VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui

Cassandra Repair Mechanisms

Lecturer* : *Dr. Pavle Mogin

SWEN 432
Advanced Database Design and Implementation

Cassandra The Fortune Teller

Plan for Repair Mechanisms

- Hinted Handoff Writes
  - How Hinted Handoff works
  - Hinted Handoff and the Consistency Level
- Background Read Repair
- Anti-Entropy Node Repair
  - Merkle Trees
  - How node repair works

Prolog

- Cassandra built-in mechanisms are techniques to over come inconsistency of nodes due to node failures
- Mechanisms:
  - Hinted Hand-Off,
  - Background Read Repair, and
  - Node Repair, also called
    - Anti-Entropy Repair, and
    - Replica Synchronisation

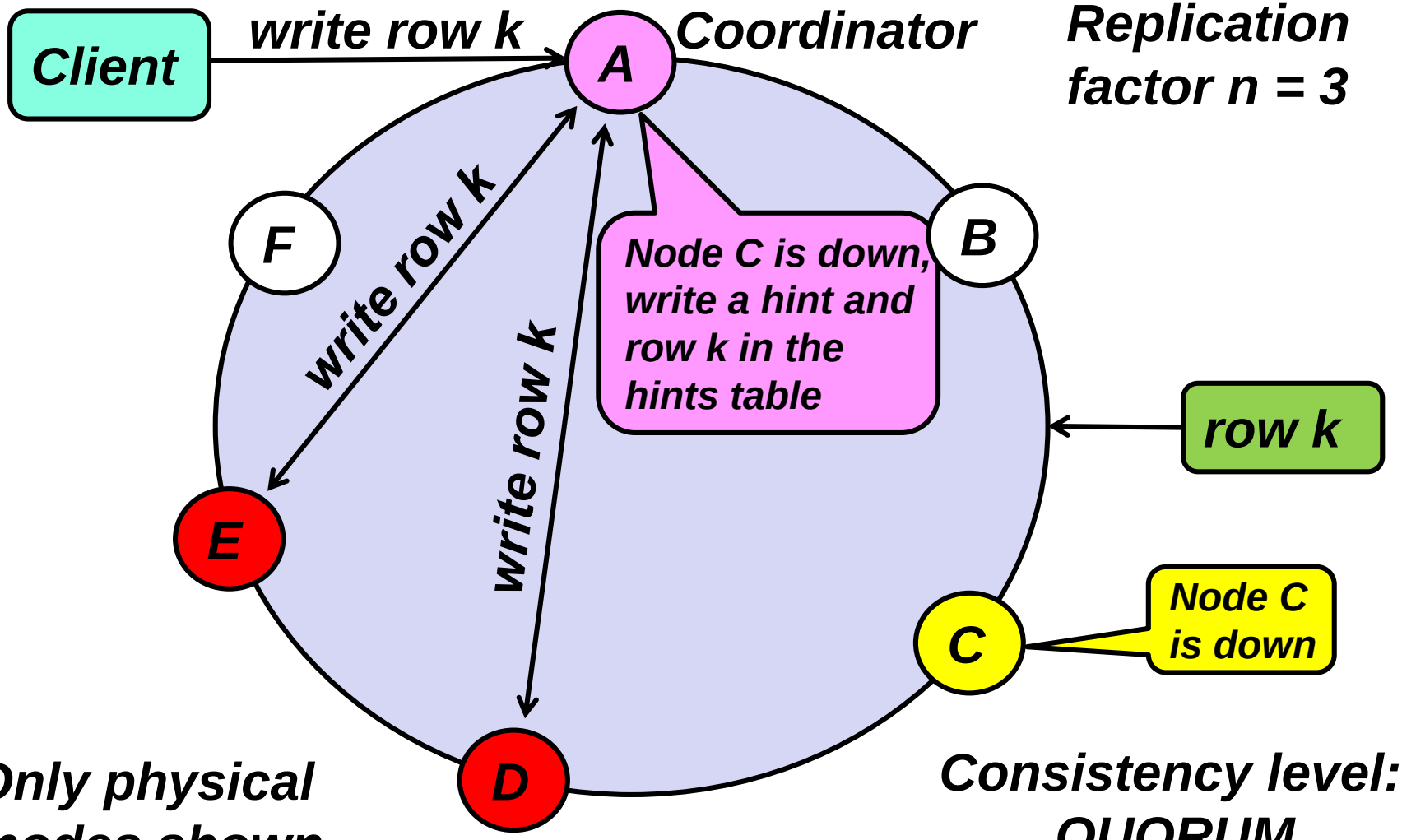
Hinted Handoff Writes

- The hinted handoff is a Cassandra mechanism that writes a hint and data to the coordinator node if some of *n* replicas are down or not replying
- The hinted handoff is applied only in the case when there are enough available replica nodes to satisfy the requested consistency level
- Hinted handoff is not a process that guarantees successful write operations, except when a client application uses a consistency level of `ANY`
- Hinted handoff is enabled by default
- It may be disabled in the `cassandra.yaml` file

