No-SQL DB (Amazon Dynamo Db & Casandra & MongoDB)

Key-Value Store.

DB

SQL No-SQL

SQL : Relational DB (Rows & Cols)
ACID : Consistentcy attain.

No-SQL : Non - Relational Data
Key-Value pairs
Key --> String
Value --> String, List, Object.
BASE : Availibility Attain.

CAP : Trade off b/w C and A.

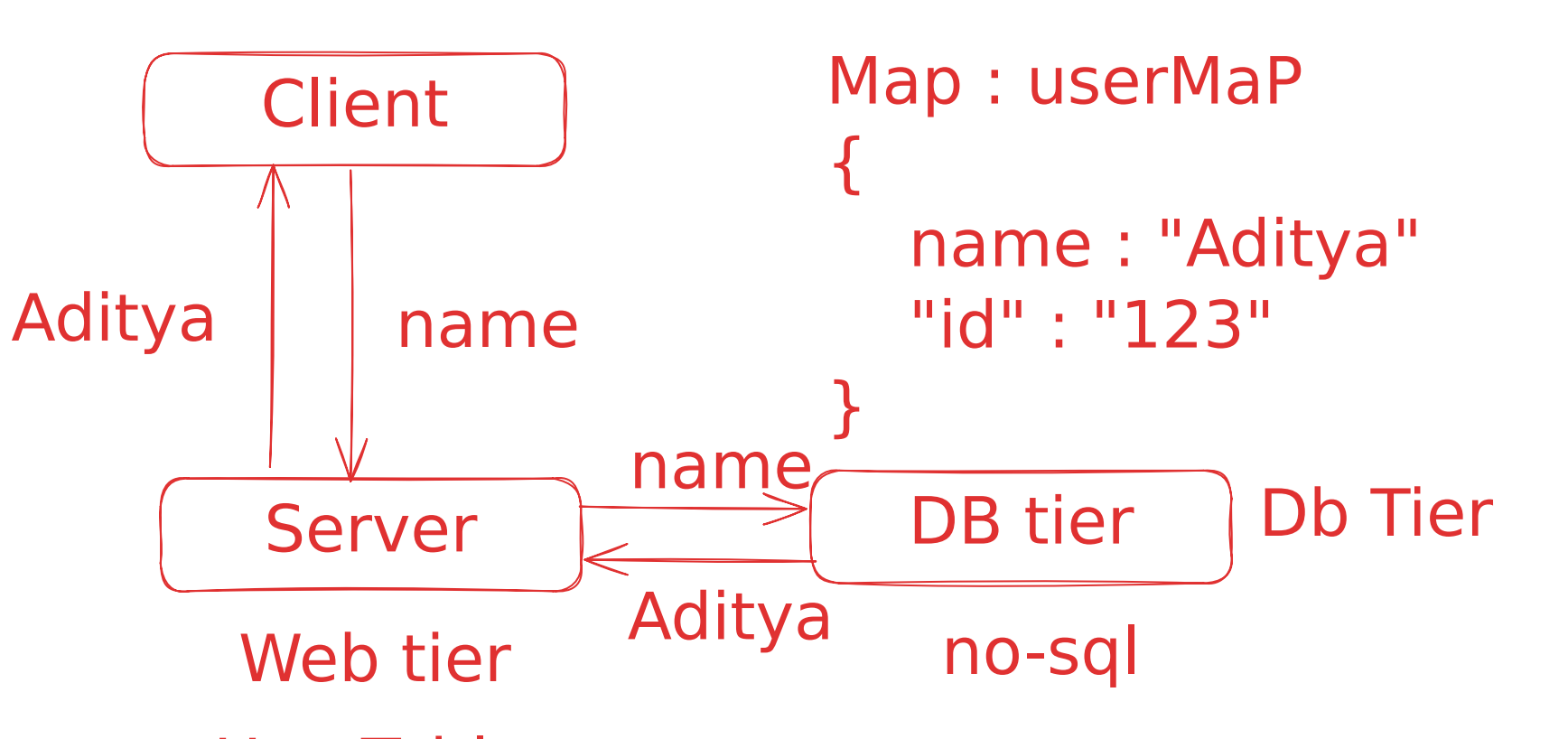
Key --> Value pairs.
String, Object, List,...
String --> Plain string, hashed string

Amazon Dynamo Db (Key-value)

Casandra DB (Column Based store)

Mongo DB (Document based store)

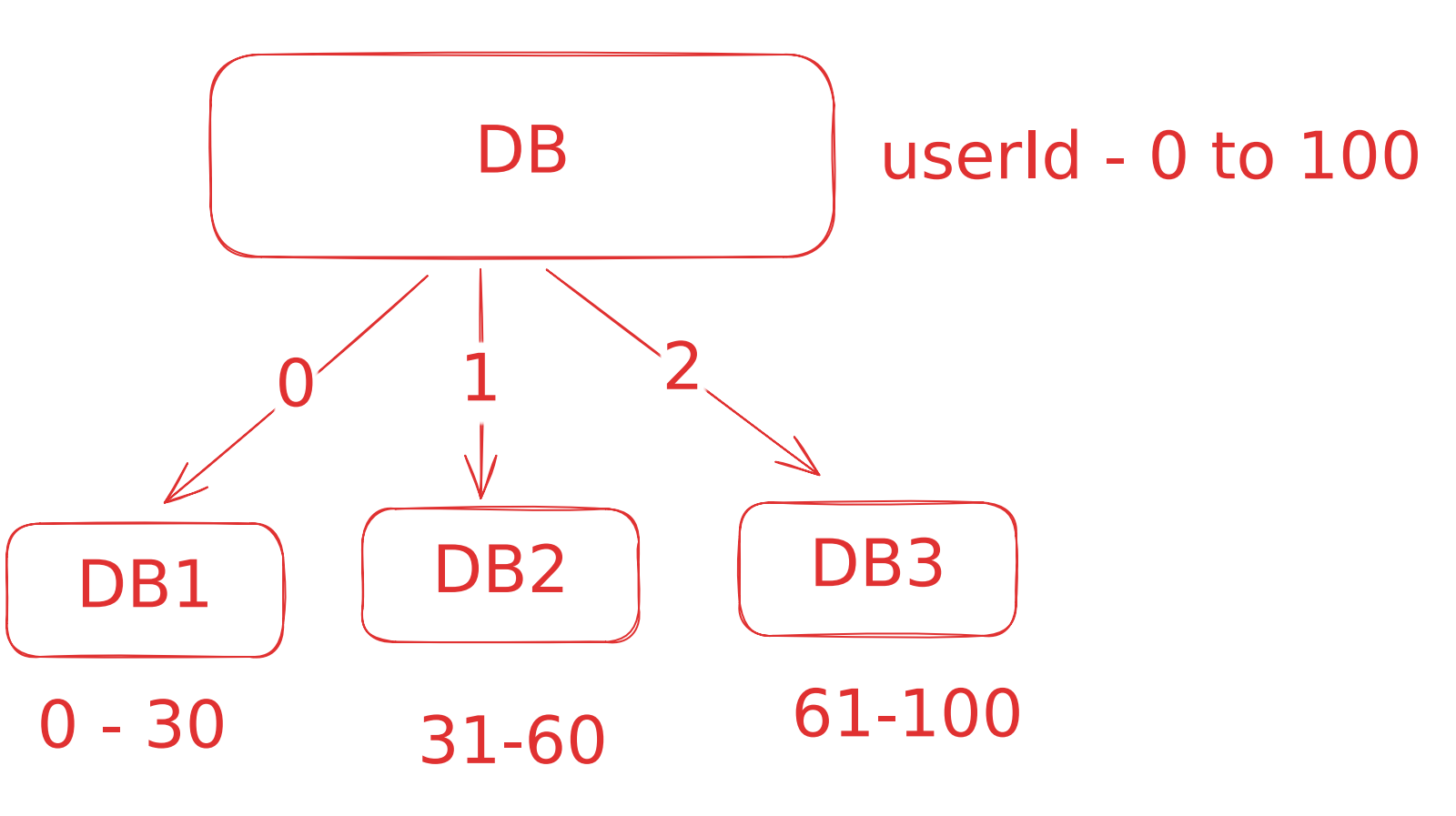
Creating our own Key-Value Store.



