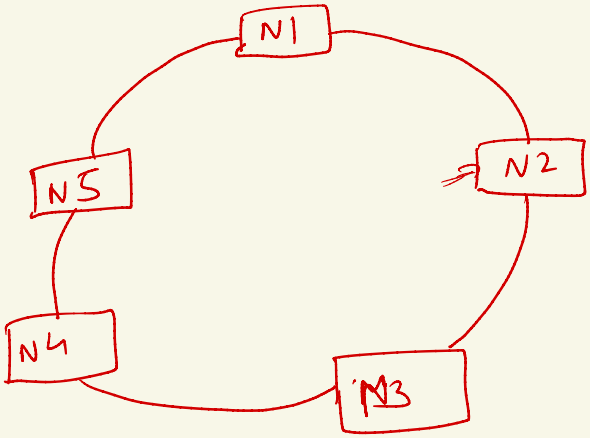
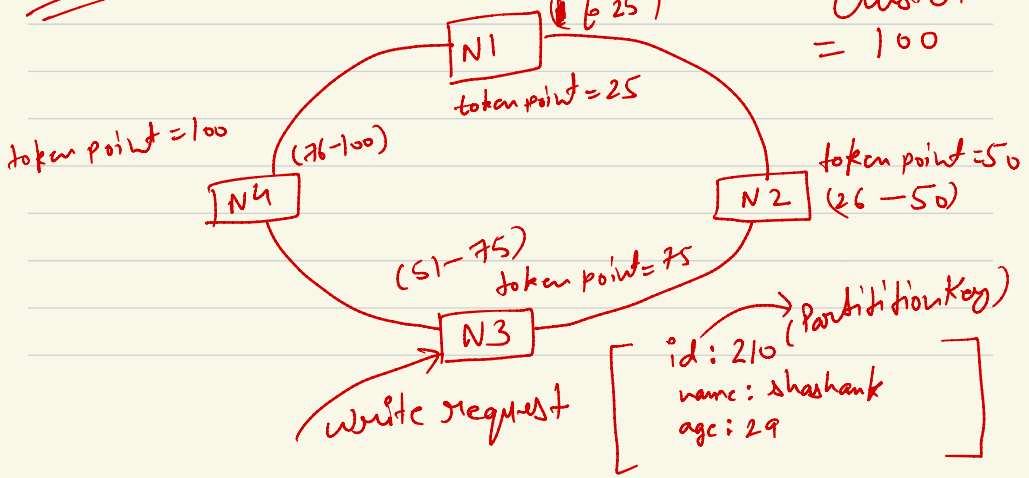
How data is partitioned in Cassandra



Token ⇒ A hash value generated for a given Partition Key.

Token Range ⇒ Each node in Cassandra will be given a range of token values.

Example: Total number of token served by Cassandra Cluster = 100



Token = hash(Partition_Key)

↳ (Partition_Key) % total_tokens

Token = {0 --- 99}

id = 210 , Token ?

Token = 210 % 100 = 10

Cassandra follows MurMur3 hash algorithm to generate tokens.

Hive

hashfunct = f(Partition_Key) % 5

= len(INDIA) % 5 = 5 % 5 = 0

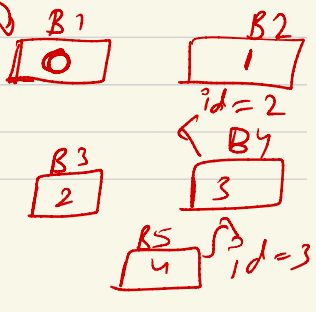
= len(CHINA) % 5 = 0

len(USA) % 5 = 3

id	Country
~~1~~	~~INDIA~~
2	USA
3	INDIA
4	INDIA
5	CHINA
6	CHINA

id=1, Partition_value = 0

id=3



Collision => if same hash value got generated for different values

In Cassandra Token range

between

-2^{63} to $+(2^{63}-1)$

→ 8 Byte long Integer Value

64 bits

$(0-2^{64})$

-1025964238 2 to 1025964238 1

Total Token Range

let say we have 4 Node in cluster

for each Node = {0 - 99} / Number of Nodes

token Range

OR

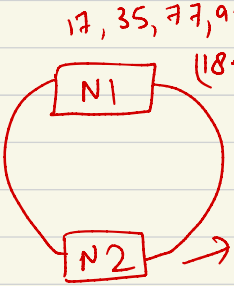
= (Total token Range) / num_tokens for each node

