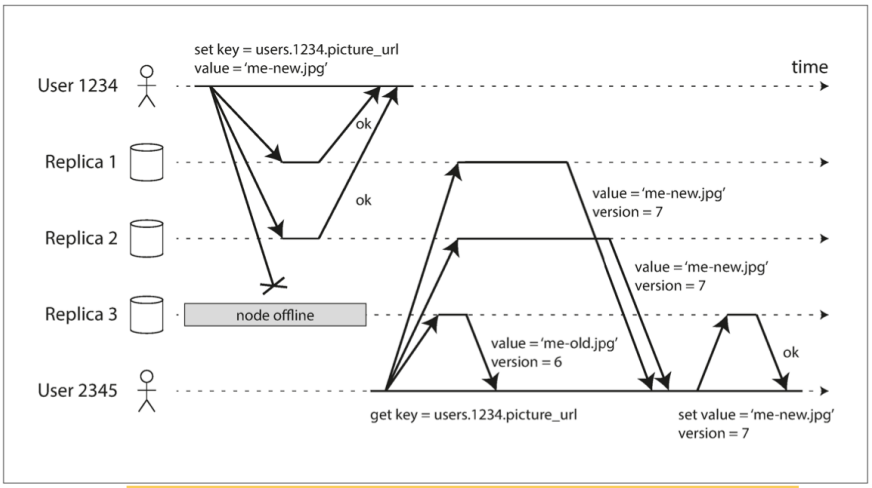
Chapter 5. Replication (Part II)

▼ Author: Pylon, Peng

Leaderless is A different approach, abandoning the concept of a leader and allowing any replica to directly accept writes from clients, is known as leaderless replication. In some leaderless implementations, the client directly sends its writes to several replicas, while in others, a coordinator node does this on behalf of the client.

Date: @March 16, 2023 10:19 PM

Topic: Writing to the Database when a node is down

Recall	Notes
Quorum writes on leaderless	 *Figure 5-10. A quorum write, quorum read, and read repair after a node outage.* Clients send a write to all replicas.
All replicas should ensure all the data is copied to every replica. After an	Two mechanisms are often used:

unavailable node comes back online, how does it catch up on the missed writes?

#1 Read repair: Repair the data if find stale value on a replica when reading.

#2 Anti-entropy process: background process that monitors the differences among replicas and repairs them constantly.

Quorums for reading and writing

Assumptions:

- n replicas
- w replicas to be considered successful for writes
- read at least r replicas

As long as w + r > n, we expect to get an up-to-date value when reading.

Reads and writes that obey these *r* and *w* values are called *quorum* reads and writes.

For reading-intensive cases, may benefit from setting w = n and r = 1.

📌 **SUMMARY:**
In the leaderless replication pattern, the client sends the write to all replicas in parallel probably by a coordinator node to do this on behalf of the client. As long as most of them are ok, we think the write is persisted successfully, known as quorum write and quorum read. e.g.

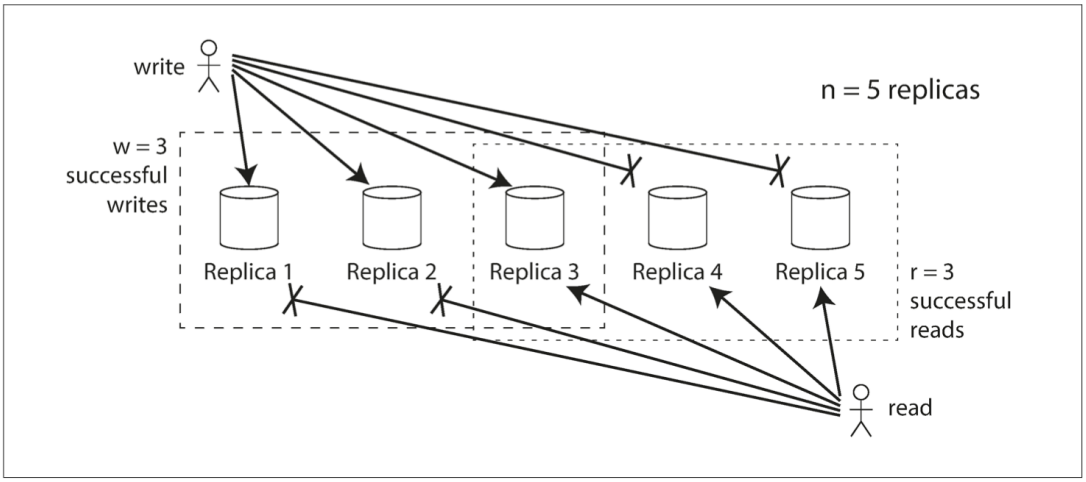


Figure 5-11. If w + r > n, at least one of the r replicas you read from must have seen the most recent successful write.

