04/20 : System Pattern – Leader And Follower : Background

Background

As I told you in my previous article about, how distributed system keeps multiple data for fault tolerance and high availability.
A System use quorum to ensure data consistency between replicas. i.e., all reads, and writes are not considered successful until most nodes participate in the operation.

However , using Quorum can lead to another problem, that is , lower availability(opposite to high availability due to quorum to meet) at any time, the system needs to ensure that at least most replicas are up and available, otherwise the operation will fail.

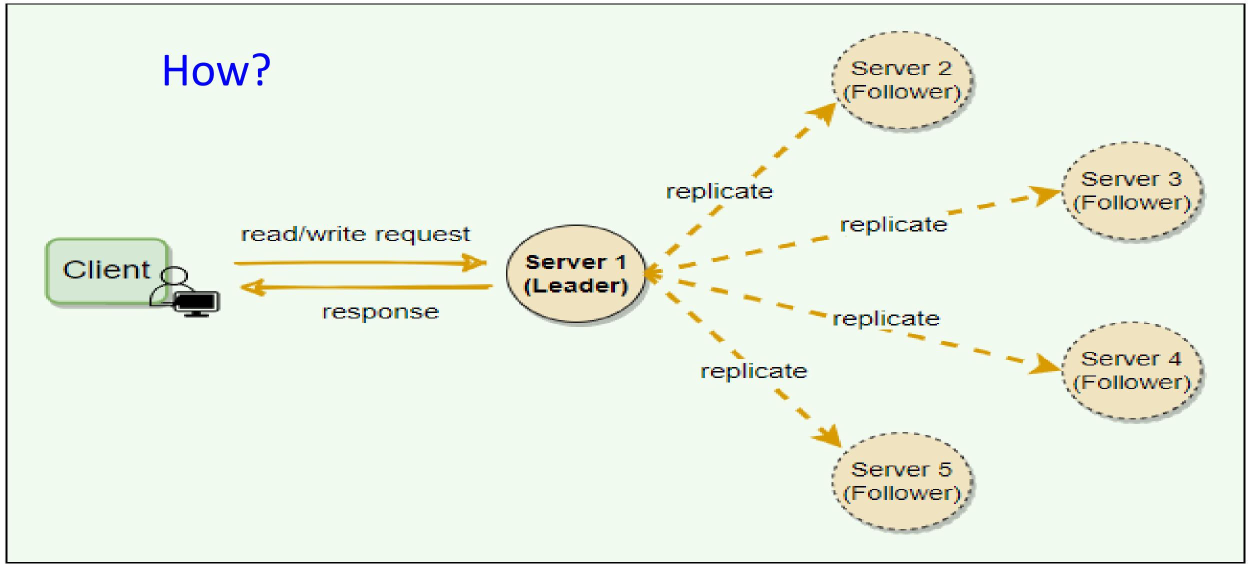
Quorum is also not sufficient, as in certain failure scenarios, the client can still see some inconsistent data.

Then what will be the solution of this problem?
So, in nutshell the quorum also not giving the guarantee otherwise.

Ok, so one idea here is to allow only one single server (**called leader)** to be responsible for data replication and to coordinate work.

So, the concept here is at any time, one server is elected as the leader. This leader becomes responsible for data replication and can act as the central point of all coordination.

The Followers only accept write from leader and serve as a backup. In case the leader fails, one of the followers can become the leader. In some cases, the follower can serve read requests for load balancing.



Examples

- ✓ In **Kafka**, each partition has a designated leader which is responsible for all reads and writes for that partition. Each follower's responsibility is to replicate the leader's data to serve as a "backup" partition. This provides redundancy of messages in a partition, so that a follower can take over the leadership if the leader goes down.
- ✓ To ensure strong consistency, **Paxos** (hence **Chubby**) performs leader election at startup. This leader is responsible for data replication and coordination.