Department of Electrical Engineering and Computer Science
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033 Computer Systems Engineering: Spring 2019

Quiz 2

There are 14 questions and 13 pages in this quiz booklet. Most questions have multiple parts. Answer each question according to the instructions given. You have two hours to answer the questions.

- The questions are organized (roughly) by topic. They are not ordered by difficulty nor by the number of points they are worth.

- **If you find a question ambiguous, write down any assumptions you make.** Be neat and legible.

- Some students will be taking a make-up exam at a later date. Do not discuss this quiz with anyone who has not already taken it.

- You may use the back of the pages for scratch work, but do not write anything on the back that you want graded. We will not look at the backs of the pages.

- Write your name in the space below. Write your initials at the bottom of each page.

This is an open-book, open-notes, open-laptop quiz, but you may NOT use your laptop, or any other device, for communication with any other entity (person or machine).

Name:

Email: **@mit.edu**

*Please use only your MIT email ID.*

GFS and ZFS

1. [5 points] For each of the following phrases, indicate whether it relates to GFS, ZFS, both, or neither. Pick the BEST choice for each line. Circle only a single choice on each line.

   A. **GFS / ZFS / Both / Neither** Reads may see inconsistent versions of data

   B. **GFS / ZFS / Both / Neither** Multiple different sizes of block/chunk

   C. **GFS / ZFS / Both / Neither** Logging based on blockchain

   D. **GFS / ZFS / Both / Neither** Leases

   E. **GFS / ZFS / Both / Neither** Checksum for child block/chunk stored in parent block/chunk

Initials:

Transactions

2. [8 points] Consider the following schedule and conflict graph for transactions T1, T2, T3, T4.   For the schedule, time increases as you move down the page.

.

T1	T2	T3	T4
write(y, 0)			
write(z, 0)			
write(x, 10)			
		read(x)	
			write(y, 20)
		read(y)	
	read(y)		
	write(z, 30)		
read(z)			
write(x, 20)			
			read(x)
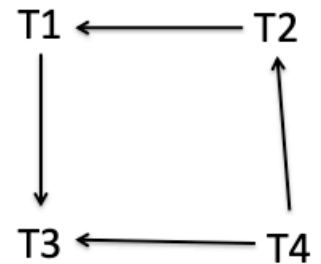
T1 ⟵——— T2
│   ↑
↓   │
T3 ⟵——— T4

A. Which arrows need to be added to the diagram so that it is complete?

    a. T1 → T2 and T3 → T1
    b. T1 → T4 and T4 → T1
    c. T1 → T2, T3 → T1, and T1 → T4
    d. T1 → T2, T3 → T1, T1 → T4, and T4 → T1
    e. T1 → T2, T3 → T1, and T3 → T2
    f. No more arrows needed, the diagram is complete as given

B. Identify a SINGLE action in the above schedule that can be removed to make this group of transactions conflict-serializable.

    a. T3: read(x)
    b. T2: read(y)
    c. T1: write(x,10)
    d. T4: write(y, 20)
    e. T2: write(z, 30)
    f. There is no single action that can be removed to produce a conflict-serializable schedule

Initials:

Replicated State Machines

3. [6 points] Consider a replicated state machine implemented by N+1 replicas. There is one primary P and N backups: B_1, B_2, .., B_N. However, there is no view server.

    Instead, this replicated state machine uses the IDs of the backups as an ordering to determine who becomes the next primary in case of failures. Specifically: if P fails, everyone in the system has agreed that B_1 will be the next primary. If B_1 fails, B_2 will become primary, and so on through B_N.

    Assume that there is an efficient heartbeat protocol that allows any replica to check the status of any other replica by sending a request and receiving a corresponding reply. The effect is that every replica knows instantaneously when some other replica is not responding. Further assume that each replica is either "up" (fully functioning, responding to messages) or "down" (not responding to messages), so you do not need to be concerned about partial failure of a replica or recovery of a replica.