Map : userMaP
{
name : "Aditya"
"id" : "123"
}

HastTable
Key : Value
name : Aditya

Horizontal Scaling in DB ---> Sharding

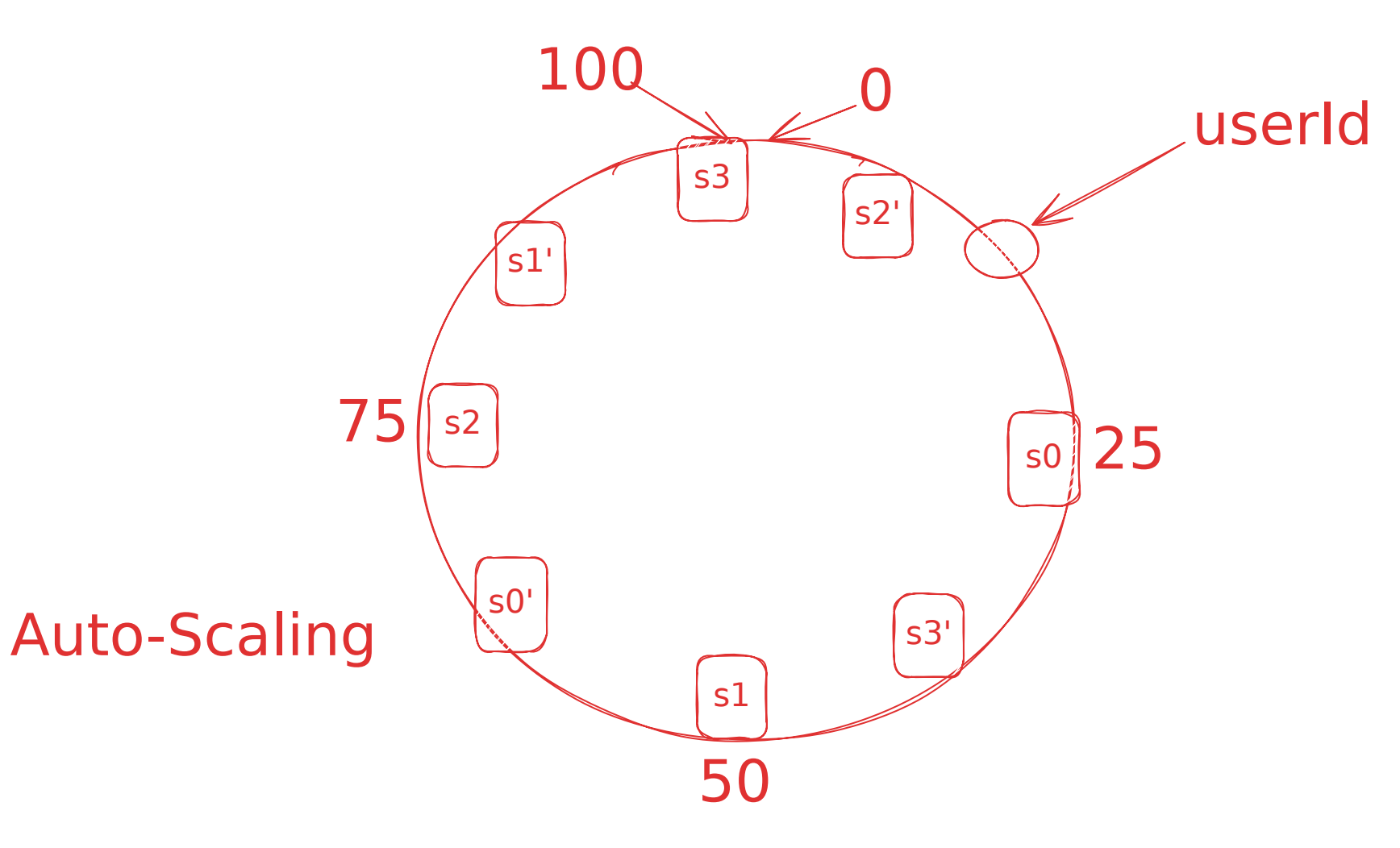


Modulo Approach

like DB migration and all check DB sharding video

