Deep Into Data Repair Mechanisms

Cassandra Paris Meetup

@XebiaFR

12/2013

Clément Lardeur

About me

Xebia

Software Engineer (Full Stack Developer)

--

DATASTAX

Cassandra Trainer, Certified Developer

--

@ClemLardeur

clardeur

Summary

- Hinted Handoff
- Read Repair
- Anti-entropy repair node

Hinted Handoff

Goals

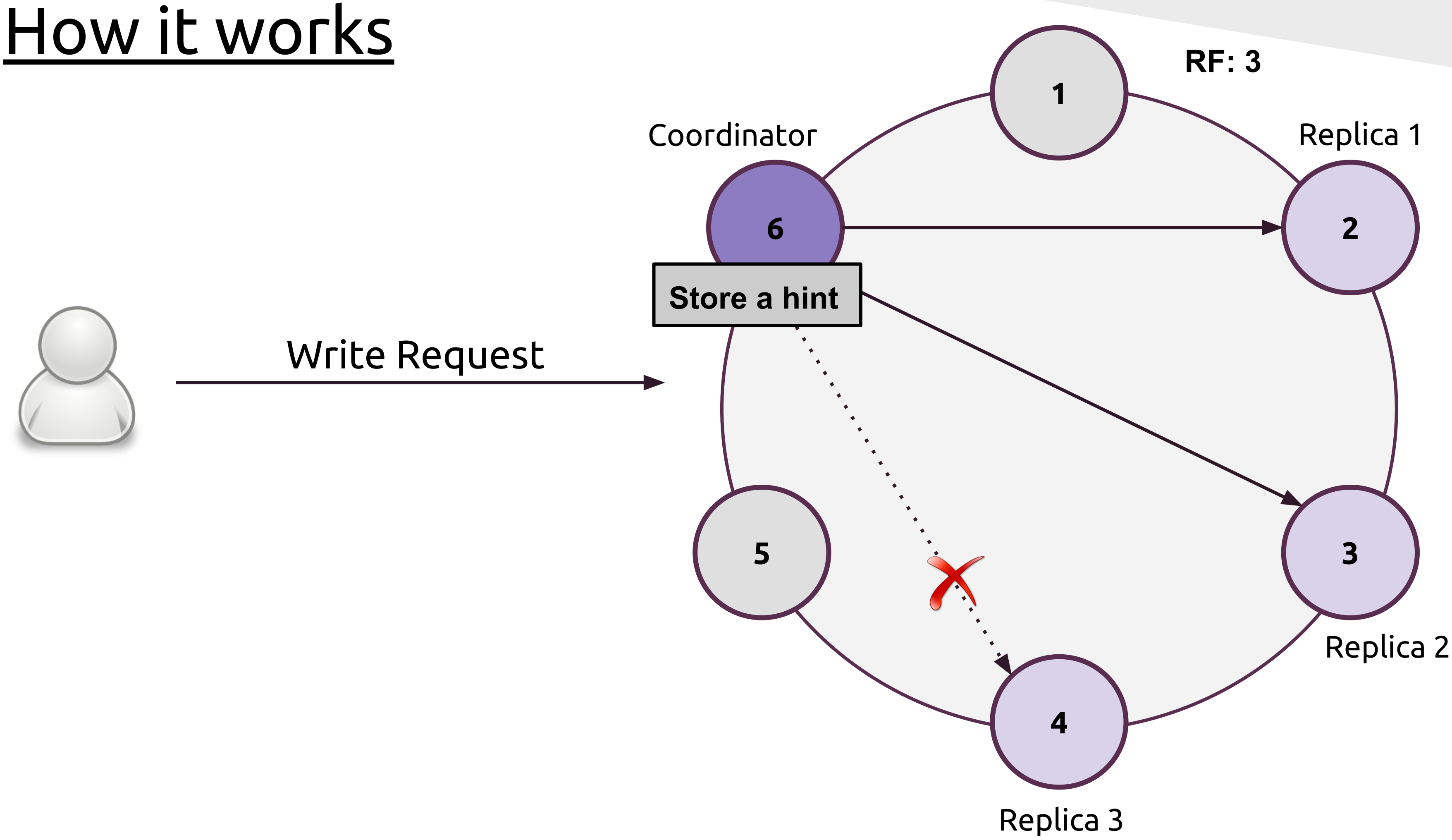
1. Offer full write availability when consistency is not required = **Extreme write availability**

2. Improves response consistency after temporary outages such as network failures = **Repair data**

`hinted_handoff_enabled :` (Default: true)

Hinted Handoff

How it works



Hinted Handoff

The coordinator store a hint, when

A replica node for the row is either known to be down ahead of time.