    We'll call it a *partition* whenever there is any impairment of replica-to-replica communication. One particular kind of partition is an *asymmetry in message delivery*: that's a partition where some replica A can't send to some other replica B, even though B can send to A.
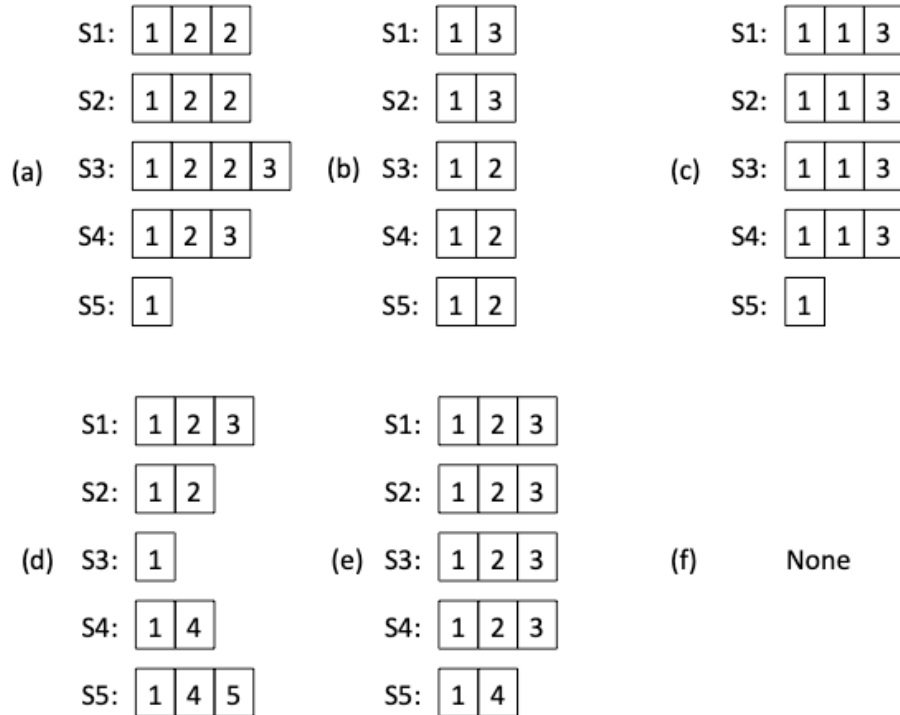
    Answer True or False to each of the following completions of the claim that "This scheme ensures that there is always **at most one** replica acting as primary **out of the entire set of replicas**…"

    A. **True / False** …in any network that is guaranteed to have no partitions.

    B. **True / False** …even in a network that allows partitions, as long as there are no asymmetries in message delivery.

    C. **True / False** … even in a network that has asymmetries in message delivery.

Initials:

Raft

4. [10 points] Each of the following diagrams shows the logs at five servers S1 through S5. Assume that the five servers are implementing five Raft replicas of a single replicated state. As in the paper, the log entries have been abstracted to show term numbers only.

(a)
S1: | 1 | 2 | 2 |
S2: | 1 | 2 | 2 |
S3: | 1 | 2 | 2 | 3 |
S4: | 1 | 2 | 3 |
S5: | 1 |

(b)
S1: | 1 | 3 |
S2: | 1 | 3 |
S3: | 1 | 2 |
S4: | 1 | 2 |
S5: | 1 | 2 |

(c)
S1: | 1 | 1 | 3 |
S2: | 1 | 1 | 3 |
S3: | 1 | 1 | 3 |
S4: | 1 | 1 | 3 |
S5: | 1 |

(d)
S1: | 1 | 2 | 3 |
S2: | 1 | 2 |
S3: | 1 |
S4: | 1 | 4 |
S5: | 1 | 4 | 5 |

(e)
S1: | 1 | 2 | 3 |
S2: | 1 | 2 | 3 |
S3: | 1 | 2 | 3 |
S4: | 1 | 2 | 3 |
S5: | 1 | 4 |

(f)   None

Identify ALL of the diagrams that are legal in Raft (if any). Clearly indicate your choice(s) by circling the corresponding letter(s).

If you decide that none of them are legal, circle choice (f) above instead.

Initials:

Concurrency control, logging, and recovery

5. [10 points] Answer True or False for these statements:

    A. **True / False** A cache is an example of volatile storage, while main memory is an example of non-volatile storage.

    B. **True / False** A NO-FORCE policy causes more work during REDO than a FORCE policy.

    C. **True / False** Pessimistic concurrency control gets its name because it causes more aborted transactions.

    D. **True / False** Two-phase locking is required when using write-ahead logging.

    E. **True / False** Hierarchical locking schemes are incompatible with ARIES.

6. [6 points] Consider the following excerpt from a simple write-ahead log. UPDATE records are of the form: UPDATE <var name>=<old value>; <var name>=<new value>. After entry 41, the system crashed.

```
Entry # | Trans ID  | Entry

31          111               UPDATE A=0; A=1
32          111               UPDATE B=0; B=1
33          111               COMMIT
34          112               BEGIN
35          112               UPDATE A=1; A=2
36          112               UPDATE B=1; B=3
37          113               BEGIN
38          113               UPDATE C=0; C=1
39          113               UPDATE A=2; A=0
40          113               COMMIT
41          112               UPDATE B=3; B=4
```

    A. During the database's recovery process, which transaction(s) get undone? Circle one or more of the transaction IDs, or circle None if you believe none are undone.