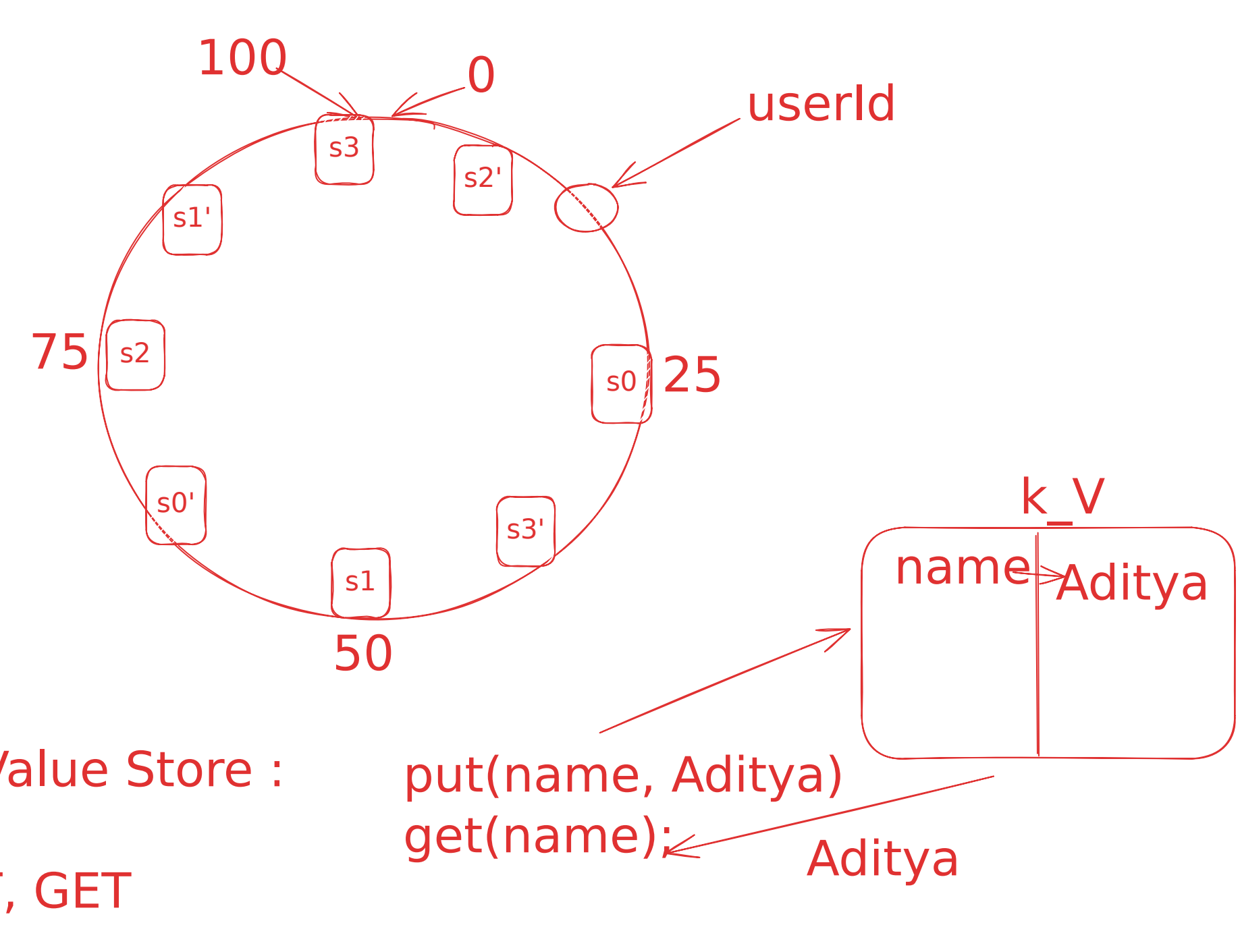
problems : Scaling was difficult.
Auto Scaling was not possible.

Consistent Hashing



Auto-Scaling

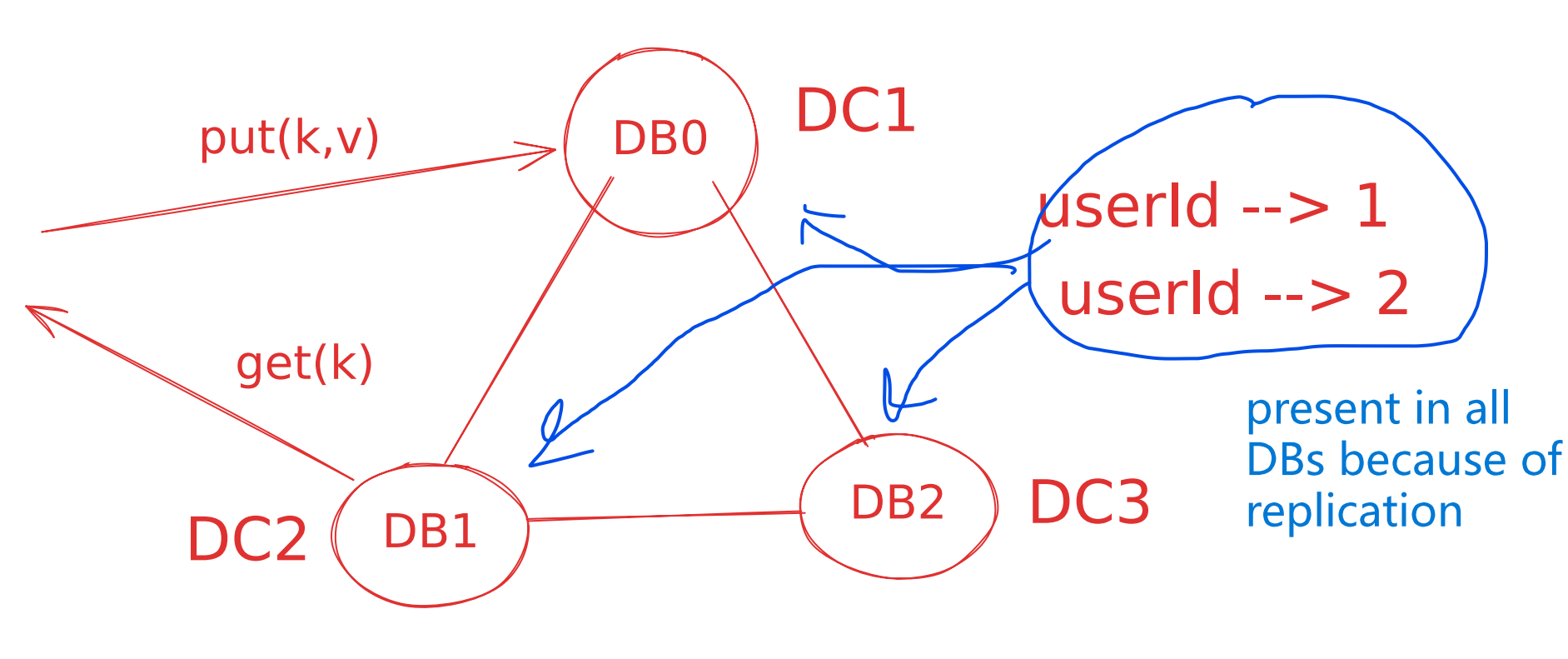
Key-Value store (Distributed)