How Hinted Handoff Works (1)

- During a write operation, the coordinator node stores a hint about unavailable replica nodes in the local `system.hints` table and the actual data being written
- A hint indicates that a write needs to be replayed to one or more unavailable nodes
- After a node discovers from gossip that a node for which it holds hints has recovered, the coordinator node sends each hinted data to the recovered node

Hinted Handoff



How Hinted Handoff Works (2)

- By default, hints are saved for three hours after a replica fails because if the replica is down longer than that, it is likely permanently dead
- If a node recovers after the save time has elapsed, a node repair should be run to re-replicate the data written during the down time

Hinted Handoff and Consistency Level

- A hinted handoff write is not applied if the consistency level of `ONE`, `QUORUM`, or `ALL` is requested but not met
  - If insufficient replica nodes are available to satisfy the requested consistency level, the `UnavalableException` will be thrown regardless of Hinted Handoff being on
- Example:
  - Assume a six node cluster, hinted handoff set on, and a keyspace with the replication factor of *n = 1*
  - Let the node A be the coordinator that needs to write a row to the node B that is unavailable
  - Cassandra refuses to write the hint to A, because there would be no way to read the data written at any consistency level until B comes back up and A forwards data to B

Extreme Write Availability

- For applications that want Cassandra to accept writes even when all the normal replicas are down, Cassandra provides consistency level `ANY`
  - The coordinator writes data in its hinted handoff table
  - Writes hinted handoff data to each of the replicas, when it comes up
- **`ANY`** guarantees that the write is durable and will be readable after an appropriate replica node becomes available and receives the hinted data

The Background Read Repair Request

- The read consistency level specifies how many replicas must respond to a read request before returning data to the client application
- If the replicas contacted are inconsistent, the coordinator issues writes to the out-of-date replicas in the background
- The coordinator also contacts in the background all the remaining replicas that own the row and issues the writes to any inconsistent ones
- This process is known as read repair
- **Warning:**
  - The read repair depends on the setting of the table property `dclocal_read_repair_chance` that is by default 0.1

Repairing Nodes

- Hinted handoff and read repair work well, but there are scenarios under which hardware failures induce:
  - Loss of hints-not-yet-replayed from requests that the failed node coordinated
  - Read repair fixes data that is actually requested, but data not requested may remain inconsistent for ever (contradicting BASE)
- So, node repair (synchronisation, or anti entropy node repair) is un evitable
- Anti entropy means comparing all the replicas of each piece of data that exist (or are supposed to) and updating each replica to the ***newest version***

Merkle Trees (1)

- Merkle trees are used to discover differences in key sets of the same key range held on different nodes
- A Merkle tree is a full binary hash tree where leaves are hashes of individual keys
- Parent nodes are hashes of their concatenated children
- Let k be the number of keys and h the height of the tree, then:
  - The number of tree leaves $2^{h-1} \geq k$ and $h = \lceil log_2 k \rceil + 1$
  - The k-th key has to be replicated $r = (2^{h-1} - k)$ times in order to get a full tree
- Example:
  - $k = 5$, $h = 4$, the number of replicated keys is $r = 3$

Merkle Trees (2)

- Two Merkle trees have the same respective node values if they are produced using the same set of keys and by applying the same hashing function
- Use of Merkle trees in comparing the key ranges of two replicas is performed in the following way:
  - If tree roots are the same, then key ranges contain the same keys
  - If tree roots are different, then their subtrees have to be compared
    - By applying the rule above recursively, one finally finds a missing key

Merkle Trees (Example) (1)

- This is an extremely simplified example, far from reality
- The only aim of the example is to give you an idea how Merkle trees might be built and used in synchronizing replicas
- Assume:
  - The replica1 contains the following keys: 159, 973, 414, 003
  - The replica2 contains the following keys: 159, 973, 414
  - We use the hash function $h(k) = k \bmod 7$
- In reality, the hash function is applied on a concatenated value of a row key, column name, and timestamp