       None   111   112   113

    B. **True / False** This log excerpt is consistent with a system that uses strict 2-phase locking (holds all locks until the end of a transaction).
    C. **True / False** The commit status of a transaction is determined by whether there is a COMMIT entry in the log, even in a system with a NO-FORCE policy

Initials:

Two-phase commit

7. [6 points] Two servers A, C have modified data on behalf of distributed transaction T. Server B is acting as the coordinator for the transaction commit. The servers may fail, and they communicate via a network where both network failures and partitions are possible.

   In contrast to the system described in lecture, there are no timeouts. Accordingly, when a remote server is not responding there is no reliable way of determining whether it has crashed or is up but unreachable. The transaction T has reached its end and is trying to commit. The following actions have happened as part of the two-phase commit:

     i. B writes "prepare" to its log
    ii. B tells A to prepare
   iii. B tells C to prepare
    iv. C writes changes and "prepared" to its log
     v. A writes changes and "prepared" to its log
    vi. A tells B that A has prepared
   vii. C tells B that C has prepared
  viii. B writes "commit" to its log

Immediately after step viii, B crashes. Answer the next two questions assuming that B has not yet recovered. Pick the BEST response.

A. Can server A commit?

     a. Yes, because B has committed
     b. No, because A does not know the outcome
     c. Yes, because both A and C have prepared
     d. No, because any failure causes an abort

B. Can server C abort?

     a. Yes, because B has crashed
     b. No, because C does not know the outcome
     c. No, because both A and C have prepared
     d. Yes, because any failure causes an abort

C. Now server B recovers and re-establishes communication with servers A and C. Can server A commit now?

     a. Yes, after B informs A that the outcome is commit
     b. No, because the transaction already committed while B was crashed
     c. No, because the transaction already aborted while B was crashed
     d. No, because now the transaction aborts although it couldn't abort before

Initials:

Passwords

8. [6 points] Consider the following schemes for storing username and password information. We consider a single line containing the fields that are needed for a single user's login lookup; the table as a whole consists of a large number of these lines.

   The different fields in the line are separated by commas (field1, field2, field3). "salt" is a randomly generated 10-digit number. Assume there are no issues with the random number generator. "|" means concatenate. The hash function is "slow" – it takes a substantial amount of computation.

   We want to defend against a so-called "rainbow table" attack, in which the adversary has already computed hashes of many common passwords.

   For each scheme, circle the correct answer to the question: Is this scheme resistant to rainbow table attacks?

   A. **Yes / No** username, salt, hash(password) | hash(salt)

   B. **Yes / No** username, salt, hash(password) | hash(password | salt)

   C. **Yes / No** username, salt, hash(hash(password) | salt)

Initials:

Secure Channels

9. [10 points] Bob has built his own secure channel between his two servers A and B. He has an efficient, reliable, and secure encryption engine that can take 512 cleartext bytes and a secret key to produce 512 encrypted bytes. It can also take 512 encrypted bytes and a (matching) secret key to produce the original 512 cleartext bytes.

    Bob's applications always send each other messages that are exactly 512 bytes long, via some sensible and workable but otherwise-unremarkable transport protocol. His implementation has one secret key X that's used for traffic going from A to B, and a different secret key Y for traffic going from B to A. The secret keys X and Y are both "good enough" (long enough, random enough) and are simply hardcoded into the servers A and B.

    Assume a threat model in which the adversary can see any traffic on the network, can create messages on the network, and can destroy messages on the network. However, the adversary cannot compromise server A or server B, and cannot determine any secret key (X or Y) from any number of collected messages.

    A. **True / False** Bob's channel may be vulnerable to a replay attack by the adversary

    B. **True / False** Bob's channel may be vulnerable to a reflection attack by the adversary

    C. **True / False** Bob's channel may be vulnerable to the adversary pretending to be A or B.

    D. **True / False** Bob's channel is vulnerable to the adversary reading the application data (cleartext) of messages being sent between A and B.

