CPS 512midterm exam #1, 10/7/2016

Your name please: ____ NetID:____________

/60
/40
/10
/10
/20
/60
/200

Answer all questions. Please attempt to confine your answers to the boxes provided. If you don't know the answer to a question, then just say something else relevant for partial credit. T/Fs and shorts are 5 points each.

P1. Transactions: T/F (60 points)

Indicate whether each statement is true or false by placing a T or F to the **left** of the statement. You can place additional notes or conditions to the right.

1. In the two-phase commit protocol (2PC), a transaction T always commits if a majority of participants vote to commit T.
2. Commitment is durable in client-server transaction systems with 2PC, as long as a majority of participants survive and retain their state, even in the presence of a network partition.
3. When a transaction T commits with 2PC, there is always some point in the protocol at which all participants have voted, but no participant knows the outcome of T.
4. When a transaction T commits with 2PC and two-phase locking (2PL), each participant releases locks held by T when it votes to commit T.
5. Failure of the coordinator during a two-phase commit (2PC) never forces the transaction to abort.
6. Failure of a participant other than the coordinator during a 2PC (failure of a storage server or RM) always forces the transaction to abort.
7. In client-server transaction systems, failure of the client during a 2PC always forces the transaction to abort.
8. In a transactional server with a NO-STEAL/NO-FORCE buffer manager (e.g., Birrell/Wobber), the server always writes updates to the log before it responds to the client.
9. In a transactional server with a NO-STEAL/NO-FORCE buffer manager (e.g., Birrell/Wobber), the server does not write updates to the file system until after it has responded to the client.
10. In a transactional server with a NO-STEAL/NO-FORCE buffer manager (e.g., Birrell/Wobber), taking checkpoints (snapshots) more frequently increases the cost of recovery.
11. In a transactional server with logging, each update record in the log must identify the transaction that issued the update.
12. Any transactional service is linearizable if each request is executed at most once.

P2. More T/Fs on Consensus and Consistency (40 points)

Indicate whether each statement is true or false by placing a T or F to the **left** of the statement. You can place additional notes or conditions to the right. **Consensus** refers to Paxos, Viewstamped Replication (VR), or Raft. These protocols are used for primary/backup replication, e.g., Replicated State Machine (RSM) replication.

1. In Consensus, any partition in the network always results in denial of service (no availability) on at least one side of the network partition.
2. In Consensus, any partition in the network always results in denial of service (no availability) on at most one side of the network partition.
3. In Consensus, no participant (other than the primary) executes a request until all participants have voted on it.
4. In Consensus an operation always commits if the majority of participants vote to commit it.
5. Commitment is durable in Consensus replication as long as a majority of participants survive and retain their state, even in the presence of a network partition.
6. A Chubby client can be in jeopardy only after it acquires a session lease at least once.
7. Any service that is replicated with Consensus is linearizable if each request is executed at most once.
8. Any service is linearizable if each request is executed at most once.

P3. Replicated State Machine (10 points)

Please confine your answers to the space provided.

1. The Replicated State Machine (RSM) model of primary-backup replication with Consensus does not scale: it does not yield higher throughput as we add servers. Why? What could go wrong if we permit any RSM replica to act as the coordinator for a write operation?

2. What could go wrong if we permit RSM backup servers to execute read operations?

P4. Chubby (10 points)

Answer each question with a sentence or a short bullet list. Please confine your answers to the space provided.

1. Chubby is a coordination service that uses primary-backup replication with Consensus, and yet it claims to be scalable to large numbers of clients. How does it overcome the scaling limitation?

2. When a Chubby server receives a request to update an object/node, it may send messages to other servers in its cell and to other clients before completing the request. List three reasons for these messages.

P5. RAMcloud (20 points)

Answer each question with a sentence or a short bullet list. Please confine your answers to the space provided.

1. RAMcloud is a key-value store that uses primary-backup replication with Consensus, and yet it claims to be scalable to large (datacenter) server clusters. How does it overcome the scaling limitation?

2. RAMcloud uses RIFL, which provides stronger linearizability assurances than classic RPC (e.g., Birrell/Nelson) because completion records are durable. Sketch an example to show a scenario in which loss of a completion records can lead to a violation of linearizability in classic RPC.

3. RIFL has higher latency than classic RPC, because the completion record for each RPC must be made durable (e.g., by writing it to disk) before issuing a reply to the client. Yet the authors of RIFL claim that the latency cost is negligible for services that support fail-over. Why is the latency cost negligible for these services?

P6. Sharding (20 points)

Given a scalable key-value store (like RAMcloud), any application-tier server (the clients of the K/V store) can access any of the service's data, so in principle any application server can handle any request. Yet many services apply a request routing function to route each request to a selected application server based on the data used by the request: sharding. Give two reasons why sharding can be helpful in the application tier in this setting.

P7. Abstraction boundary (20 points)

In modern distributed services, the API that the application sees is typically different from the RPC API for the service. Why? Illustrate with examples from NFS, Chubby, RAMcloud/RIFL, and other services we have discussed.

P8. Serializability vs. Concurrency (20 points)

Transactional serializability does not imply serial execution. Give a specific example of two transactions for which any interleaving of operations is a serializable execution.