KEVIN JOHNSON

MODULE 5 READING COMPREHENSION

1. **What is Leader-Based Replication? Give some examples and discuss some problems. (4 Pts)**

   In leader-based replication, one of the replicas (each node that stores a copy of the database) is designated the leader. When clients want to write to the database, they must send their requests to the leader, which first writes the new data to its local storage. The other replicas are known as followers. Whenever the leader writes new data to its local storage, it also sends the data change to all of its followers as part of a replication log or change stream. Each follower takes the log from the leader and updates its local copy of the database accordingly, by applying all writes in the same order as they were processed on the leader. When a client wants to read from the database, it can query either the leader or any of the followers. However, writes are only accepted on the leader.

   One issue that arises in leader-based replication is handling the failure of the leader. In this instance, one of the followers needs to be promoted to be the new leader. Clients then need to be reconfigured to send their writes to the new leader and the other followers need to start consuming data changes from the new leader. This process is called failover and consists of three main steps:

   - Determining that the leader has failed.
   - Choosing a new leader.
   - Reconfiguring the system to use the new leader.

   Failover also has several potential problems:

   - If asynchronous replication is used, the new leader may not have received all the writes from the old leader before it failed. The most common solution for this is for the old leader's unreplicated writes to be discarded.
   - In certain fault scenarios, two nodes could believe they are the leader. If both leaders accept writes and there is no process for resolving conflicts, data is likely to be lost or corrupted. As a safety catch, some systems have a mechanism to shut down one node if two leaders are detected.
   - There can be some ambiguity in determining the right timeout before the leader is declared dead. A longer timeout means a longer time to recovery in the case where the leader fails. However, if the timeout is too short, there could be unnecessary failovers.

2. **Define, compare, and contrast the following types of Multi-Leader Replication: Circular Topology, Star Topology, All-to-All Topology. (3 Pts)**

   A replication topology describes the communication paths along which writes are propagated from one node to another. If there are two leaders, leader 1 must send all of its

writes to leader 2, and vice versa. With more than two leaders, various topologies are possible.

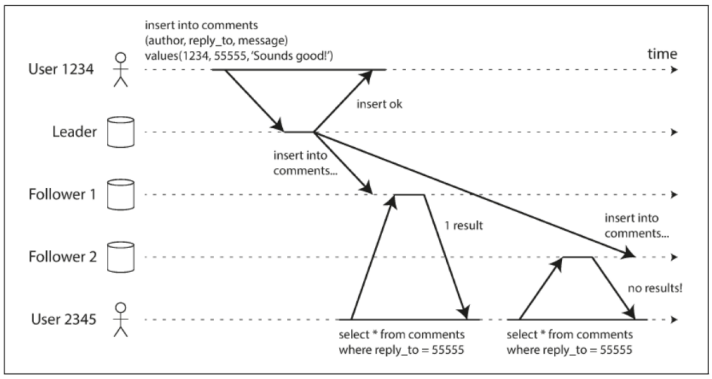
In a circular topology, device connections create a circular data path. Each node is connected to two others. In circular topology, each node receives writes from one node and forwards those writes to one other node. One disadvantage of circular topology is that if any individual connections in the ring is broken, the entire network is affected.

In a star topology, all nodes are individually connected to a central connection point. One designated root node forwards writes to all of the other nodes. An advantage to star is that the configuration is easy to setup.

All-to-all is the most general topology, in which every leader sends its writes to every other leader. Benefits of this topology is that it is robust and faults are diagnosed easily. However, installation and configuration is difficult.

In circular and star topologies, a write may need to pass through several nodes before it reaches all replicas. Therefore, nodes need to forward data changes they receive from other nodes.

3. **Explain what is going on in the following architecture and what we can do to fix it. (3 Pts)**



In this example, User 2345 is making the same query twice, first to a follower with little lag and then to a follower with greater lag. The first query returns a comment that was added by user 1234 but the second query does not return anything because the lagging follower has not yet picked up that write. The second query is observing the system at an earlier point in time than the first query. As a result, user 2345 will see user 1234's comment appear and then disappear, which will cause confusion.

To fix this problem, monotonic reads can be used. Monotonic reads means that if one user makes several reads in sequence, they will not see time go backward, meaning that they will not read older data after having previously read newer data. One way of achieving monotonic reads is to make sure that each user always makes their reads from the same replica. However, if that replica fails, the user's queries will need to be rerouted to another replica.