Date: @March 20, 2023 9:09 PM

Topic: Limitations of Quorum Consistency

Recall	Notes
What is Quorum consistency?	Quorum consistency is a consistency model that requires a certain number of replicas to agree on a value before it can be considered up-to-date. Specifically, for a system with *n* replicas, a write is considered successful if it is written to at least *w*

replicas, and a read is considered successful if it reads from at least *r* replicas, where *w* + *r* > *n*. Reads and writes that obey these *r* and *w* values are called *quorum* reads and writes. However, there are limitations to quorum consistency, such as the potential for write conflicts and the possibility of stale reads.

What is a sloppy quorum?

A sloppy quorum is a trade-off approach in which writes and reads still require *w* and *r* successful responses, but those may include nodes that are not among the designated *n* "home" nodes for value. It is used in a large cluster where the client can connect to some database nodes during a network interruption but not to the nodes that it needs to assemble a quorum for a particular value.

What are the limitations of Quorum Consistency?

1. there is no longer a guaranteed overlap between the *r* nodes and the *w* nodes with sloppy quorum
2. write conflicts if two writes occur concurrently. LWW based on timestamp can lead to data loss due to clock skew.
3. read, write happen concurrently, it’s undetermined whether the read returns the old or the new value.
4. distribute transaction issues. cannot rollback when write is failed
5. If a node carrying a new value fails, and its data is restored from a replica carrying an old value, the number of replicas storing the new value may fall below *w*, breaking the quorum condition.
6. Timing edge cases around linearizability and quorums.

📌 **SUMMARY:**
From an operational perspective, it's important to monitor whether your databases
are returning up-to-date results. Even if your application can tolerate stale reads, you
need to be aware of the health of your replication.

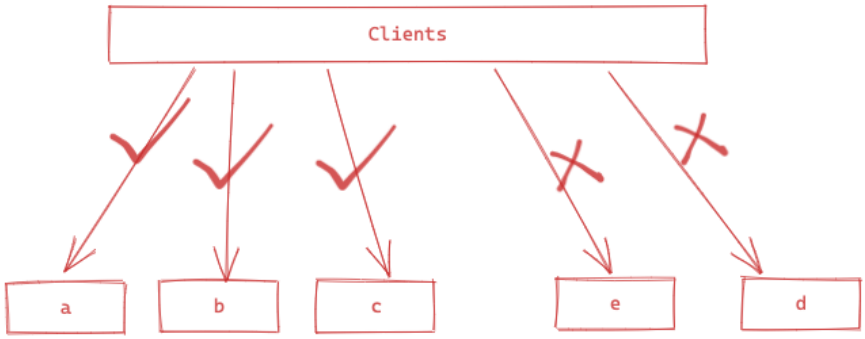
Date: @March 20, 2023 9:29 PM

Topic: Sloppy Quorums and Hinted Handoff

Recall

In a large cluster (with significantly more than *n* nodes) it's likely that the client can connect to *some* database nodes during the network interruption, just not to the nodes that it needs to

Notes



#1 Is it better to return errors to all requests for which we cannot reach a quorum of *w* or *r* nodes?

#2 Or should we accept writes anyway, and write them to some nodes that are reachable but aren't among the *n* nodes on which the value usually lives?

assemble a quorum for a particular value. In that case, we have to face a trade-off.

The latter approach is known as a sloppy quorum: writes and reads still require *w* and *r* successful responses, but those may include nodes that are not among the designated *n* “home” nodes for value. ???

Hinted handoff?

📌 **SUMMARY:**
High availability and low latency are the advantages of leaderless replication with quorums configured appropriately.

Date: @March 20, 2023 9:46 PM

Topic: Detecting Concurrent Writes

Recall

Conflicts arise during read repair or hinted handoff. It is because events may arrive in a different order at different nodes due to variable network

Notes

delays and partial failures.

How do we resolve conflicts? Techniques around this?

