design Key Value DB : (Dynamo DB)

[Amazon Add to Cart] Feature

Goals : ① Scalability
② Decentralization
③ Eventual consistency.

Steps : ① Partition
② Replication
③ Get & Put OPs
④ Data Versioning
⑤ Gossip protocol
⑥ Merple Tree

① Partition : we introduce n nodes (each node act as Partition) and manage data request with the help of consistent Hashing, this would lead to Scalability.

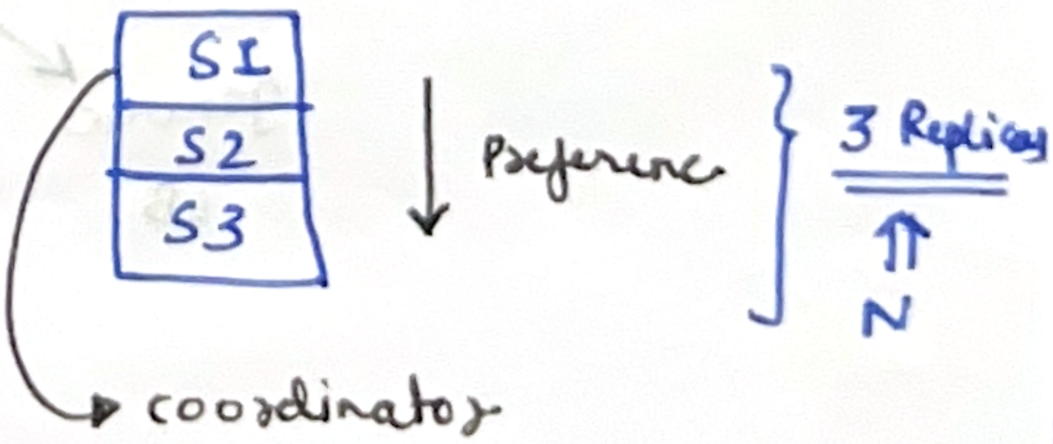
decentralization : even if there are failures in the nodes, the system should not get down, even in failures system should fulfill the request for all incoming keys.

② Replication : Keeping a copy of the data to other servers in order to fallback in case of any node failures

N = Replication count
- 1 is the main node called coordinator
- N-1 nodes in clockwise dir. where the copy of data would be made.

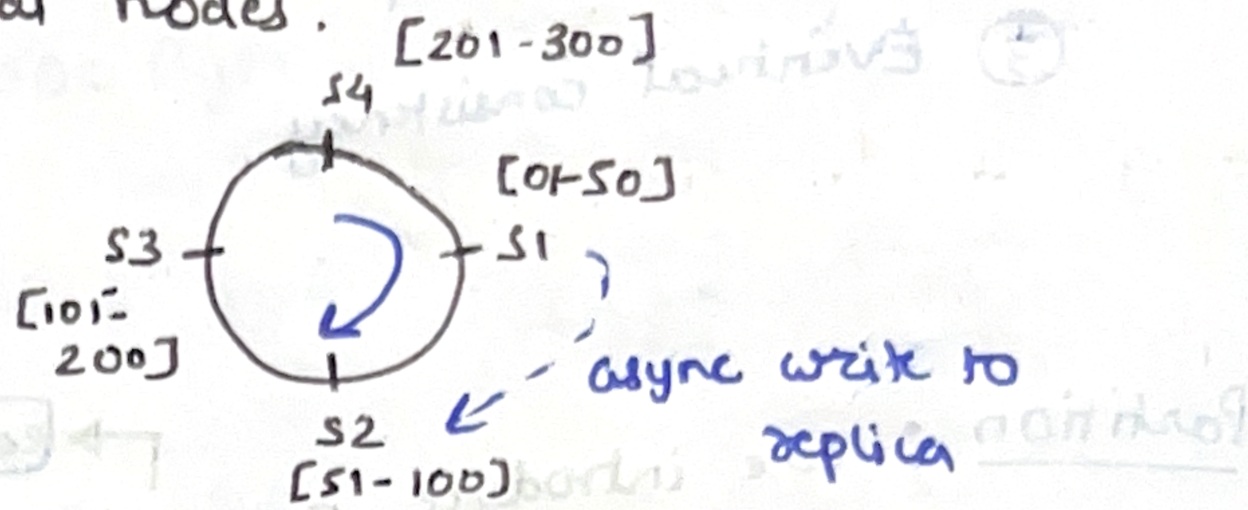
each Key Range Maintains [Preference List]

[1-50] Key Range



⇒ it may not choose the next N-1 servers immediately next to coordinator, Replication nodes are also chosen based on regional preference as well.
& skips virtual nodes.

③ Get & put ops:



Put

Client → Put (CAR) —find Key→ 45 → write into server

[1-50] Range

Pref list : S1, S2, S3 ← Coordinator

we send success response to client when;

no. of server to read from while reading (get req.) → R + W > N ← no. of replicas.

W ↑ no. of server to wait for the async copy to complete

get we already have pref. list for each key Range

there is a LB → request → LB → C.H.