    E. **True / False** The adversary may be able to infer something about Bob's application from the communication pattern between A and B.

Initials:

DNSSEC

10. [6 points] You're talking with someone who read the DNSSEC paper but missed a couple of important ideas. Pick the SINGLE BEST correction for their not-quite-right claims.

   A. "With DNSSEC, there's no reason to be concerned about getting faked domain information when you're working in a coffee shop. The connection between your browser and the DNS server is protected, so you are safe from spoofing and interception attacks on DNS."

      a. Actually, TLS is secured by a completely different mechanism.
      b. Unfortunately, DNSSEC doesn't secure the browser-to-DNS communication.
      c. Yes, but first you have to take part in an elaborate key-signing ceremony.
      d. DNS contains public information, so spoofing and interception are not real threats.

   B. "DNSSEC uses encryption to hide server addresses from hackers."

      a. DNSSEC actually uses signatures for integrity
      b. DNSSEC actually hashes server addresses
      c. DNSSEC only encrypts the roots, not all servers
      d. Zone walking is actually how servers are protected

Cryptosystem failure

11. [8 points] Circle true or false for the following statements:

   A. **True / False** Weak cryptographic algorithms are the primary problem with the security of systems deployed in the real world.

   B. **True / False** Back-door capabilities are a risk to security.

   C. **True / False** Secret algorithms and secret implementations strengthen the security of a system.

   D. **True / False** Trustworthy components assembled together guarantee a secure system.

Initials:

Botnets, Blockchain, Bitcoin

12. [10 points] Marie has come up with an idea to combine botnets and blockchain. She figures that the current running Bitcoin system can be used as a coordination mechanism between bots and her C&C server. Each new bot will post its availability in a transaction on the Bitcoin blockchain so the C&C server can see it. Likewise, the C&C server will post commands for each of the bots in the same blockchain.

    Assume that Marie has written all of the code for the bots and the C&C server.

    Answer True or False for each of these claims about the resulting system.

    A. **True / False** A nonpermissioned blockchain (like Bitcoin) is unnecessary for this application, since Marie controls all of the potential writers.

    B. **True / False** If Marie's coordination scheme depends on her bots adding blocks directly (mining), it may be slow and unpredictable.

    C. **True / False** If Marie's scheme depends on sending transactions via other miners, she will have to pay them.

    D. **True / False** A long consistent public history is important whenever a bot is trying to coordinate with a C&C server.

    E. **True / False** Regardless of how well the coordination works, Marie's bots are likely to make her rich via Bitcoin mining.

Initials:

Meltdown

13. [8 points] A colleague is proposing mitigation strategies for Meltdown in an architecture they are designing. For each proposal, choose the BEST response.

A. "We could eliminate speculative execution."

    a. No Meltdown attacks would be affected by that change
    b. That could help against Meltdown, but cost too much performance
    c. That is a mitigating technique recommended in the published paper
    d. That would fix the problem, and eliminate stack smashing too

B. "We could eliminate caching."

    a. No Meltdown attacks would be affected by that change
    b. That could help against Meltdown, but cost too much performance
    c. That is a mitigating technique recommended in the published paper
    d. That would fix the problem, and eliminate stack smashing too

C. "We could add a mode that splits the address space."

    a. No Meltdown attacks would be affected by that change
    b. That could help against Meltdown, but cost too much performance
    c. That is a mitigating technique recommended in the published paper
    d. That would fix the problem, and eliminate stack smashing too

D. "We could change the operating system to map in less of the kernel's address space."

    a. No Meltdown attacks would be affected by that change
    b. That could help against Meltdown, but cost too much performance
    c. That is a mitigating technique recommended in the published paper
    d. That would fix the problem, and eliminate stack smashing too

Initials:

Cheap Point

14. [1 point] Consider this to be our market research (again): We want to know which papers you particularly liked or disliked.

    Please pick papers that we read this term, and describe them in a way that we can recognize. Although the rest of the quiz is about material starting with GFS, you can tell us a paper from the first part of the term if you prefer.

    Feel free to comment further about why you like or don't like a paper, but first be sure you've completed all the previous questions.

    A. What was your favorite paper?

    B. What was your least favorite paper?

End of Quiz 2

Please double-check that you have written your name and email ID on the first page and initialed all of the pages

Thanks for taking 6.033! Have a great summer!

Initials: