Transactions3:

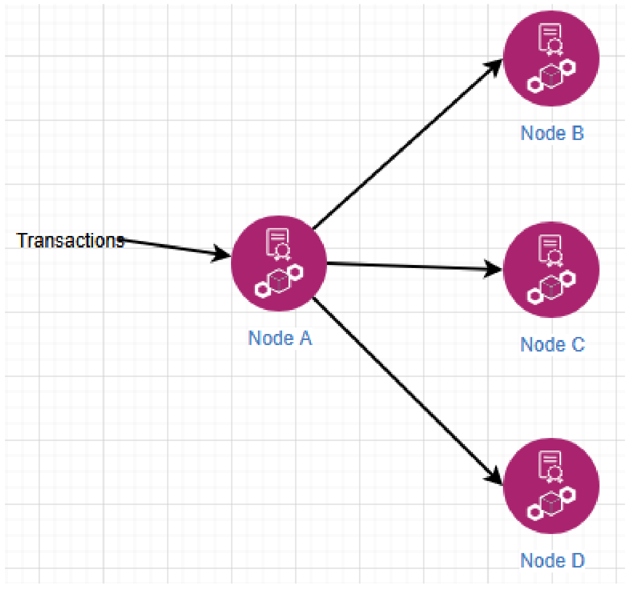
consider a Single node application

User ==> Server ==> Database ==> backup into WAL

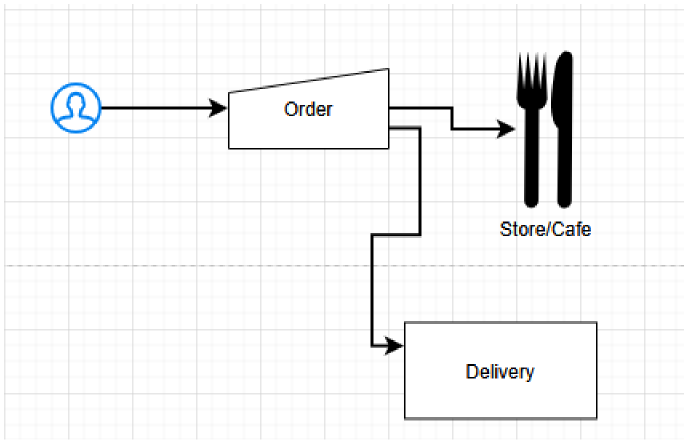
We are creating an order on an e-commerce site. Even if our database crashes, we can recover from WAL log file. WAL contains all the database operations.

Transactions ==> say our WAL itself is not updated ==> We will rollback because WAL is updated ==> Say WAL is updated and DB crashes ==> Recover DB when DB is back online.

Consider Multiple nodes
If we commit a transaction on Node A, we got to make sure the same transaction is committed on Node B, C and D as well



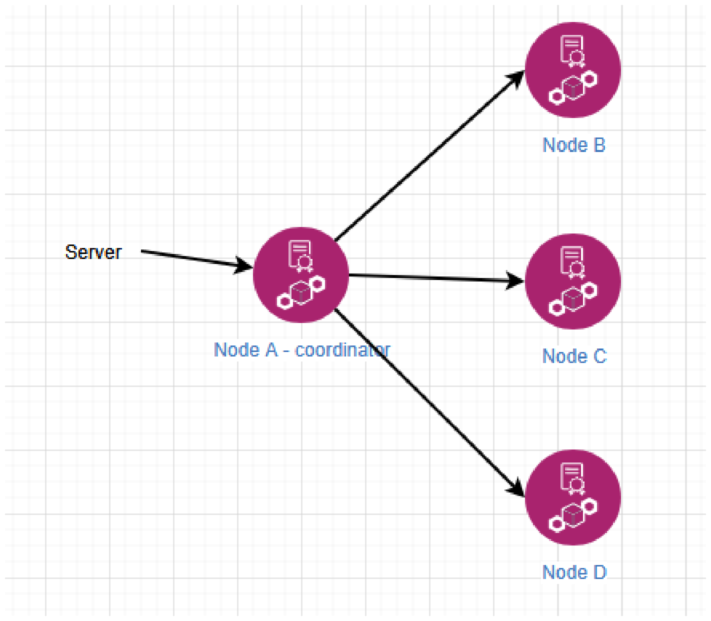
Lets say Node D is not updated. We have to fix with commits. It is called as 2PC – 2 Phase Commit. We have an application like Doordash.



We have to deal with the store and Delivery. Say even if the Store is available to make our food and if the Delivery person is not available the Order should be failed. If Delivery person is ready but Store is closed even that case, the order should be failed.