OR

A replica node for the row does not respond to the write request.

Hinted Handoff

Hints are replayed, when

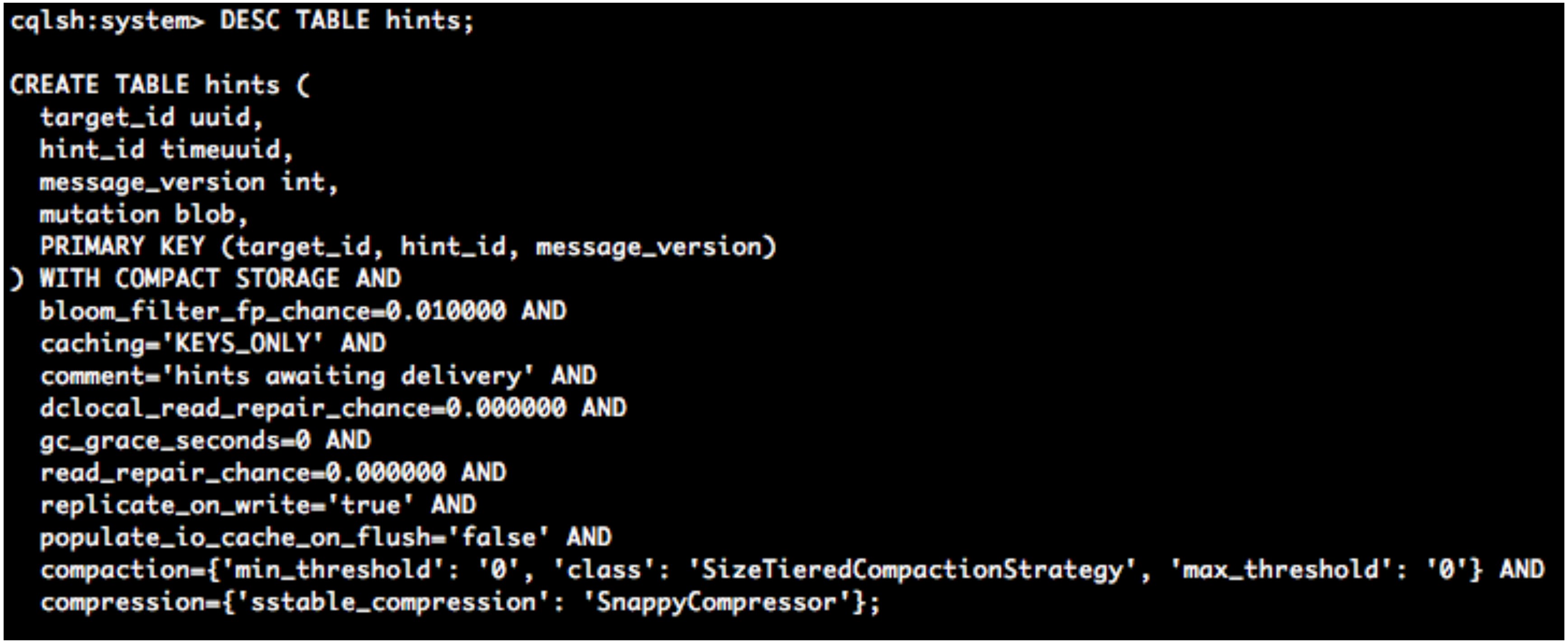
A node is making alive by the Gossiper

OR

The node checks every ten minutes for any hints for writes that timed out during an outage too brief for the failure detector to notice through gossip