num_tokens = Number of Ranges each Node can handle

Node = 2

num_tokens = 4

token_range = 0-100



Consistency Levels in Cassandra

Q.) based on CAP theorem Cassandra lies where?
↳ AP (Available, Partition Tolerance)

Q.) in Cassandra (Yes) Can we impose consistency?

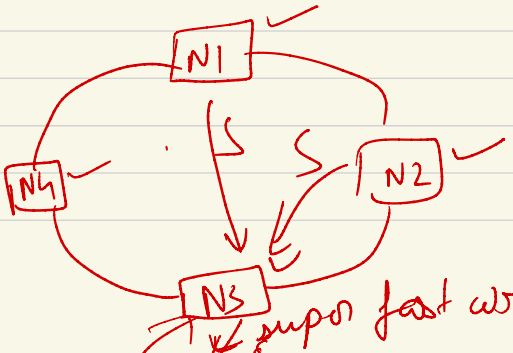
Q.) what will be the impact if we apply consistency? (Speed or performance)

What happen for consistency?

① write → if consistency not maintained the write operation will fail

② Read → if consistency not maintained then Read will fail

RDBMS is there any intermediate state?

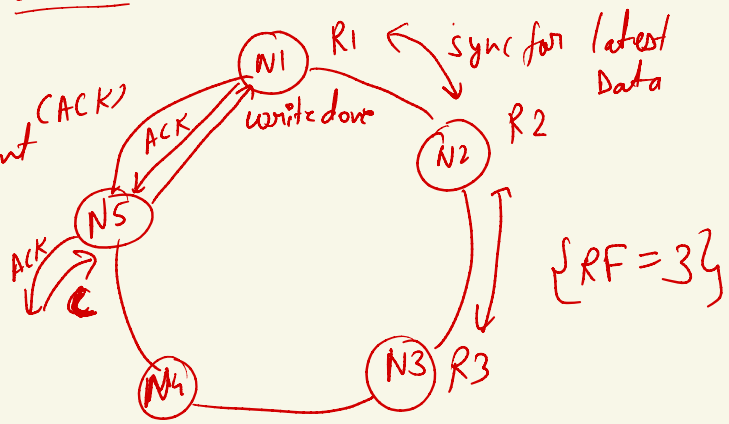


super fast write operation

write request (S)
(forcefully write it for ALL)

Consistency for write operations

↳ Based on acknowledgment from Replicas



→ ONE : it needs acknowledgement from only one Replica Node.
in this case data will be synced asynchronously (from backend) on remaining Replicas.

→ TWO : it needs acknowledgement from 2 replica nodes.

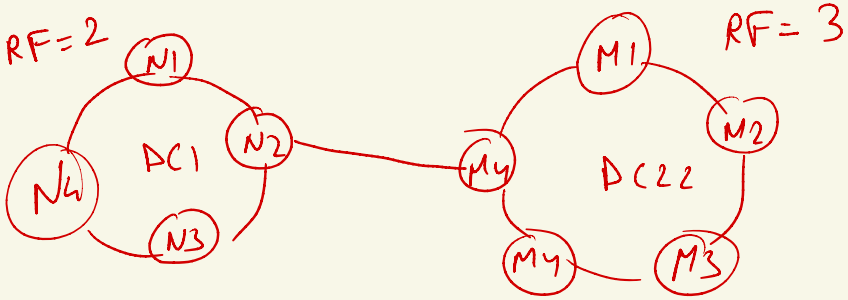
Quorum Calculation formula = (sum of Replication factor / 2) + 1 (majority)

if we have 100% of something then what least value we need for majority? (51%)

arr of (10) = | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

$= (n/2 + 1) = (10/2) + 1 = 5 + 1 = 6$

: QUORUM ⇒ needs acknowledgement from 51% or majority of replica nodes across all the data centers.



