Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.5840 Distributed System Engineering: Spring 2025

Exam II

Please write your name on the bottom of each page. You have 120 minutes to complete this exam.

Some questions may be much harder than others. Read them all through first and attack them in the order that allows you to make the most progress. If you find a question ambiguous, write down any assumptions you make. Write neatly. In order to receive full credit you must answer each question as precisely as possible.

You may use class notes, papers, and lab material. You may read them on your laptop, but you are not allowed to use any network. For example, you may not look at web sites, use ChatGPT, or communicate with anyone.

The maximum number points available is 76.

Gradescope E-Mail Address:

Name: ____

Grade statistics for Exam 2

$$\begin{aligned} \text{max} &= 74.5 \\ \text{median} &= 60.5 \\ \mu &= 58.95 \\ \sigma &= 7.22 \end{aligned}$$

Name: ______________________________

I FaRM

Consider the following statements about FaRM as described in *No compromises: distributed transactions with consistency, availability, and performance*. For each statement, circle True or False.

1. [8 points]:

True / False : Short leases are important for FaRM, because FaRM must reconfigure to remove a failed replica of a region before FaRM can continue to use the region.

True / False : For small messages RDMA performs 4× better than RPC because the CPU is a performance bottleneck for RPCs.

True / False : To obtain write locks FaRM uses RDMA writes so that the destination machine's CPU doesn't have to do any work on a lock request.

True / False : For the TATP workload the median latency on the left end of the graph is 9 microseconds (see Figure 7), rather than the 19 microsecond mean commit latency mentioned in Section 6.3, because not all operations update multiple rows.

Answer: 1. True, since FaRM uses primary-backup replication instead of Paxos/Raft and all replicas of a chunk must be up for FaRM to be able to continue using the chunk. 2. True; see Figure 3. False; in this case FaRM uses RDMA to implement RPC, which runs locking code on the destination machine. 4. True; see description of TATP workload and the explanation for the left-end of the graph.

Name: ____________________

II Chardonnay

Answer these questions with reference to *Chardonnay: Fast and General Datacenter Transactions for On-Disk Databases* by Eldeeb *et al.*

2. [3 points]: Section 6.1 says that all of a read-write transaction's writes are tagged with a version containing the same epoch number. For example, if a read-write transaction executes at around the time the Epoch Service changes the epoch from 20 to 21, either all the transaction's updated records should have a VID prefixed with 20, or all should be prefixed with 21, but not a mix of the two. Explain briefly why having the *same* epoch number in all the written items' versions is important for correctness.

Answer: A read-only (snapshot) transaction must see record versions that, for every read-write transaction, either reflect all of that read-write transaction's writes, or none. Chardonnay's strategy is for read-only transactions to read a snapshot as of the end of some epoch. This only works if every read-write transaction's writes appear all in the same epoch, and are not spread over multiple epochs.

3. [3 points]: Section 6.1 describes how a transaction chooses a version ID (VID): it is the current epoch, with a counter appended. Suppose, instead, that version IDs were just the current epoch (with nothing appended). Briefly explain how this change could cause snapshot reads to yield incorrect (non-serializable) values.

Answer: Read-only (snapshot) transactions would not be able to tell which version was the last one in an epoch, and thus might read a mix of versions from early and late in the epoch.

4. [3 points]: Section 8 says that the locks for a transaction are acquired one at a time using a "chain" technique. It would be faster if the client sent out all the lock requests in parallel. Briefly explain why sending a transaction's lock request messages in parallel would be a bad idea.

Answer: The purpose of acquiring the locks one at a time is to ensure that they are acquired in a deadlock-avoiding order. Sending out the requests in parallel would not ensure that, and could lead to deadlock.

Name: ____________________

III Grove

Consider “Grove: a separation-logic library for verifying distributed systems” by Sharma et al. For each statement, circle True or False.

5. [8 points]:

True / False : Because any replica in Grove can serve a Get request, Grove achieves higher throughput for Get’s than for Put’s (see Figure 19), but sacrifices linearizability for Get’s.