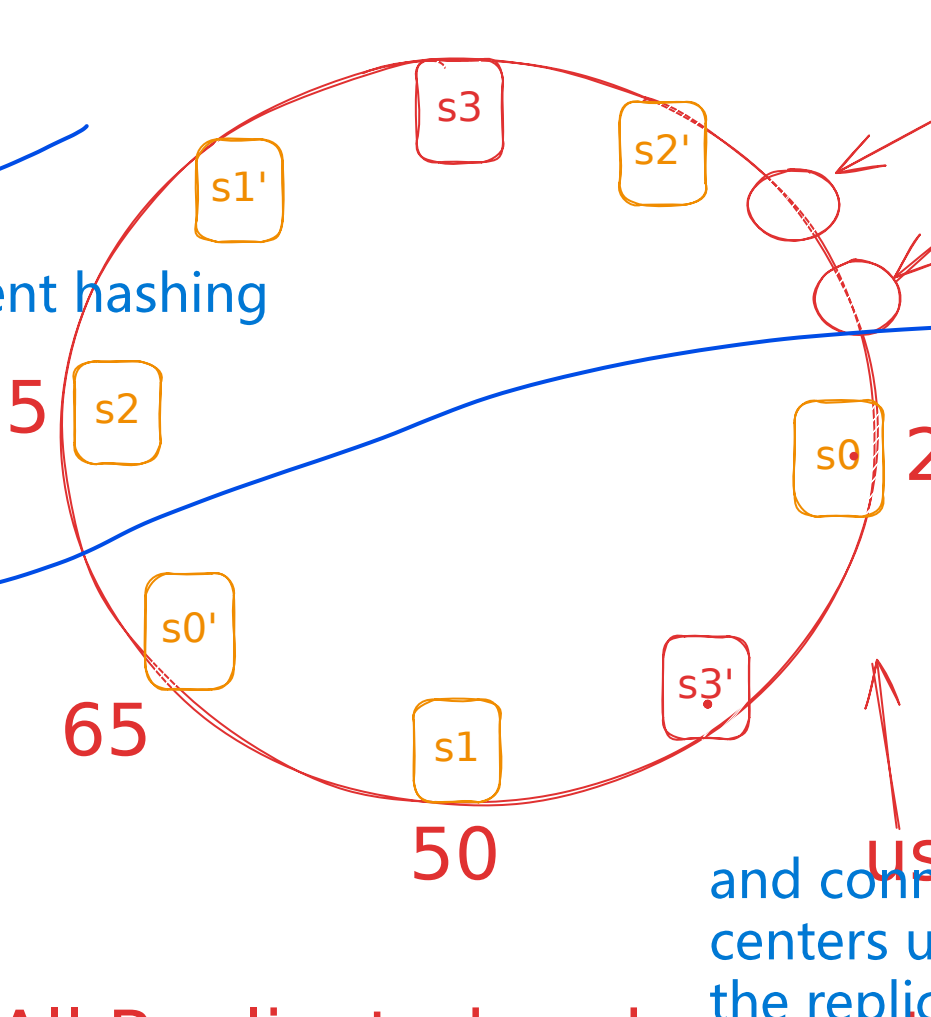
Key-Value Store :
put(name, Aditya)
get(name);
Aditya

POST, GET

POST : put(key, value) --> 200 ok
GET : get(key) --> value



Data Replication (MOngo, Casandra Dynamo)



userId = 1
userId = 2
N = 3

why do we need to replicate since we are sharding the DB in consistent hashing

DB = no.of non virtual servers that data replication has to be done

here replication happens only in non virtual servers

userId = 3

and connect server from different data centers using fast internet connection to get the replication

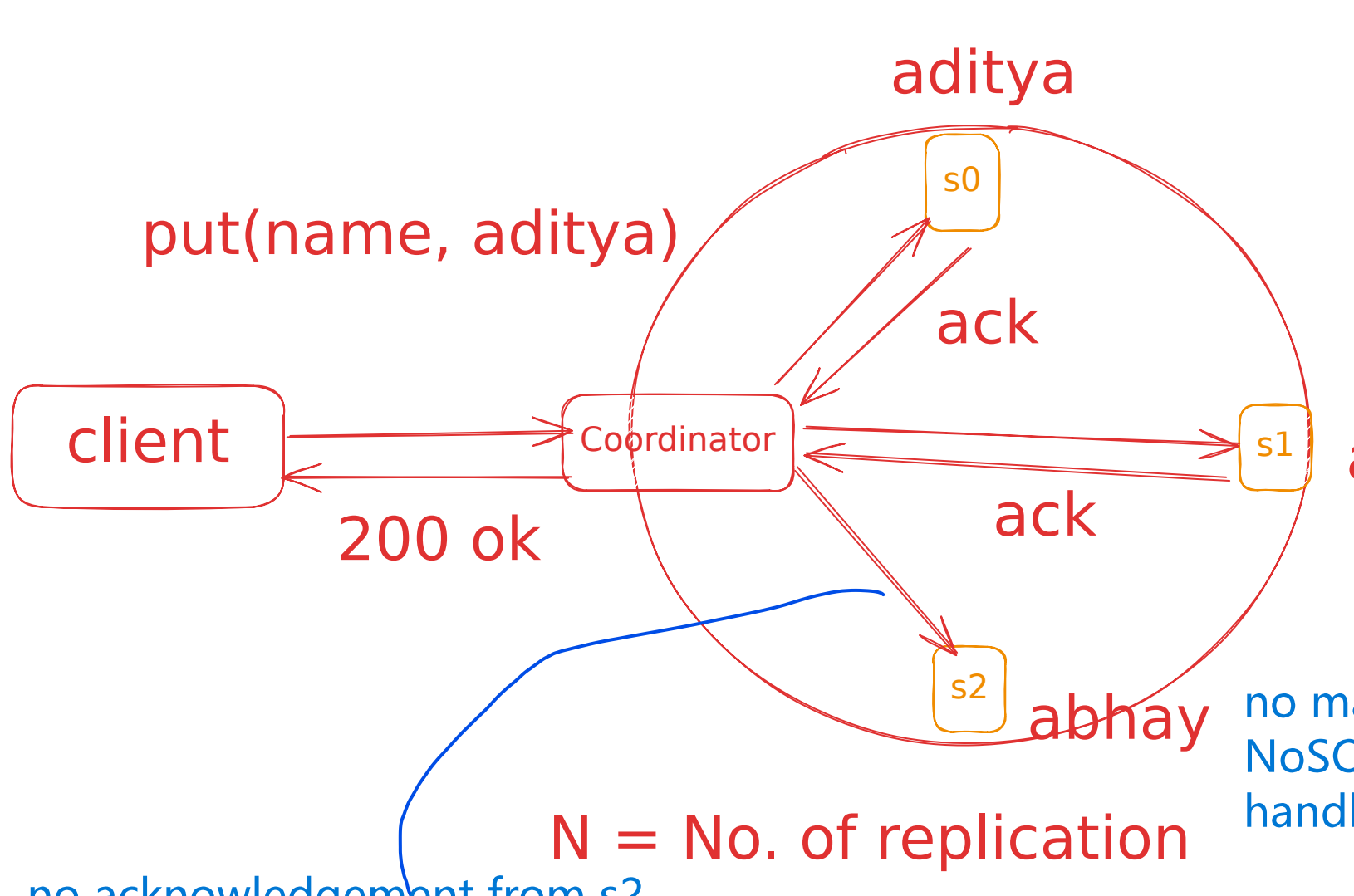
s1,s2,s3 → All Replicated nodes must lie in diff data centers.