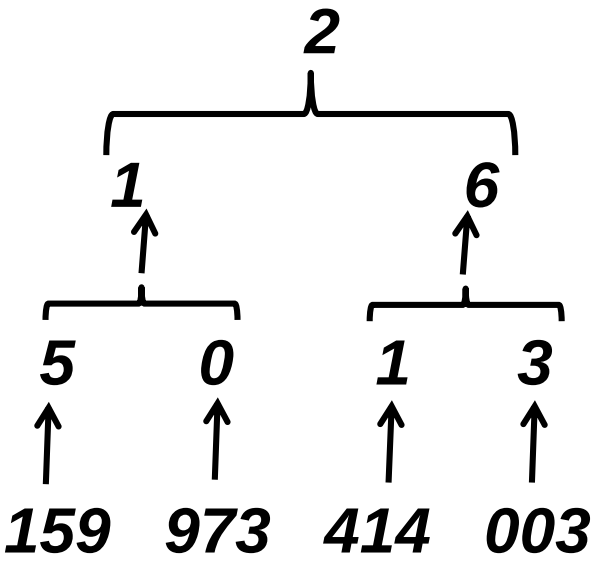
Merkle Trees (Example) (2)

Hash function: *k* (*mod 7*)

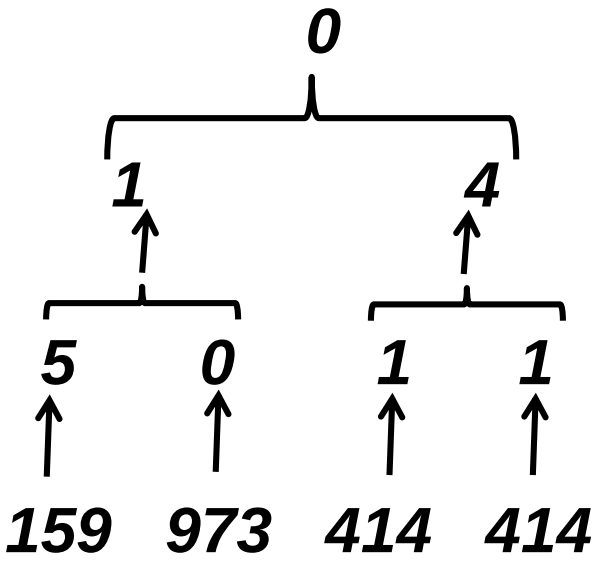
replica1 keys: *159, 973, 414, 003*

replica2 keys: *159, 973, 414*

Replica1



Replica2



Node Repair Types

- Frequent data deletions and nodes going down are common causes of data inconsistency
- The node repair is started using the `nodetool`
- There are several types of node repair:
  - Sequential - One node is repaired after another, and done in full, all SSTables are repaired (default)
  - Incremental - Persists already repaired data, which allows the repair process to stay performant and lightweight as datasets grow providing repairs are run frequently
  - Partitioner range - Repairs only the first range returned by the partitioner for a node
    - This repair type operates on each node in the cluster in succession without duplicating work

Basic Principles of Node Repir

- A repair begins with the repair leader sending out a prepare message to its peers
- Each node involved in the repair has to construct its Merkle tree from all the SSTables it stores, making the calculation resource intensive
- Once the leader receives a Merkle tree from each node, it compares the trees and issues streaming requests
- This allows for repairs to be network efficient as only rows identified by the Merkle tree as inconsistent are sent across the network

Resurrection of Deleted Data

- Marking data with a tombstone signals Cassandra to retry sending a delete request to a replica that was down at the time of delete
- If the replica comes back up within the grace period of time, it eventually receives the delete request
- However, if a node is down longer than the grace period, the node can miss the delete because the tombstone disappears after `gc_grace_seconds`
  - The grace period is by default 10 days
  - When the node gets up again, the deleted data resurrect

Node Repair and Deletes

- Cassandra documentation recommends doing repair once a week
  - Since it must be run before `gc_grace` expires to ensure deleted data is not resurrected
    - The parameter `gc_grace_seconds` is a Cassandra table property with the default value of 10 days
    - Specifies the time to wait before garbage collecting tombstones (deletion markers)
- The repair can be skipped if:
  - The cluster has synchronized time,
  - Deletes are done via TTL only, and
  - Inserts are done with a TTL

Summary

- The hinted handoff is a Cassandra mechanism that writes a hint and data to the coordinator node if some of *n* replicas are down or not replying
  - If insufficient replica nodes are available to satisfy the requested consistency level, the coordinator will not write the hint
- Read repair synchronizes replica inconsistencies
  - If a data item is never read it may remain inconsistent for ever
- Node repair is a manually initiated procedure that synchronizes replicas:
  - To detect the inconsistencies between replicas faster and to minimize the amount of data transfer between nodes, Cassandra uses Merkle trees