CSE 444 Practice Problems

Distributed DBMS

1. **Two-phase Commit**

   (a) In the two-phase commit protocol, what happens if the coordinator sends PREPARE messages and crashes before receiving any votes?

      i. What is the sequence of operations at the coordinator after it recovers.

      Solution:
      After the coordinator restarts, the recovery process will find that a transaction was executing at the time of the crash and that no commit protocol log record had been written (remember that the coordinator does not force-write any log records before sending PREPARE messages). The recovery process will abort the transaction by "undoing" its actions, if any, using the UNDO log records, writing an abort record, and "forgetting" it.

      ii. What is the sequence of operations at a subordinate that received the message and replied to it before the coordinator crashed.

      Solution:
      If the subordinate sent a NO vote, it knows that the transaction will be aborted because a NO vote acts like a veto. The subordinate does not care if the coordinator crashes or not. It aborts the local effects of the transaction and "forgets" about it.

      If the subordinate sent a YET vote, it cannot make any unilateral decisions. If the subordinate notices the failure of the coordinator (for example by using a timeout), it hands the transaction over to the recovery process. The recovery process will find that it is in the prepared state for the transaction. It will periodically try to contact the coordinator site to find out how the transaction should be resolved. As we discussed above, after the coordinator recovers, it will abort the transaction and will answer "abort" upon receiving an inquiry message. The subordinate will then abort the transaction and "forget" about it.

iii. What is the sequence of operations at a subordinate that did not receive the message before the coordinator crashed.

Solution:
If the subordinate notices the failure of the coordinator (for example by using a timeout), it hands the transaction over to the recovery process. This time, the recovery process will find no commit protocol log records for this transaction. It will abort it and "forget" about it.

(b) In the two-phase commit protocol, why do subordinates need to force-write a prepared log record before sending a YES VOTE? To answer this question, use an example failure scenario. Show what happens if a subordinate does NOT force-write the prepared log record, then show what happens if the subordinate does force-write the prepared log record.

Solution:
Let's assume that all nodes respond to the coordinator with a YES VOTE. In this case, the coordinator will force-write a commit log record and will send COMMIT messages to all the subordinates. At this point, the transaction is considered to have committed. It should thus be durable.

Let's first consider the case when a subordinate does not force write a prepared log record and crashes after sending a YES VOTE. In this scenario, upon restarting, the recovery process will find no commit log records for the transaction. It will abort the transaction and "forget" about it. The system will then be in an inconsistent state. The transaction should have committed at all sites. Instead, one site aborted it.

Let's now consider the case when a subordinate does force write a prepared log record and crashes after sending a YES VOTE. In this scenario, upon restarting, the recovery process will find that it is in the prepared state for the transaction. It will periodically try to contact the coordinator site to find out how the transaction should be resolved. In our scenario, the final outcome of the transaction is a commit. Because the coordinator cannot forget about a committed transaction until it receives final ACKS from all nodes, it will correctly respond with a COMMIT message to the inquiry. The subordinate will then be able to properly commit the transaction.

2. **Replication**

(a) In eager master replication, when the master fails, why does a group of replicas need to have the majority of nodes in order to elect a primary and continue processing requests?

Solution:
The secondaries cannot distinguish between the crash-failure of the master and a network partition. To maintain consistency, the system must ensure that only one partition processes requests at any time. To ensure this property, only the partition that has the majority of nodes is allowed to elect a new primary and continue.

(b) What are the differences between eager and lazy replication? Please list differences in the approaches and differences in the properties that result. Discuss master vs group replication if appropriate.

Solution:
In eager replication, all updates are applied to all replicas as part of a single transaction. In case of a network partition, only the majority partition is allowed to continue processing requests. This scheme thus favors consistency over availability and has a high runtime overhead.

In lazy replication, only one replica is updated as part of the original transaction, leading to a better performance than eager replication. Updates then propagate to other replicas asynchronously. With lazy master replication, when the master fails, it is possible for the system to lose the most recent updates that did not yet propagate to other replicas. With lazy group replication, multiple replicas can perform conflicting update operations. Conflicts must later be resolved manually or by using some pre-defined rules. Lazy group replication thus favors availability over consistency. It does not ensure single node serializability. It can only guarantee convergence.