i.e. if two servers lies in the same data center and we replicate these servers and somehow if the data center is down then now no use of replicating

Replication --> Same data in diff DBs.

in the design of these DBs(no sql) some of the non virtual DBs are (not all) are also replicated hence two or more DBs can hold same data

Achieveing Consistency



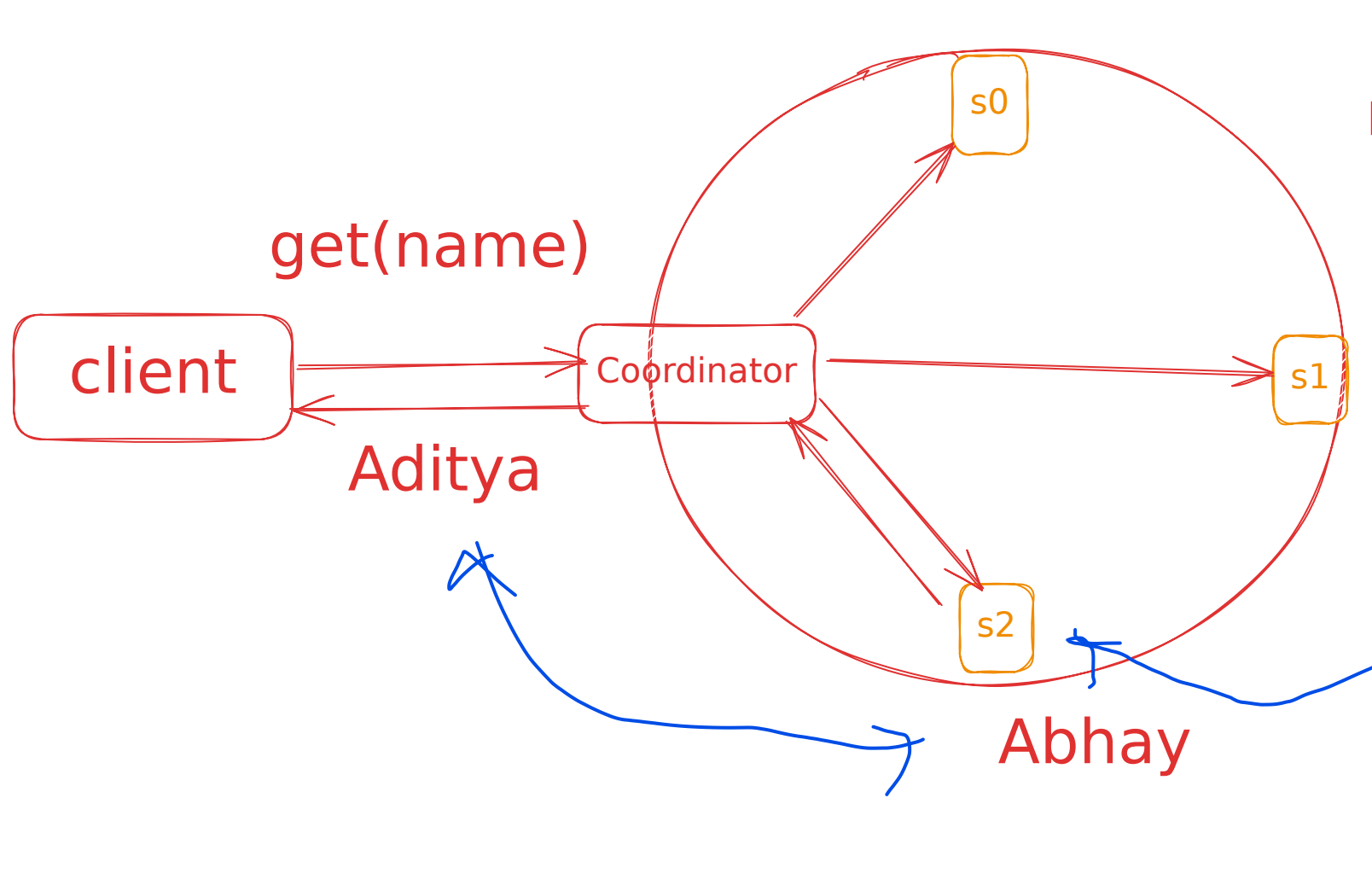
W : 2

no master slave architecture in NoSQL architecture, coordinator handles it

N = No. of replication

no acknowledgement from s2

Quoroum Census
N = 3
R : Read Quoroum (R = 1)
W : Write Quoroum (W = 1)
How much Ack is needed from servers.