True / False : If a response to a Get is lost, the Grove `exactlyonce` library will resend the Get, which the server executes again and the second response may contain a different value than the first (but lost) response contained.

True / False : If the Go code for vKV had a bug that caused a backup to lose the effect of a Put after a crash, then Grove’s specification and proof would catch this bug.

True / False : If the `configservice` had an infinite loop that caused `Reconfigure` to never return, then this bug would be caught by Grove’s specification and proof.

Answer: 1. False; Grove doesn’t sacrifice linearizability for reads from backups. 2. True; for example, if a Put happened between the first and second Get (but it doesn’t matter for correctness). 3. True; that is a safety property that Grove’s specification and proof capture. False; Grove doesn’t prove liveness properties.

Name: ______________________________

IV DynamoDB Transactions

With respect to *Distributed Transactions at Scale in Amazon DynamoDB* by Idziorek *et al.*, imagine that a read-only transaction (TransactGetItems) and a read-write transaction (TransactWriteItems) execute concurrently. The read-write transaction updates multiple items stored on different storage nodes; the read-only transaction reads the same set of items. Serializability requires that the read-only transaction see all of the items as they were before the read-write transaction, or all as they are after the read-write transaction (but not a mix of before and after).

6. [5 points]: For the above scenario, which mechanisms help to ensure this all-before or all-after property? Refer to the design as described in the paper's Sections 2 and 3 (but not Section 4). Circle all that apply.

A. The item.timestamp < input.timestamp check in Listing 3.

B. Multiple time-stamped versions stored for each key.

C. Log sequence numbers (LSNs).

D. Two-phase locking.

E. item.ongoingTransactions

Answer: **C** and **E** are correct. **A** is not correct because read-only transactions do not involve the time-stamps. **B** is not correct because DynamoDB doesn't store multiple versions of a given record. **D** is not correct because read-only transactions don't use two-phase locking.

Name: ______________________________

V AWS Lambda

Consider the paper, and guest lecture about, *On-demand container loading in AWS Lambda* by Brooker et al. For each of the following statements, indicate whether it is true or false.

7. [8 points]:

True / False : AWS Lambda is attractive to customers because the customer can spawn many lambdas in response to a load spike without having to provision machines in advance.

True / False : Replication of chunks in the AZ-level cache is important to ensure that chunks are not lost forever when a cache node fails.

True / False : Erasure coding of cached chunks helps improve tail latency, because a worker can reconstruct a chunk without having to download all stripes of the chunk.

True / False : The convergent encryption scheme described in Section 3.1 helps protect against an attacker who compromises a worker and attempts to read containers of any customer.

Answer: 1. True; customers can spawn many Lambdas in response to a spike in load. 2. False; AWS Lambda replicates for low latency instead of durability. 3. True; erasure coding allows the client to reconstruct the data with a few stripes without having to wait for all stripes. 4. True; the goal is that a worker can access only the data that it needs to run the function sent to it, but because "any" is ambiguous we accepted False too.

Name: ____________________

VI Lab 4

Ben Bitdiddle is implementing the `rsm` package from Lab 4. He uses a shared reply table to implement communication between the `rsm`'s `Submit()` method and the reader goroutine. The reply table maps a completed operation's `id` to its reply from `DoOp()`. In addition, `rsm` contains a `nextId` field that is initialized to 0 and is incremented by 1 each time `Submit()` is called.

His code takes the following steps:

A. The kvraft server calls `rsm.Submit()` with a client request `req`.

B. `Submit()` increments `rsm.nextId`, packs the request into a new operation `Op{id: rsm.nextId, req: req}`, and calls `rsm.rf.Start()` with the `Op`. If `rsm.rf.Start()` indicates that the current peer is the Raft leader, it then waits for the reply table to contain a reply under the key `Op.id` (the one passed to `Start()`).

C. When the `rsm`'s reader goroutine receives an `ApplyMsg` from the `applyCh`, it calls `DoOp()`, then populates the reply table with the result from `DoOp()` under key `Op.id`.