LB → generic LB
LB → Partition aware LB

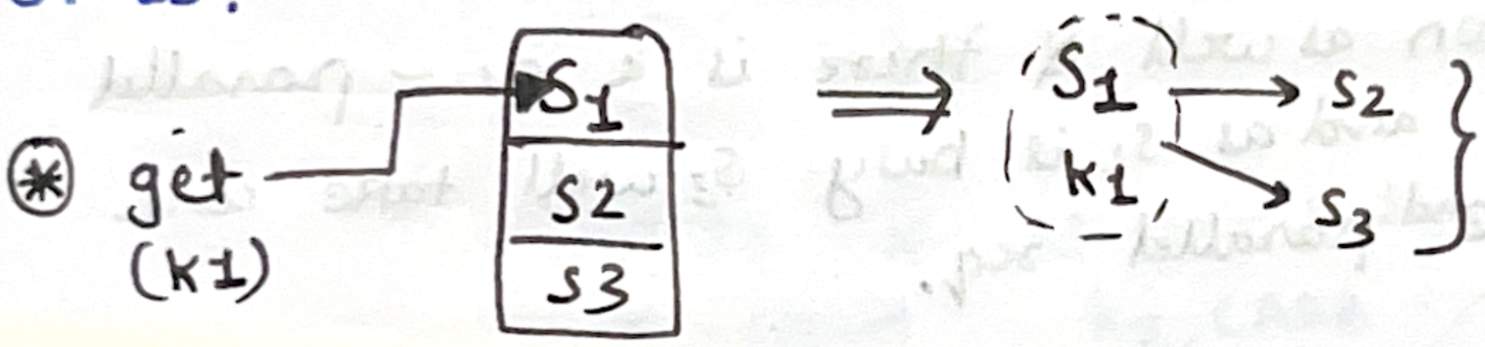
generic LB: get or put request comes to any node, now each and every node is aware of All Key Range's pref. List, so it will take a hop and send request to coordinator. if coordinator is down next one will take the charge from assigned node. (same for get)

- hopping is the main issue, high latency

partition aware: here the LB is aware of the Pref. List and partition so requst will go to coordinator directly.

- low latency because of minimized hopping.

* control will come to coordinator in case of all the type of LB.



will ask other node to get copy of k1 and will wait for R no. of responses from server

⇒ once S1 (coordinator) get all the required replica's (R); it would respond back to client for get Req.

why do we need R & W for above operation; before declaring the operation as success?

⇒ Data versioning will answers that.

④ Data Versioning :

There might be a chance of failures, right?

✱ So it might possible that due to network failure or any othe failure, the keys can get inconsistent as they might not be (yet) updated in other nodes.

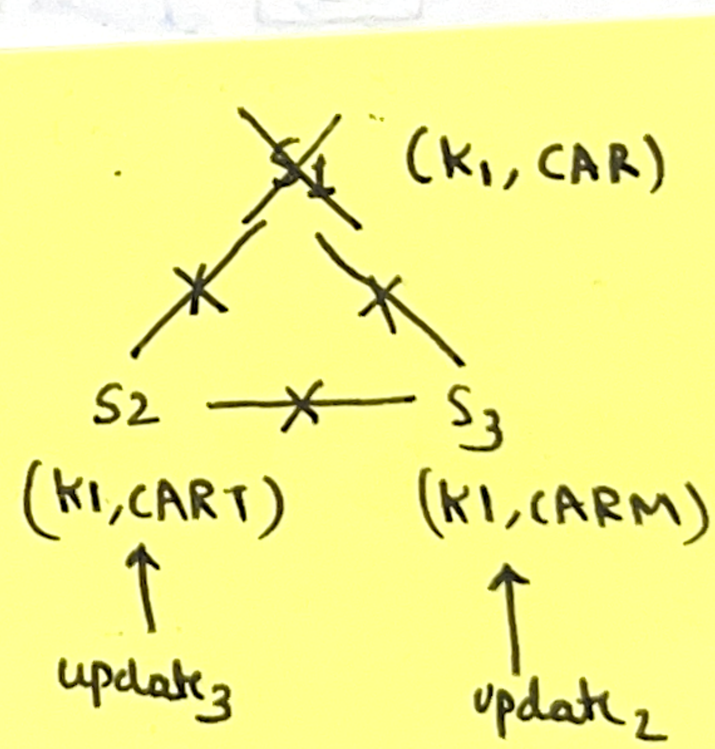
[For example, on second update of same key, out main coordinator S1 goes down, so it would contain non-updated entries.]

↳ more complexity arises when S_2 and S_3 lose connection as well & there is a new parallel request and as S_2 is busy S_3 will take care of 2nd parallel req.

to resolve this, we use something called Vector clock.

each entry in server for key contains additional information

List OF [server, counter]



all 3 are inconsistent.