N : No. of replicas
R : Read Quorum
W : Write Quorum

N, R, W --> Trade off b/w Consistency and Latency. consistency directly propotional Latency

slower writes

R = 1, W = N : System is optimized for faster Reads.
W = 1, R = N : System is optimized for fastert writes.
W + R > N : Strong Consistency.
W + R <= N : Weak Consistency.

in this case collision occurs since two servers may have different response

may read unupdated dat

coordinator handles the collision case

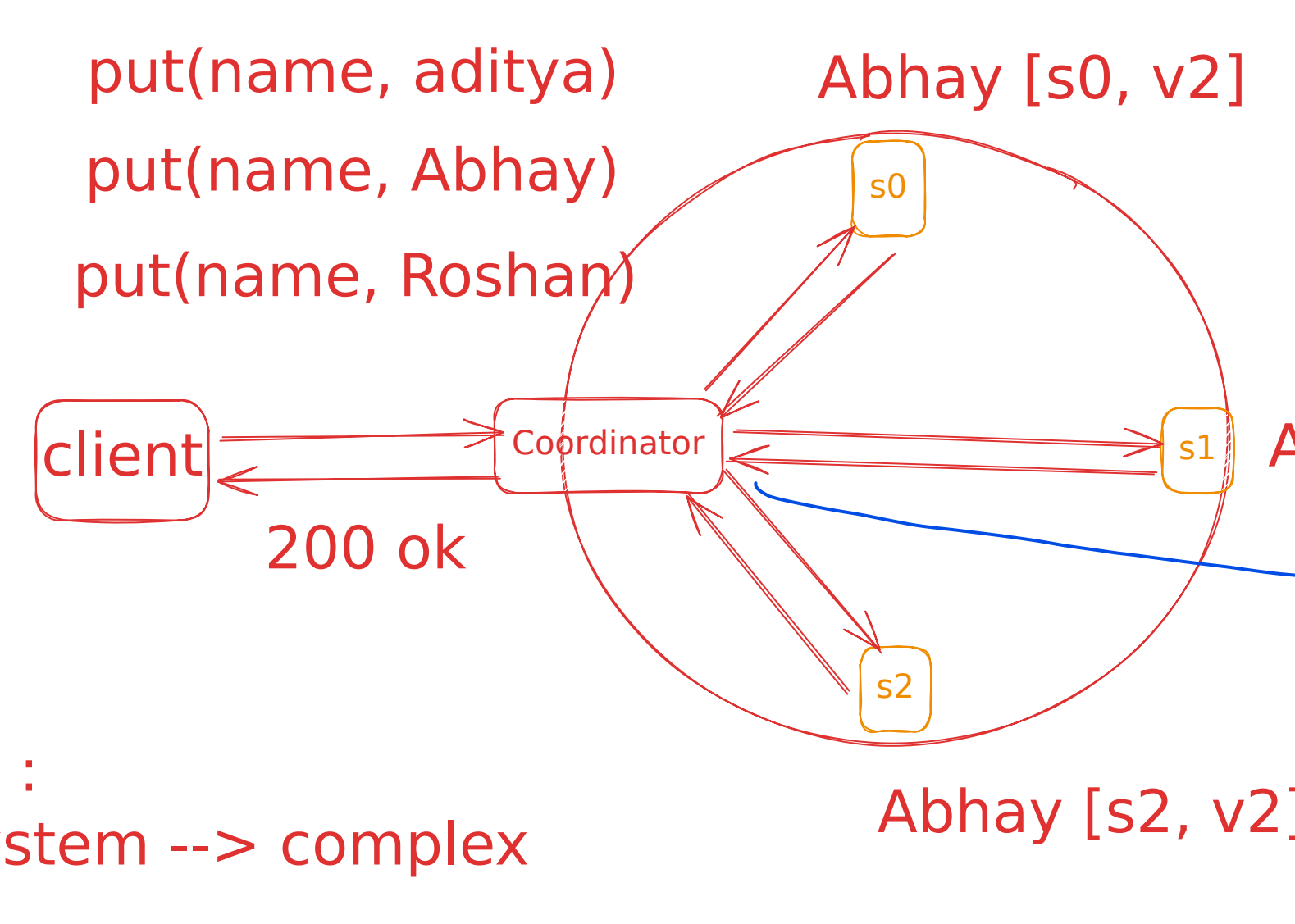
CAP : C & A --> Eventual Consistency

can be achieved

Consistency Models:
1. Strong Consistency : Client never sees wrong data
2. Weak Consistency : We prefer A over C. Client may see wrong data sometimes.
3. Eventual Consistency : We prefer A over C. Client may see wrong data sometimes, but given enough time, the consistency will be achieved.

MongoDb, Casandra and Dynamo all prefer Eventual Consistency.

Balance check --> Db (Read / Log DB) : Eventual
Credit/Debit --> Db (Write Intensive) : Strong



N : 3
W : 2
R : 1

after responding to the client ,coordinator has its own log file which is updated