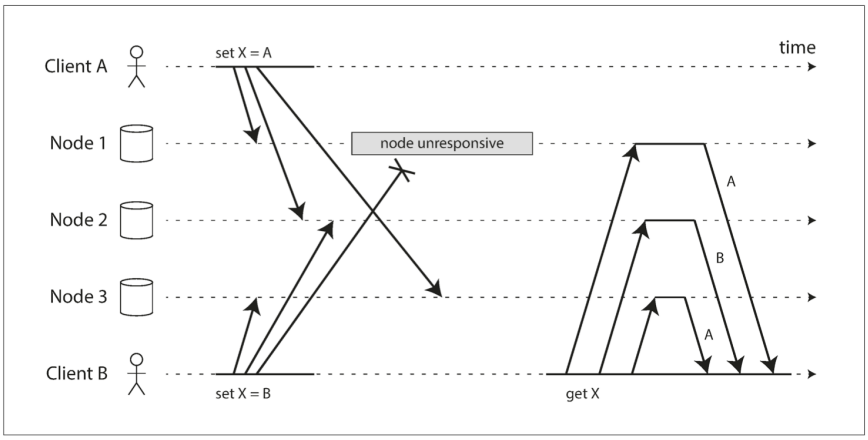


Figure 5-12. Concurrent writes in a Dynamo-style datastore: there is no well-defined ordering.

#1 LWW

#2 The “happens-before” relationship and concurrency. Causally dependent: Sequential. no causal dependency ⇒ Concurrent

#3. Capture the “happens-before” relationship

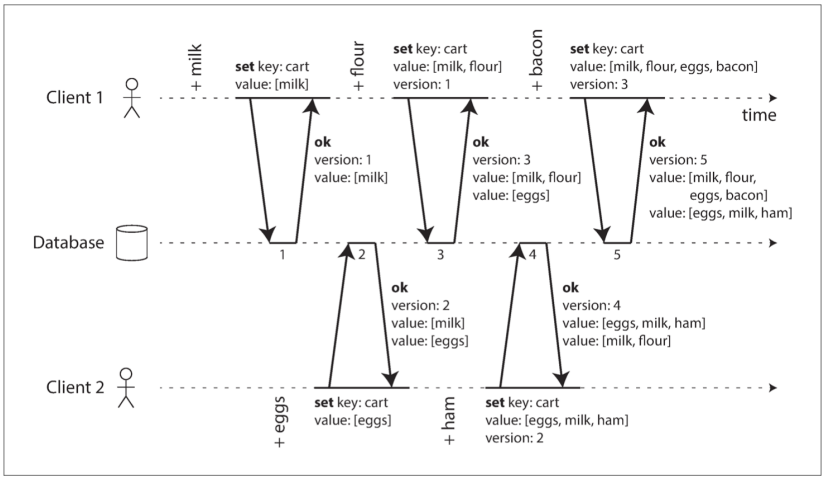


Figure 5-13. Capturing causal dependencies between two clients concurrently editing a shopping cart.

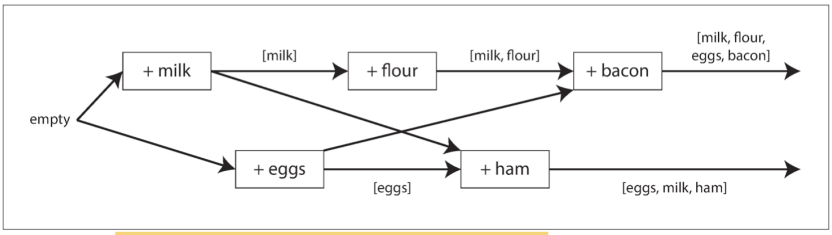


Figure 5-14. Graph of causal dependencies in Figure 5-13.

#4 Merging concurrently written values.

#5 Version vectors

SUMMARY:

Leaderless replication is an approach to replication that abandons the concept of a leader and allows any replica to directly accept writes from clients. In some implementations, the client directly sends its writes to several replicas, while in others, a coordinator node does this on behalf of the client. For leaderless replication, the client sends the write to all replicas in parallel, probably by a coordinator node doing this on behalf of the client. As long as most of them are okay, we think the write is persisted successfully, known as quorum write and quorum read. In this pattern, reads and writes that obey the *r* and *w* values are called *quorum* reads and writes.