① put (K1, CAR) → S1 K1, CAR [S1, 1]

S2 — S3

K1 CAR [S1, 1] K1, CAR [S1, 1]

② put (K1, CART) → S1 K1, CART [S1, 2] → we increase counter

S2 — S3

K1, CART [S1, 2] K1, CART [S1, 2]

Now node S1 goes down & there is partition bet^n S2 & S3.

③ put (K, CARM) & put (K1, CARR) in parallel :

~~S1~~

put → S2 --X-- S3 ← put

K1, CARM [S1, 2] [S2, 1] K1, CARR [S1, 2] [S3, 1]

Now if we get (K1) once S1 is back as well;

get (K1) → S1 ← S2, S3

S1 [S1, 2]

S2 [S1, 2] [S2, 1]

S3 [S1, 2] [S3, 1]

} get will send this ^ to client

Last writ win

both

K1 CARM & K1 CARR

↓ client has Algo for conflict res & will do put req.

stale state

S1 K1 CARR [S1, 3] ...

S2 — S3

K1 CARR [S1, 3]... K2 CARR [S1, 3]...

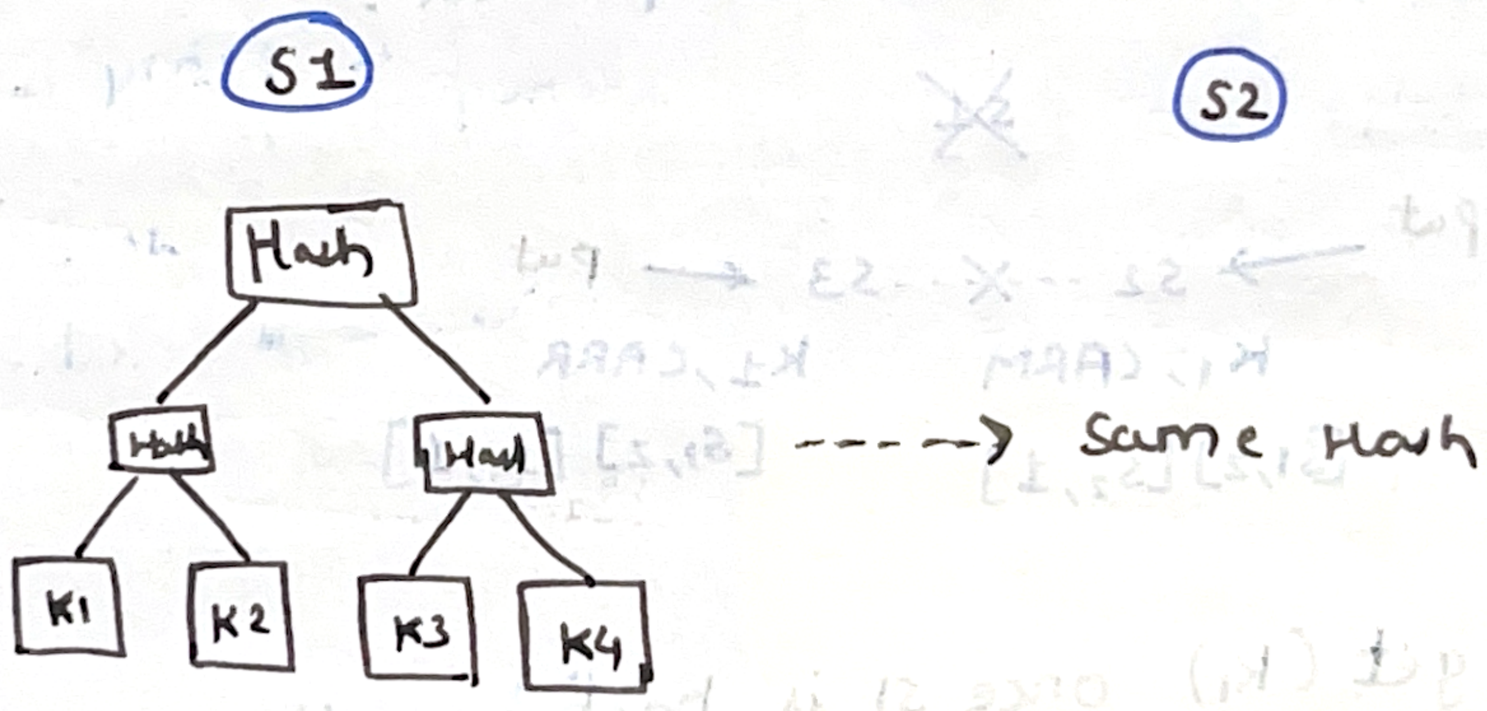
put (K1 CARR) → S1 [K, CARR)

thus in case of failure we would get eventual consistency.

(5) <u>gossip protocol</u>: each server in the pool is aware about if any of the other server node is down, they maintain List of servers & availability & refreshes list by sending heartbeat every sec.

- node is considered down if more than one servers notices & reports failure heartbeat report.

(6) <u>Merple Tree</u>:



to check sync betn the server (comparing millions of record) merple tree starts from Top and figure out the inconsistencies.

in case of failure, Replica is promoted to become coordinator.