D. When `Submit()` sees the `Op.id` key in the reply table, it deletes the `Op.id` key/value pair from the reply table and returns the corresponding reply.

Assume that all omitted parts of Ben's design are correct.

8. [5 points]: Ben notices that his implementation can result in incorrect behavior. Explain why.

Answer: Each peer will, when it is leader, assign operations IDs starting with ID zero. Thus, if there is a change in leader, the new leader may assign IDs to new operations that are the same as IDs being waited for by `Submit()`s in the old leader, but the operations are different. So clients may receive responses for the wrong operations.

Name:

VII Ray

Consider the following Ray program, which creates a `sqrt_task` task for each number in the list `mylist`. The creation yields a DFut and the caller waits for the tasks to complete by calling `get` on each future. The code is as follows:

```
# A call to sqrt_task yields a DFut
@ray.remote
def sqrt_task(n):
  # sqrt is a python function, which returns the square root of its argument
  return sqrt(n)

@ray.remote
def sum_task(f):
  # sum is a python function, which takes a future and returns the sum
  l = get(f)     # collect the list l
  return sum(l)  # return the sum of the numbers in list l

# A call to sqrt_list_task yields a shared DFut
@ray.remote
def sqrt_list_task(n_list):
  # start tasks and collect futures
  l = [ ]              # list holding DFuts
  for i in n_list:     # iterate over list of numbers
    l.append(sqrt_task(i))

  r = [ ]
  for f in l:
    r.append(get(f))   # collect the result

  return r  # return a SharedDFut for r

# invoke sqrt_list_task with a large list of numbers, sum, and print result
f = sqrt_list_task(mylist)
s = sum_task(f)
print(s)
```

Name: ____________________

Assume Ray behaves in the way described in *Ownership: a distributed futures system for fine-grained tasks* by Wang et al., and Ray is running on a cluster of computers.

For each of the following statements, indicate whether it is true or false.

9. [8 points]:

True / False : a Ray worker may start running `sum_task` before `sqrt_list_task` has finished

True / False : the driver that invokes `sum_task` receives the list with square-rooted numbers from the worker that ran `sqrt_list_task`.

True / False : the driver is the owner for each future that `sqrt_task` returns.

True / False : the driver is the owner for the shared future returned by `sqrt_list_task`.

Answer: 1. True, since remote invocations are asynchronous. 2. False; the worker running `sum_task` will fetch the data from the worker that ran `sqrt_list_task`. 3. False; the worker who runs `sqrt_list_task` is the owner of these futures; 4. True; the driver starts `sqrt_list_task` and is thus the owner.

Name: ____________________

VIII SUNDR

Consider the straw-man design in the paper *Secure Untrusted Data Repository (SUNDR)* by Li *et al*.

In the straw-man design, a client asks the server to append a "fetch" operation to the history after reading the history. Each client also remembers the most recent operation it has appended to the history. One reason for these mechanisms is to prevent the server from first showing some operations to the client, and then later hiding those operations.

Suppose one modified the SUNDR straw-man protcol in three ways. First, eliminate "fetch" operations, so that the history only includes modify operations. Second, have each client remember the last entry in the most recent history the client obtained from the server; call this remembered entry ELAST. Third, each client checks that ELAST is present in the next history it obtains from the server. With these modifications, a client can still detect a situation in which the server first shows an operation to a client, and then omits that operation in a later history shown to the same client.

10. [5 points]: It turns out this modification is a bad idea. Explain briefly how this modification would allow a malicious server to violate fork consistency.

Answer: Suppose the last two entries in the (correct) history are E1 followed by E2. When client C1 asks for the history, the SUNDR server could return the history just through E1, omitting E2. C1 would then remember E1 as its ELAST. This is a fork (and is allowed by fork consistency) because C1 isn't seeing the complete history. When C1 next asks the SUNDR server for the history, the SUNDR server could return the complete history, including E2. C1 would accept this history because ELAST (=E1) is present and all other checks pass (e.g. E1's and E2's signatures over the preceding history will validate). At this point the fork has been healed (since C1 sees the previously concealed E2); this is a violation of fork consistency. Note that every entry (including whatever is in ELAST) has a signature over the entire preceding history, and the client checks all of these signatures every time it obtains a history from the SUNDR server, so the SUNDR server cannot successfully change anything in the history before a client's ELAST.