Cons :
1. System --> complex

Log file :
d0 --> v1
d0 ---> v2

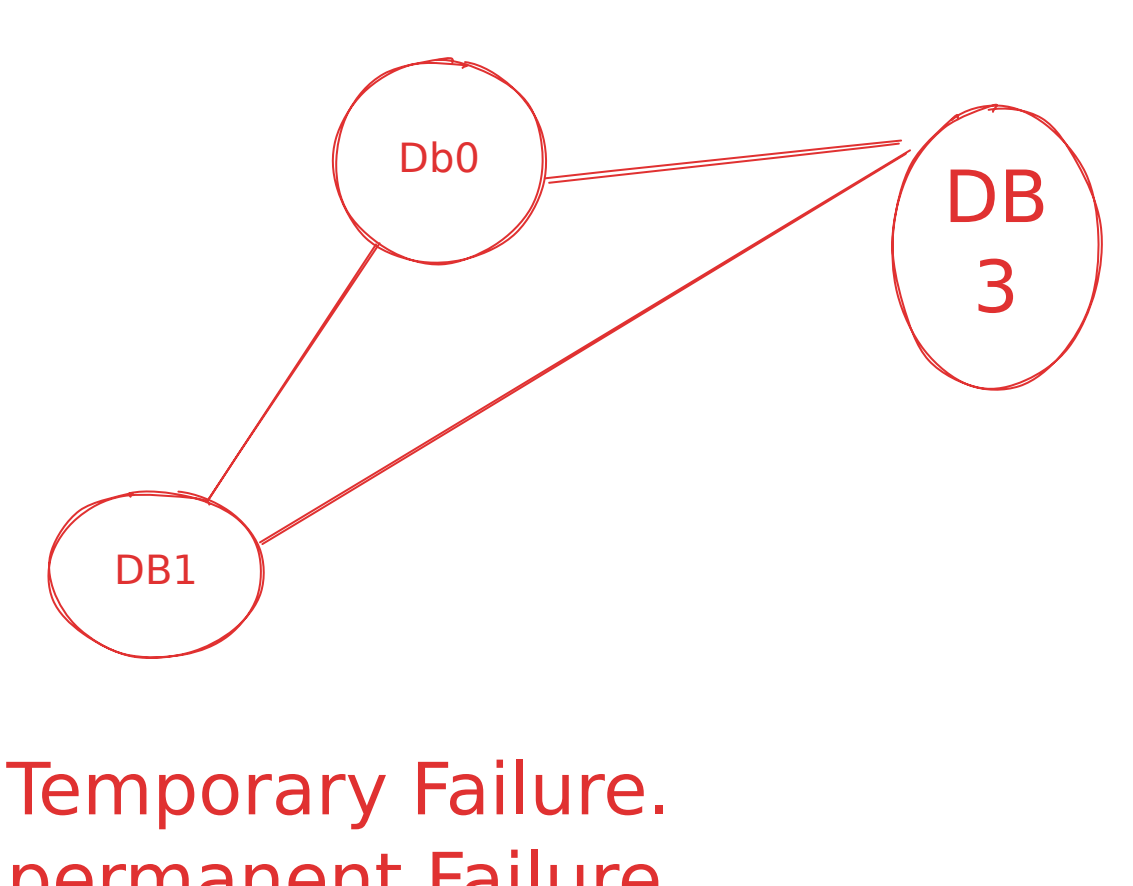
when r=1

Versioning :

Vectors --> [server, version]

1. Mongo, Casandra , Dynamo --> Client (put)
2. Wrong data --> send to client and then check log file and checks the version of the data from the responded server and updates the version comparing with other server data

Handling Failures:



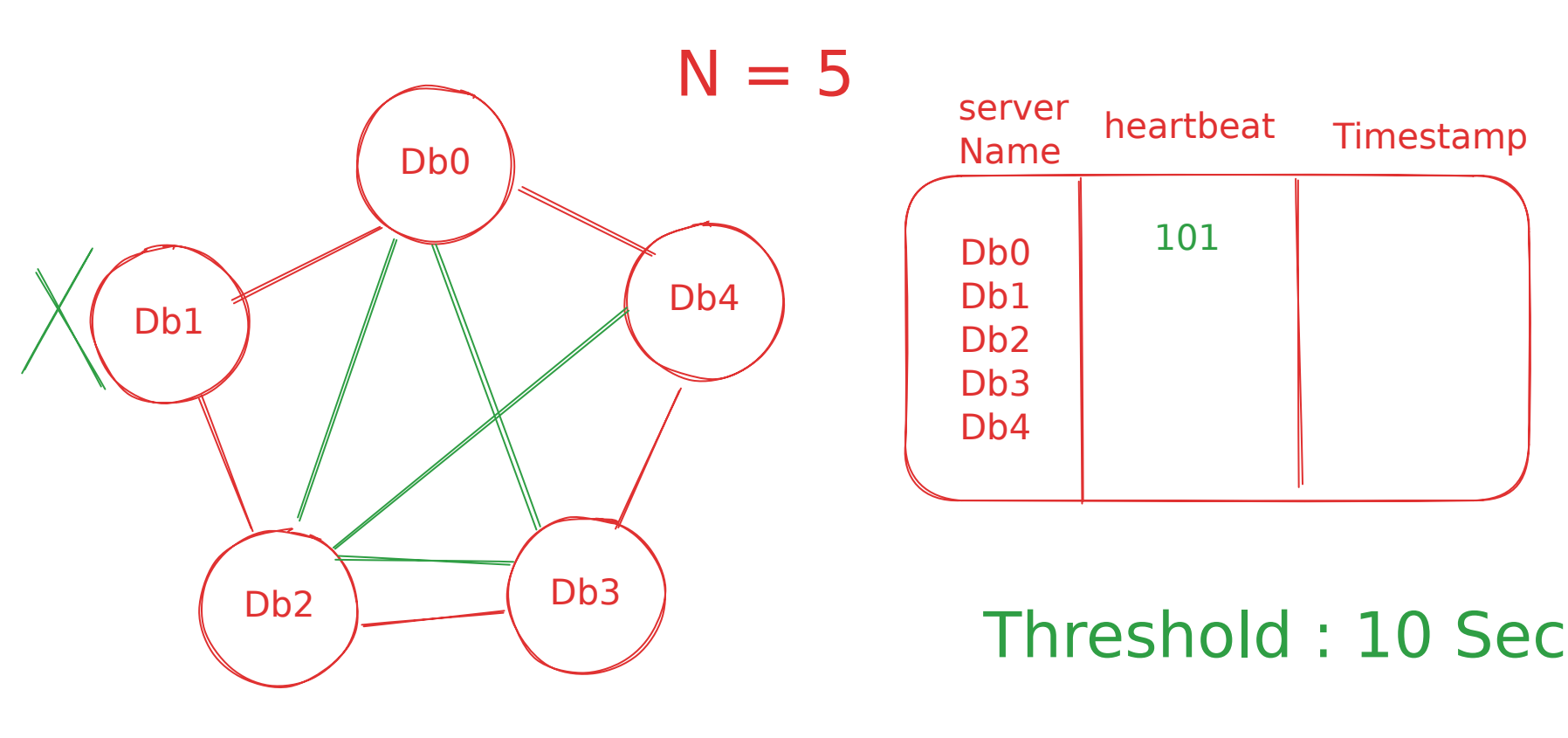
1. Temporary Failure.
2. permanent Failure.

In case of Temp failure:
Assign next server in hash ring for temporary data push/pull.

When server is up again --> Migration is needed.