Hinted Handoff

system.hints



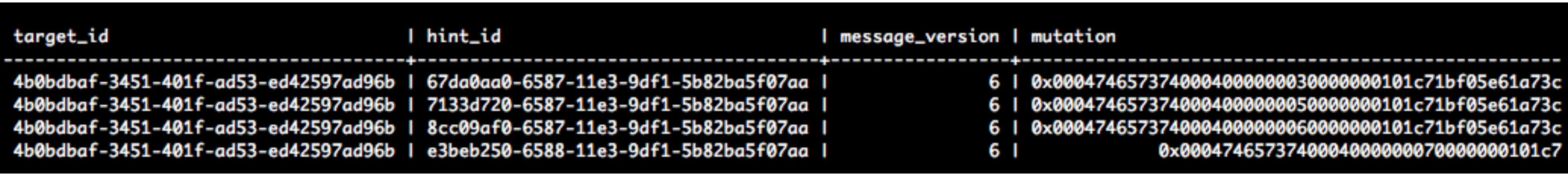
```
cqlsh:system> DESC TABLE hints;

CREATE TABLE hints (
  target_id uuid,
  hint_id timeuuid,
  message_version int,
  mutation blob,
  PRIMARY KEY (target_id, hint_id, message_version)
) WITH COMPACT STORAGE AND
  bloom_filter_fp_chance=0.010000 AND
  caching='KEYS_ONLY' AND
  comment='hints awaiting delivery' AND
  dclocal_read_repair_chance=0.000000 AND
  gc_grace_seconds=0 AND
  read_repair_chance=0.000000 AND
  replicate_on_write='true' AND
  populate_io_cache_on_flush='false' AND
  compaction={'min_threshold': '0', 'class': 'SizeTieredCompactionStrategy', 'max_threshold': '0'} AND
  compression={'sstable_compression': 'SnappyCompressor'};
```

Hinted Handoff

`system.hints`

- `target_id` : node ID concerned by the hint
- `hint_id` : hint ID (with a timestamp)
- `message_version` : internal message service version
- `mutation` : the actual data being written

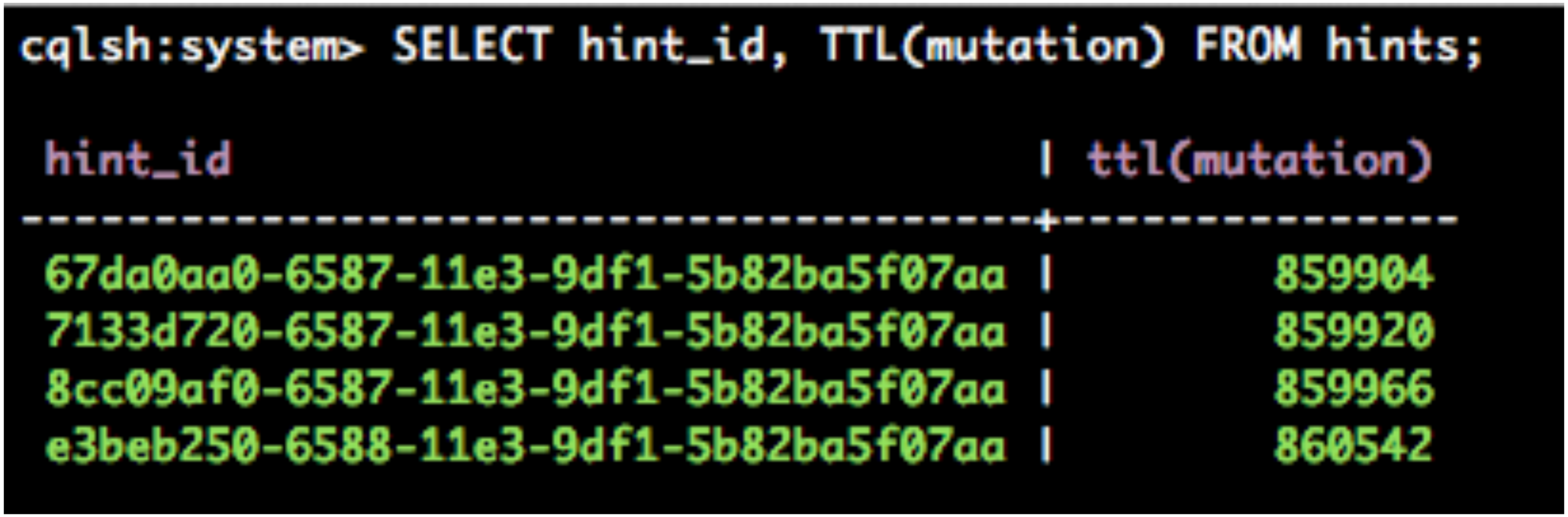


```
 target_id                            | hint_id                              | message_version | mutation
--------------------------------------+--------------------------------------+-----------------+---------------------------------------------------
 4b0bdbaf-3451-401f-ad53-ed42597ad96b | 67da0aa0-6587-11e3-9df1-5b82ba5f07aa |               6 | 0x0004746573740004000000030000000101c71bf05e61a73c
 4b0bdbaf-3451-401f-ad53-ed42597ad96b | 7133d720-6587-11e3-9df1-5b82ba5f07aa |               6 | 0x0004746573740004000000050000000101c71bf05e61a73c
 4b0bdbaf-3451-401f-ad53-ed42597ad96b | 8cc09af0-6587-11e3-9df1-5b82ba5f07aa |               6 | 0x0004746573740004000000060000000101c71bf05e61a73c
 4b0bdbaf-3451-401f-ad53-ed42597ad96b | e3beb250-6588-11e3-9df1-5b82ba5f07aa |               6 |             0x0004746573740004000000070000000101c7
```