Name: ______________________________

IX Bitcoin

Bitcoin: A Peer-to-Peer Electronic Cash System, by Nakamoto, mentions in Section 4 that the cryptographic hash of a valid block must start with a certain number of zero bits. Assume that the hash algorithm is SHA-256, which returns a 256-bit hash.

11. [3 points]: You are trying to mine a new block. The required number of zero bits is seven. You set the block's 32-bit nonce field to a random value, and compute the SHA-256 hash of the block. What's the probability that the first seven bits of the hash are zeros? Circle the one best answer.

A. 1/2

B. 1/7

C. 1/128

D. 1/256

E. 1/249

F. $1/(2^{32})$

Answer: **C** (1/128) is correct.

Name: ____

Ben runs a Bitcoin node. A few hours ago Ben's node learned about block B747, which is a valid block. Ben sees a transaction T27 in B747 that pays some money to a certain public key, signed by the correct private key. Ben would like to steal the money involved in T27. He modifies his copy of block B747 so that the payee's public key in T27 is Ben's own public key. He doesn't change anything else in B747. He modifies his Bitcoin node software to announce the block to other nodes as if it were a valid block.

12. [3 points]: Which of the following will cause other Bitcoin nodes to decide that Ben's B747 is invalid? Circle all that apply.

A. The "Prev Hash" field in the next block in the chain doesn't refer to Ben's B747.

B. Other peers will already know about the real B747.

C. The "Prev Hash" field in Ben's B747 isn't valid.

D. The hash of Ben's B747 won't start with enough zeroes.

E. The signature in T27 in Ben's B747 isn't correct.

Answer: **D** and **E** are correct. **A** and **B** are not correct: peers have to at least temporarily accept otherwise-valid blocks with no successor because they might turn out to be the start of a new winning fork. **C** is not correct because Ben didn't modify the Prev Hash field, so it continues to refer to the predecessor of the original B747. **D** is correct because modifying the block will modify its cryptographic hash; the real B747's hash started with enough zeroes, but a modified B747 is fantastically unlikely to happen also to start with enough zeroes. **E** is correct because the signature was correct for T27's original payee public key, so the signature won't be correct with Ben as the payee.

Now Ben is designing a new crypto-currency system, identical to Bitcoin, except with a different agreement scheme to resolve forks in the block-chain: instead of the longest fork winning, nodes compute the hash of the last block in each fork, and the fork with the lowest last-block hash value wins. Ben reasons that all nodes will compute the same hashes, and thus all nodes will agree about which fork wins.

13. [3 points]: Why is Ben's fork-resolution idea a disaster? Explain briefly.

Answer: In real Bitcoin, if an attacker wants to eliminate a transaction that occurs many blocks in the past by creating a fork from before that transaction, the attacker has to *sustain* a block mining rate faster than the main chain long enough to catch up, which requires compute power believed to be too expensive for most attackers. But with Ben's scheme, an attacker only needs to mine a single block that happens to have a hash smaller than the corresponding block in the main chain; then all nodes will switch to the attacker's new short fork. The attacker needs relatively little compute power to mine this single block.

Name: ____________________

X PBFT

Consider the PBFT protocol as described in the paper *Practical Byzantine Fault Tolerance* by Castro and Liskov.

For each of the following statements, indicate whether it is true or false.

14. [6 points]:

True / False : Assume all replicas of a PBFT-replicated system are running the same software and the software has a bug. If an attacker exploits this bug in all replicas to control each replica, then the attacker can commit arbitrary operations in the log.

True / False : If an attacker controls the primary of a PBFT-replicated system, then the attacker can commit arbitrary operations in the log.

True / False : Honest replicas must include in a VIEW-CHANGE the messages already prepared so that a new primary cannot omit already-committed operations.

