (instead of sending the entire file like in AFS).

P.S.: If you're wondering why Illinois/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, and…

BEGINNING OF YOUR DISTRIBUTED SYSTEMS CAREER! ======