Hinted Handoff

Hint TTL

- `max_hint_window_in_ms` : default 10800000 = 3H



```
cqlsh:system> SELECT hint_id, TTL(mutation) FROM hints;

 hint_id                              | ttl(mutation)
--------------------------------------+---------------
 67da0aa0-6587-11e3-9df1-5b82ba5f07aa |        859904
 7133d720-6587-11e3-9df1-5b82ba5f07aa |        859920
 8cc09af0-6587-11e3-9df1-5b82ba5f07aa |        859966
 e3beb250-6588-11e3-9df1-5b82ba5f07aa |        860542
```

Hinted Handoff

Hint optimizations

- `max_hint_delivery_threads` : default 2

  ⇨ Need to be increased for multiple DC

- `hinted_handoff_throttle_in_kb` : default 1024

  (Maximum throttle in KBs per second, per delivery thread)

Hinted Handoff

Key points

- Hinted handoff is enabled by default
- Hinted handoff is an optional part of write
- Hinted handoff is an optimization
- Hints have a TTL
- Hints are uncorrelated to consistency level

Read Repair

Goals

1. Ensure that all replicas have the most recent version of frequently-read data.

2. Anti-entropy real-time mechanism.

`read_repair_chance :` (Default: 0.1)

Read Repair

Global read setup

- Determine replicas to invoke
  - ConsistencyLevel vs Read Repair
- First data node respond with full data set, others send digest
- Coordinator waits for ConsistencyLevel