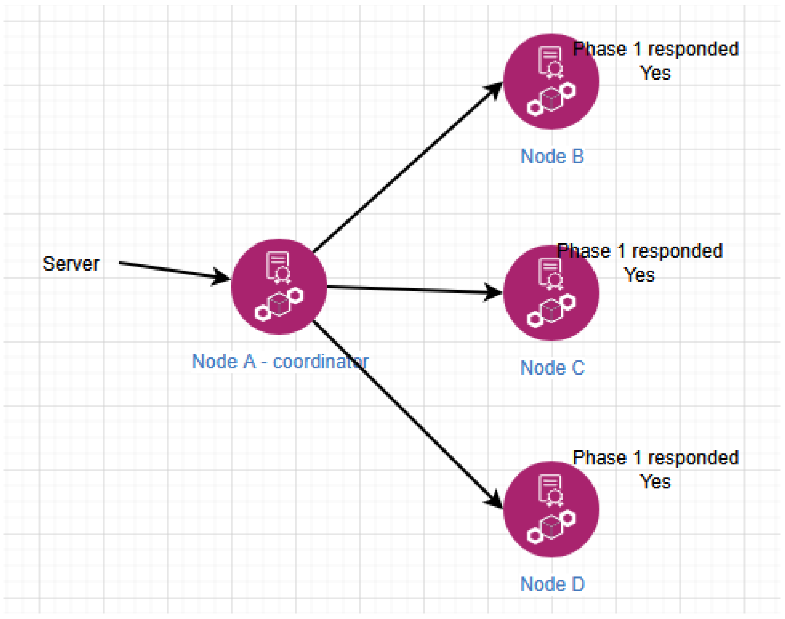
Phase 1: Confirming can we commit? Both should be ready, Store and Delivery person. Only of thats the case, we will make the actual commit in Phase 2.
We will make one node as the coordinator. It will ask other nodes for their readiness.

Phase 2: If all say Yes, Proceed. If anyone says No, we will do a rollback
What if the coordinator goes down after getting transaction & before Phase 1 ==> we will do the rollback

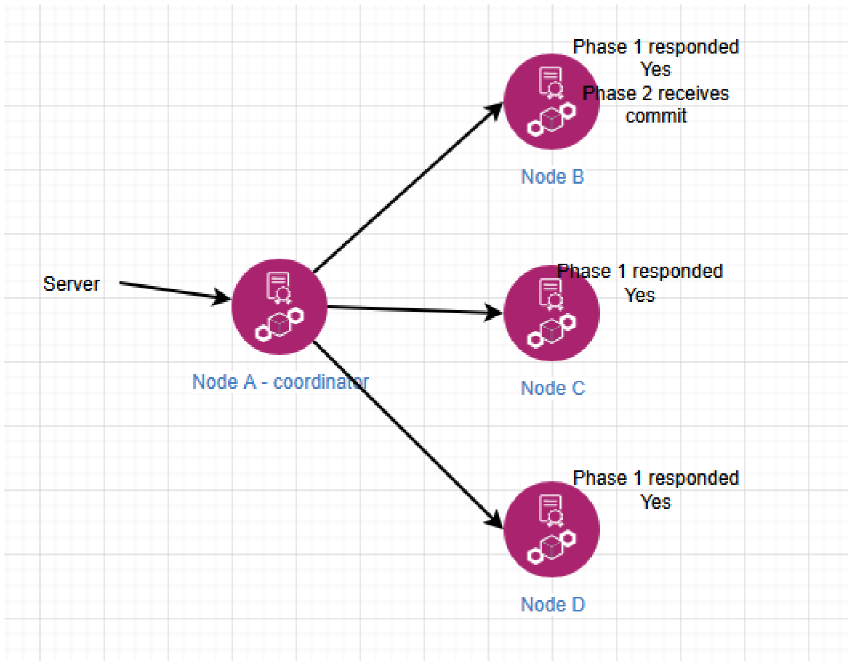


Say Coordinator has updated Node B, C and D then fails. We will do a rollback. There should be a wait period for Nodes.

What if the coordinator sends commit (Phase 2) to another Node then goes down?
Coordinator sends a readiness check to Node B, C and D then the coordinator goes down.