Answer: 1. True; PBFT doesn't guarantee correctness if more than f machines are compromised. 2. False; the attacker must control more than f machines to be able to compromise PBFT; 3. True; see protocol description.

15. [2 points]: If there are 10 machines in our system, and at most 2 machines are malicious, how many REPLY messages does a client need to receive before it knows its operation was executed? (Circle best answer)

A. 1
B. 3
C. 4
D. 6
E. 7

Answer: B: f+1, where f = 2; as stated in the paper, we only need to guarantee that one honest replica voted REPLY.

Name:

XI 6.5840

16. [1 points]: Which lectures/papers should we **omit** in future years?

- FaRM **(4)**
- Chardonnay **(11)**
- Grove **(85)**
- Memcache at Facebook **(7)**
- DynamoDB **(25)**
- AWS Lambda **(19)**
- Ray **(14)**
- SUNDR **(8)**
- Bitcoin **(4)**
- PBFT **(12)**

17. [1 points]: Porcupine, the linearizability checker used in the labs, comes with a visualizer that displays in a web browser the entire history of client and tester operations. Circle the one closest to your experience with the visualizer.

- I haven't used it for debugging. **(10)**
- I've tried using it, but it wasn't helpful in my case. **(27)**
- I've read the visualization, and it occasionally helped me identify the issue in my code. **(78)**
- I've read the visualization, and it often helped me identify the issue in my code. **(15)**
- I've added my own annotations, and it enhanced my debugging experience. **(18)**
- Other (please briefly describe your experience): **(3)**

18. [1 points]: Do you have any feedback for us about 6.5840?

Lectures and Notes.

- Easier-to-read/clearer lecture notes/slides (7)
- Lecture notes are helpful (1)

Name: ____________________

- Recording lectures (4)
- Guest lectures interesting but lacking depth (1)
- Like Grove (1)
- Like Lambda lecture, less so about Dynamo (1)
- Love guest lectures (2)
- Some guest lecture notes or slides are missing (1)
- More notes for Lambda and Dynamo (1)
- Adding lecture reading group/recitation (2)
- Big picture example/tie everything together (2)

Labs—Content and Design.

- Less cumulative/dependent labs (12)
- Labs more connected to lectures (2)
- More diversive labs (3)
- Bitcoin lab (1)
- Bring back Spark (1)
- Good Raft implementation at the end (1)
- Release Raft lab solution (so can build off lab 4/5 with correct Raft) (4)
- Less Raft (4)

Labs—Debugging and Support.

- Better error messages for labs (1)
- Better debugging experience for lab 4 and 5 (1)
- Extra debugging tools (4)
- Ways to debug common bugs in labs (1)
- Annoying bugs in tester (1)

Name: ____________________

- More help on catching up previous labs (1)

Labs—Logistics.

- Lab for writing test cases for 5D (1)
- Clearly communicate how to do/submit 5D (1)
- Split lab 5B/C due date (1)
- Lab checkoffs (1)
- Solution after lab deadline (1)
- Extra credit for labs (2)
- Labs worth more points (1)

TAs, OHs, and Support.

- More OHs (9)
- More TA per OHs (4)
- More TAs (6)
- More late hours (2)
- More support/bug fixes for WSL (1)
- Lab hour suggestions (1)

Exams and Grading.

- Classroom not big enough for exams (1)
- No 9am exams (1)
- Less true/false exam questions (1)
- Less multiple-choice questions on exams (1)
- Exam too hard (2)

Readings.

Name: ____________________

- Release answers to reading questions (1)
- Papers interesting but isolated (1)
- More background materials for reading (1)
- Some of the papers too dense (1)
- More visualizations for the readings (1)
- Weekly quizzes (1)
- Fewer papers (1)

Topics and Content Requests.

- Would be useful to understand Paxos before learning Raft (1)
- More paper questions (1)
- More AI/LLM systems (1)
- Explain linearizability more (1)
- Practice problems (1)

Miscellaneous.

- Final project guidelines more specific (1)
- Recitations (3)
- Custom Porcupine colors (1)
- More units (1)

End of Exam II

Name: ______________________________