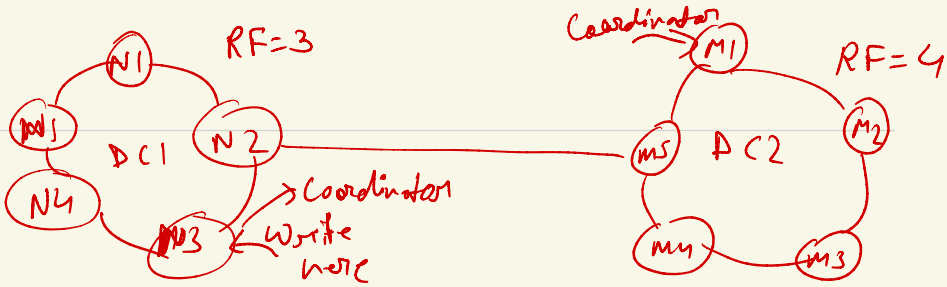
Value of QUORUM = $(2+3)/2 + 1$

$= (5/2) + 1$

$= 2 + 1$

$= 3$

: LOCAL_QUORUM → need acknowledgement from 51% or a majority of replica nodes just within the same data center as the coordinator.



LOCAL_QUORUM

for DC1 = $3/2 + 1$

= 2

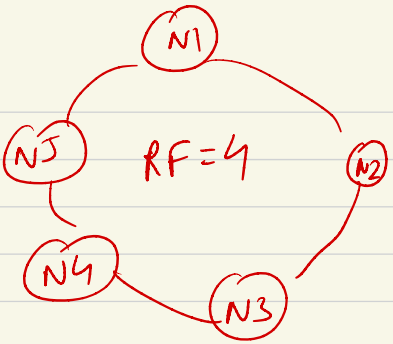
LOCAL_QUORUM

for DC2 = $4/2 + 1$

= 3

: ALL → needs acknowledgement from all replica nodes

So it need ACK from all 4 Replicas



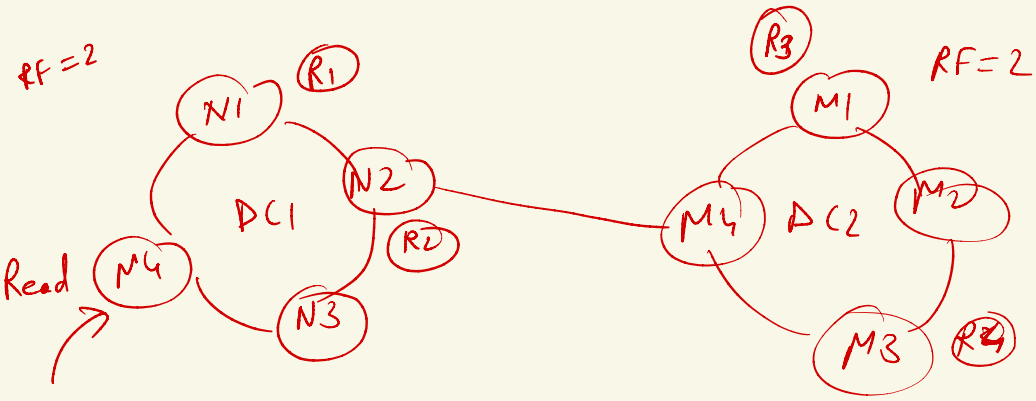
Ex: Update sales using Consistency Quorum

Set total_purchase = 100 where cust_id = 4;

Consistency for Read Operation

: ONE → Only one replica node returns the data.