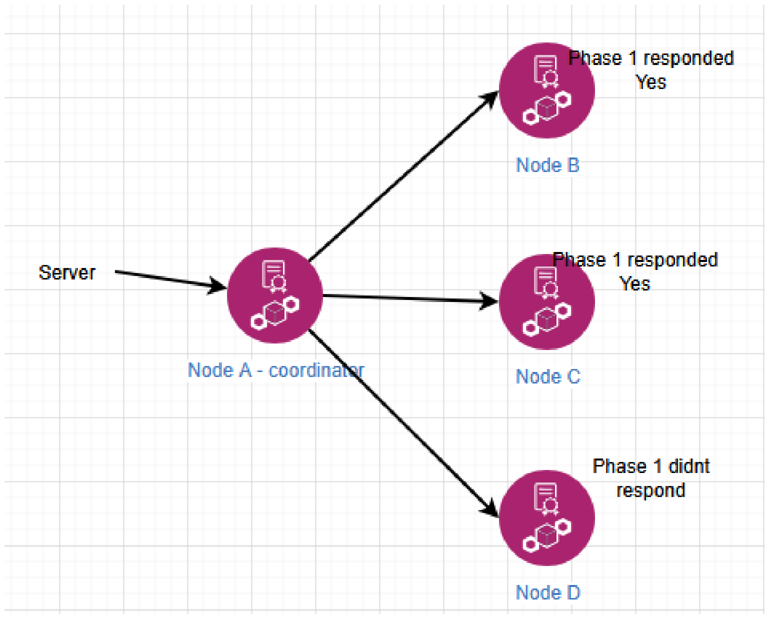


Coordinator sends the commit (Phase 2), only Node B is updated with the latest commit then Coordinator goes down



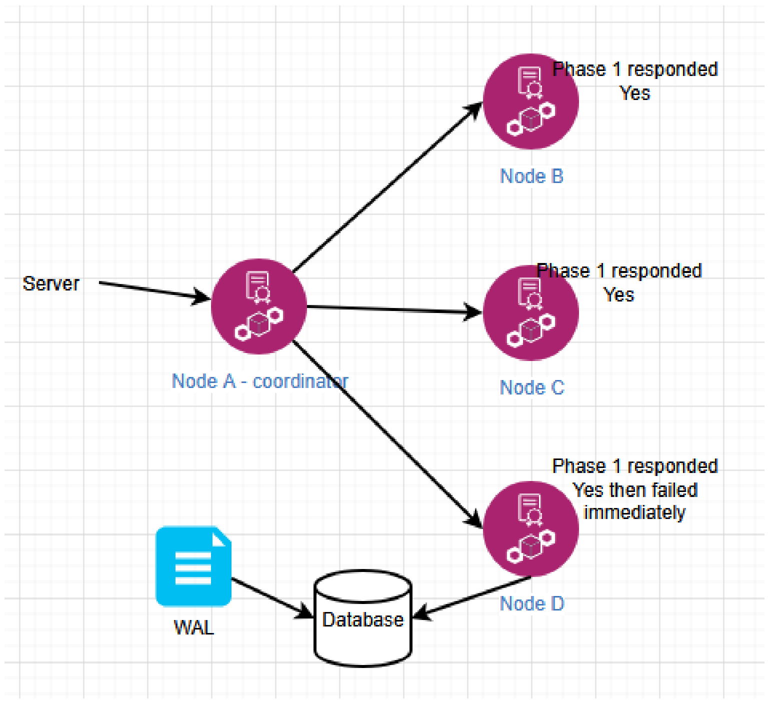
Easy solution is rollback but lets say rollback is costly. We can apply other policies like Gossip to updated Node C and D. or maybe, human intervention is required to fix this. This is one of the drawbacks of this design.

What if one of the nodes is failed? Before Phase I, Node D is down and hence didn’t respond to coordinator



In Phase I, even if one of the responses is No, we will simply rollback

In this case, all Nodes responded Yes to Phase 1 then Node D failed immediately before Phase 2 commit.



Say Node D fails after writing its latest commits on WAL. That means when Node D recovers, it can get the data from WAL. What if WAL itself is not updated? Gossip protocol or human intervention. WAL is essential for one of the ACID property, it is Durability.

Drawbacks of 2 Phase commits:
- Even if we implement 2PC there is a chance of human intervention.
- It is doing the communication twice like TCP that means Latency
- After Phase 1 it is blocking all the nodes. After Phase 1, all Follower nodes are expecting and waiting for Phase 2

Complexity is 2(n-1), we are sending that many requests. Say there are 1 Leader node and 3 Follower nodes. How many communications? 3 for first communication (Phase 1) with nodes + 3 for second communication (Phase 2) with the nodes
To overcome challenges of Phase 2, we have 3 Phase commits.

<u>3PC (3 Phase Commit)</u>
1. can commit?
    Yes Ready
    - If all responded Yes: Proceed else Rollback
2. PreCommit
    - updating WAL
    - Not updating DB
3. DoCommit:
    - update the database

even if the DB fails, we know our WAL is already updated in Phase 2.
Yes 3PC is better than 2PC but at what cost. Complexity is 3(n-1), we are sending that many requests. Say there are 1 Leader node and 3 Follower nodes. How many communications? 3 for first communication (Phase 1) with nodes + 3 for second communication (Phase 2) with the nodes + 3 for Phase 3. So 9 in total. Or 3(n-1) = 3*(4-1) = 9 communications
Time Complexity = 3(n-1)