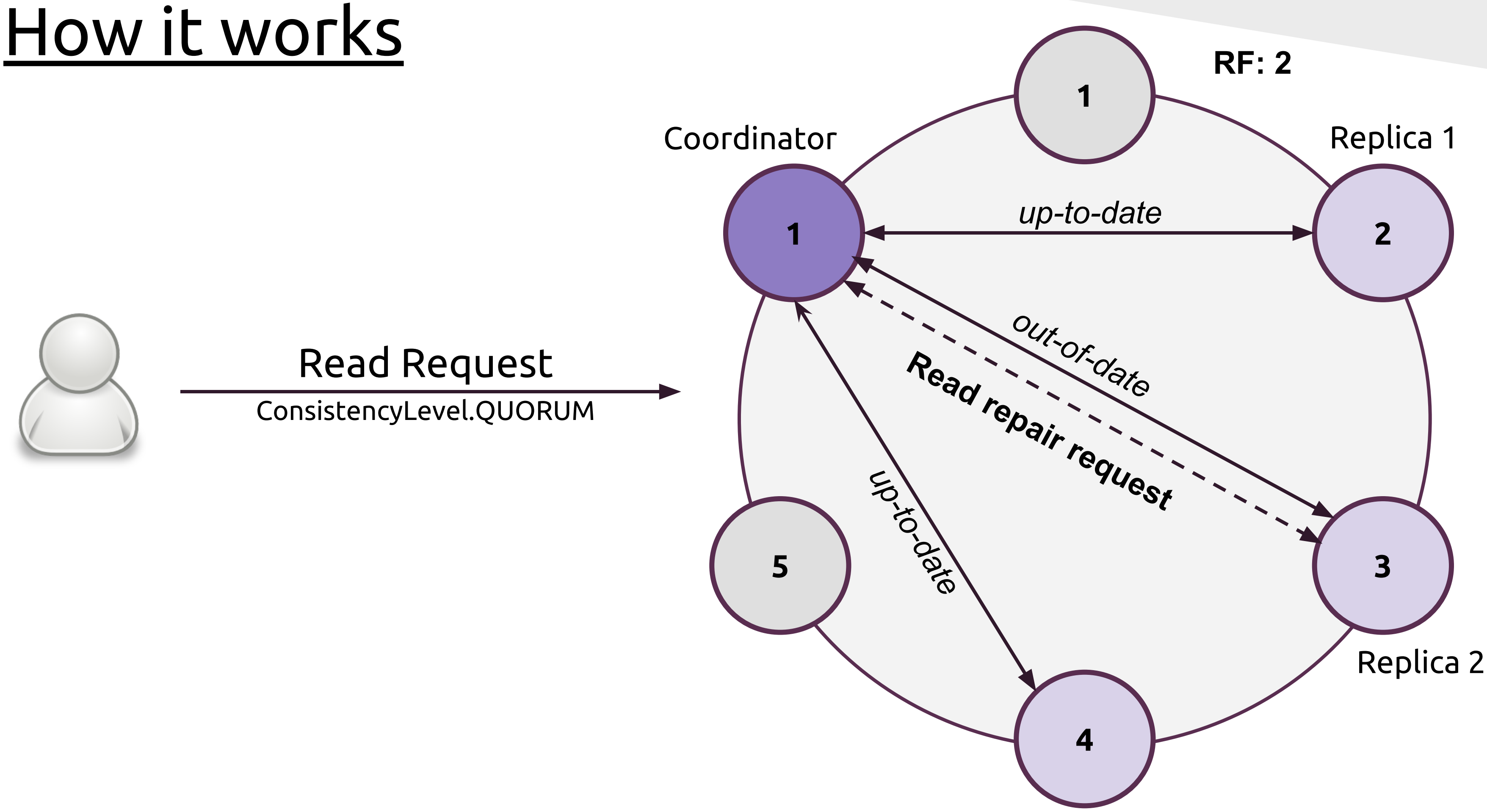
Read Repair

Consistent reads - algorithm

- Compare digests
- If any mismatch
  - re-request to same nodes (full data set)
  - compare full data sets, send update
  - block until out-of-date replicas respond
- Return merged data set to the client

Read Repair

How it works



Read Request

ConsistencyLevel.QUORUM

Read Repair

The coordinator send a read repair, when

A replica node for the row has responded an out-of-date value.

OR

The read repair chance declared on the column family is activated.

Read Repair

Read repair configuration

- `read_repair_chance` : Default to 0.1
- `dclocal_read_repair_chance` : Default to 0.0

➡ Configured by Column Family (Table)

Read Repair

Key points

- Consistent read is a part of read request
- Read repair is a probability to sync data
- Read repair is configured by column family
- Read repair can be Local DC or Global

Anti-entropy repair node

Goals

1. Ensures that all data on a replica is made consistent.

2. Repair inconsistencies on a node that has been down for a while.

```
nodetool repair <keyspace> [table] <opts>
```

Anti-entropy repair node

Nodetool repair, when

During normal operation as part of regular, scheduled cluster maintenance.

OR

During node recovery after a failure or on a node that has been down for a while.

OR

On nodes that contains data that is not read frequently.