: QUORUM → returns the data from a quorum with the most up to dated data.



{cust_id=200}

QUORUM = 3

↳ Output from R1 → Last update happened on 20-NOV-2022 10:00PM

↳ Output from R2 → 20-NO-2022 9:00PM

↳ output from R3 → 20-NOV-2022 11:00 PM

: LOCAL_QUORUM → Returns a result from LOCAL_QUORUM with latest data and follow coordinator node for local quorum value.

: ALL → reads data from all Replica nodes and

Returns latest record based on timestamp.

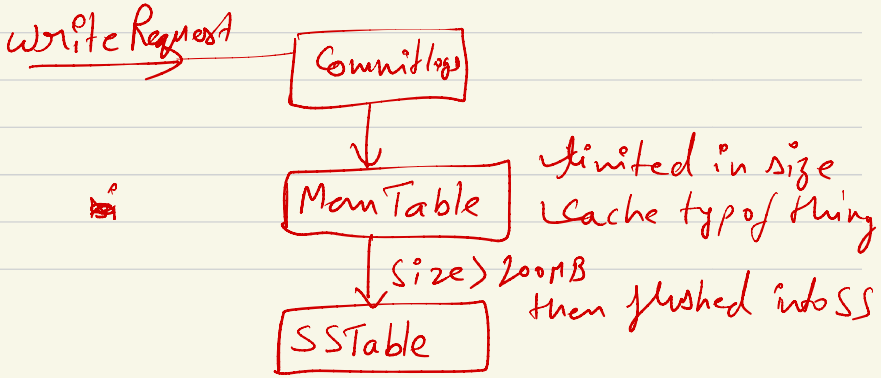
Ex: Select total_purchase from sales
using consistency ALL where customer_id=5;

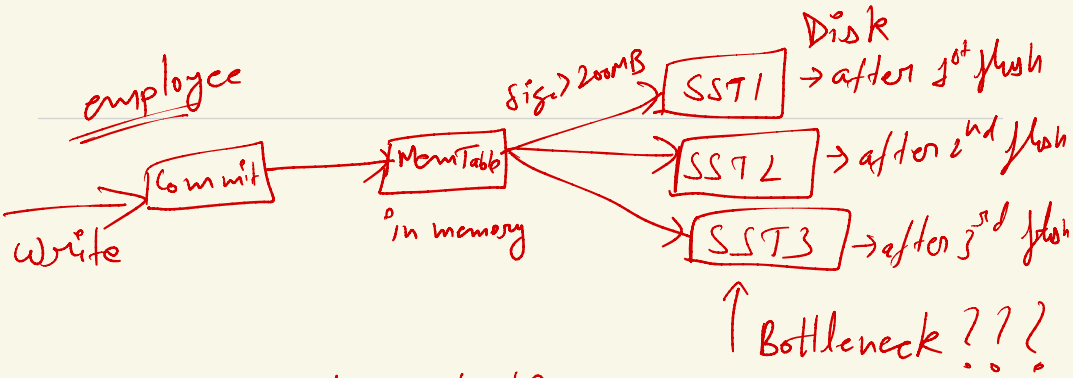
Write Path in Cassandra

→ Commit log ⇒ transaction logs will be written here for crash recovery. Non-volatile, so if power failure happens still we can recover.

→ Memtable ⇒ memory cache to store the copy of the data.

→ SSTable ⇒ the final ~~destination~~ destination where actual data gets stored in the disk.