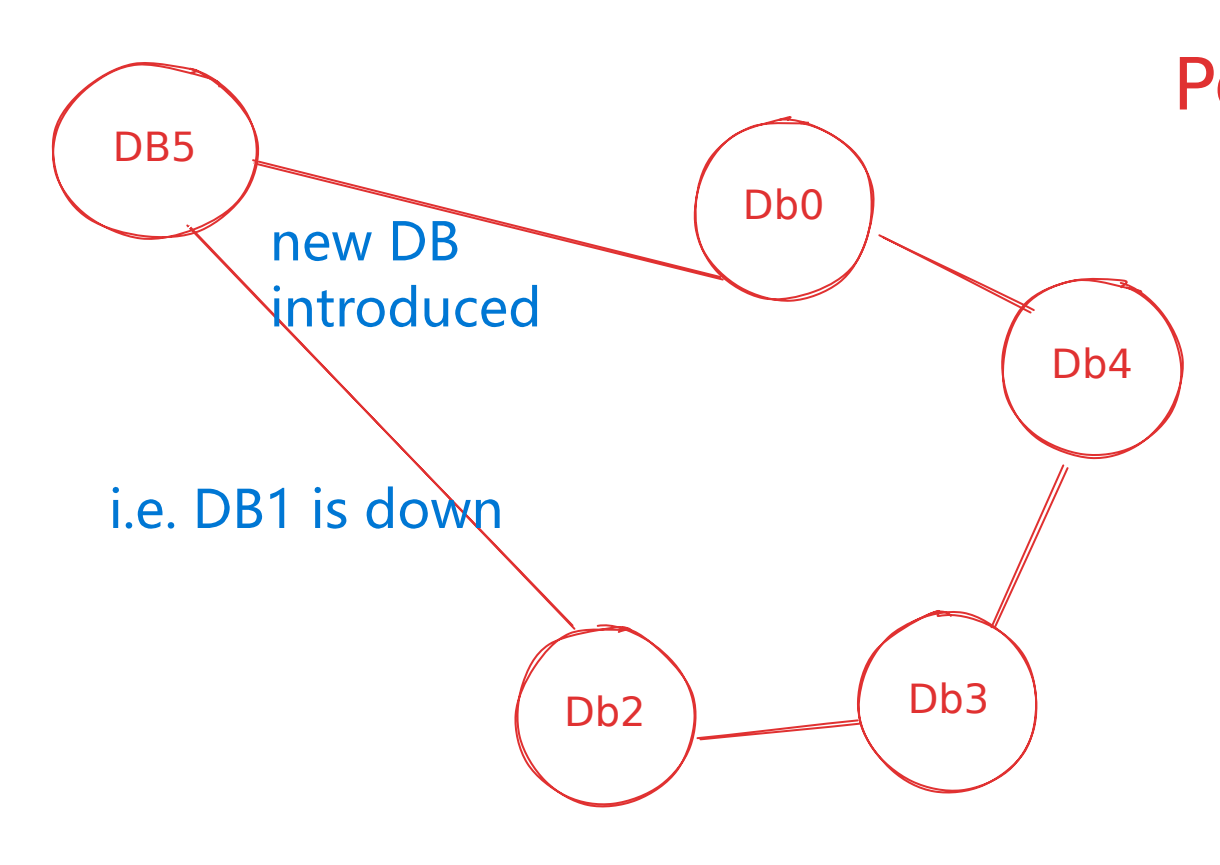
In case of permanent failure:
Create new DB Server & migrate old data. and more data to migrate this time

How to detect Failure ?
-> Gossip Protocol



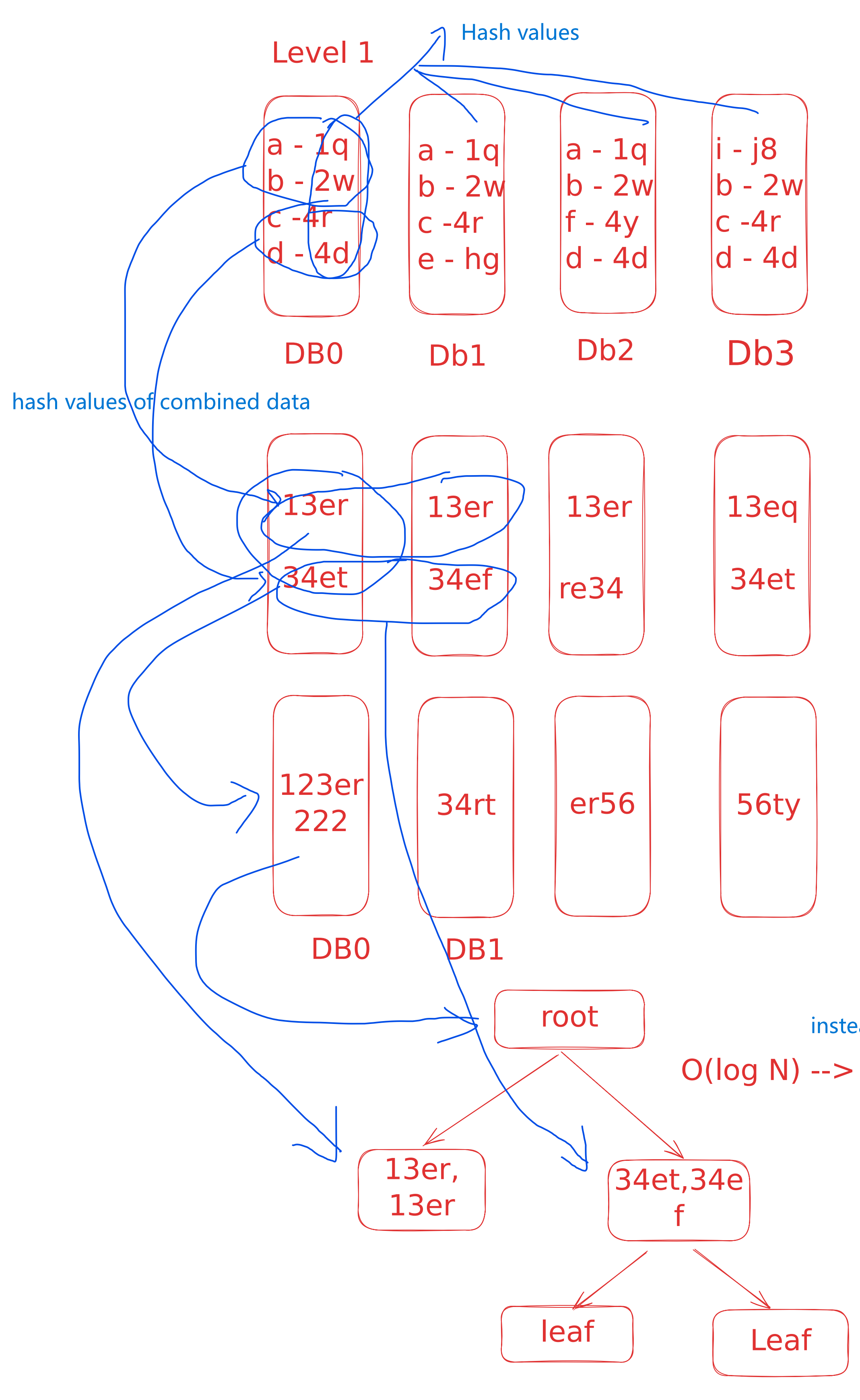
Db 1 --> Mark as down.

2. Markle Protoco



Permanently Down

Eventual



Internally Casandra, Mongo, Dynamo

Memory Cache,
SS tables, (k1,v1), (k2,v2)
Bloom Filters