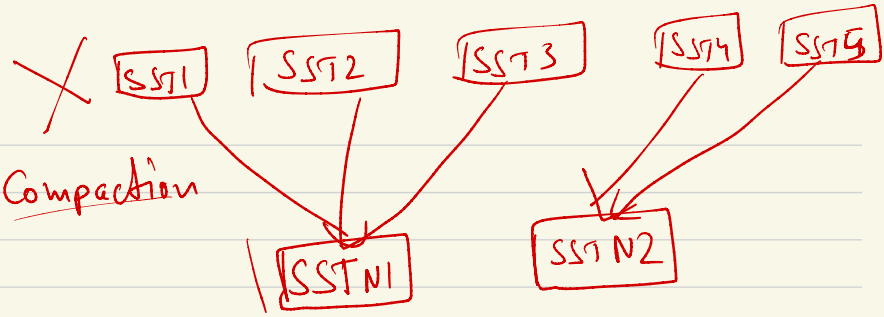




time to time

↓

Compaction Happens in Cassandra for SSTables



Write Path

1. Cassandra appends writes in commit log for durability.
2. Data then will be written in a in memory cache known as Memtable
3. The memtable store writes in a sorted order until reaches the

memory threshold value.

④ Then it is flushed to Sorted String table called as SSTable. Write operations are atomic on Row Level, means values for each column of that Row will be written/updated or none will be written.

Read path in Cassandra

① Cassandra first check if the data is available in the memtable

② if not found then data will be read from SSTable.

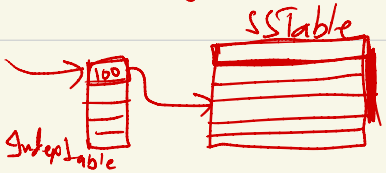
③ To optimize the reads

↳ Cassandra uses Bloom filter technique to minimize the data scan.

↳ Cassandra uses indexing on SSTable

↓

metadata for Actual Data



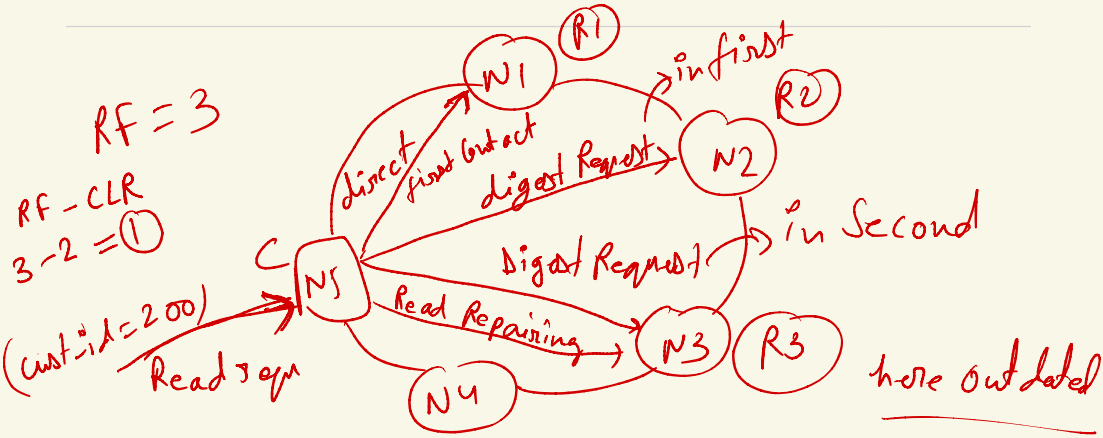
↳ Compaction also helps to improve reads

Types of Reads coordinator can send to Replica?

- ↳ Direct Read Request
- ↳ Digest Request
- ↳ Background Read Repair Request

Direct Req:
Coordinator node contacts one Replica node, after that digest request will be send to the remaining replica nodes based on consistency level. The digest request will check if data on the replicas are upto date or not. Then coordinator will send the digest request to remaining nodes.

If any replica node have out of date data, a background read repair request will be sent. Read repair request ensure that the requested row(data) is made consistant on all replicas involved in a read Query.



Select with consistency Quorum

= 2