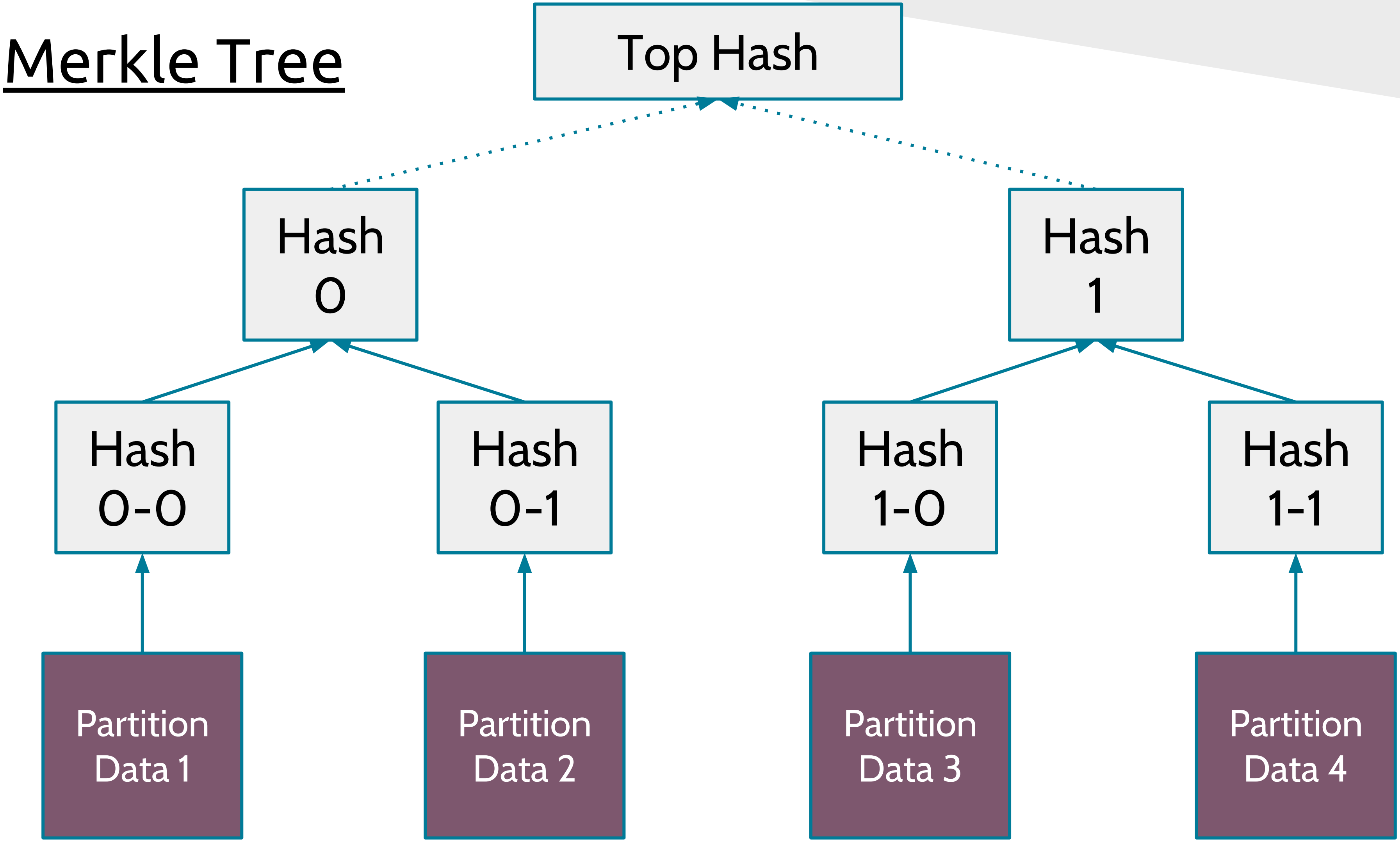
Anti-entropy repair node

How it works

- Determine peers nodes with matching ranges.
- Triggers a major (validation) compaction on peers.
  - i.e. do the read part of the compaction stage
- Read and generate the Merkle Tree on each peer.
- Initiator awaits trees from peers.
- Compare every trees to every other trees.
- If any differences, the differing nodes exchange conflicting ranges.

Anti-entropy repair node

Merkle Tree



Top Hash

Hash 0

Hash 1

Hash 0-0

Hash 0-1

Hash 1-0

Hash 1-1

Partition Data 1

Partition Data 2

Partition Data 3

Partition Data 4

Anti-entropy repair node

Cautious

- Building the Merkle Tree is disk I/O and CPU intensive
  - due to validation compaction

- Overstreaming occurs
  - due to the streaming of partitions

Anti-entropy repair node

Options

- `-pr` (--partitioner-range) : repairs only the main partition range for that node

➡ Use `-pr` for periodic repair maintenance, and execute repair on each node.

➡ Don’t use `-pr` for a recovering node because other replicas for that node need to be repaired too.

Anti-entropy repair node

Options

- `-snapshot` : only one replica at a time do computation => make sequential repairs.

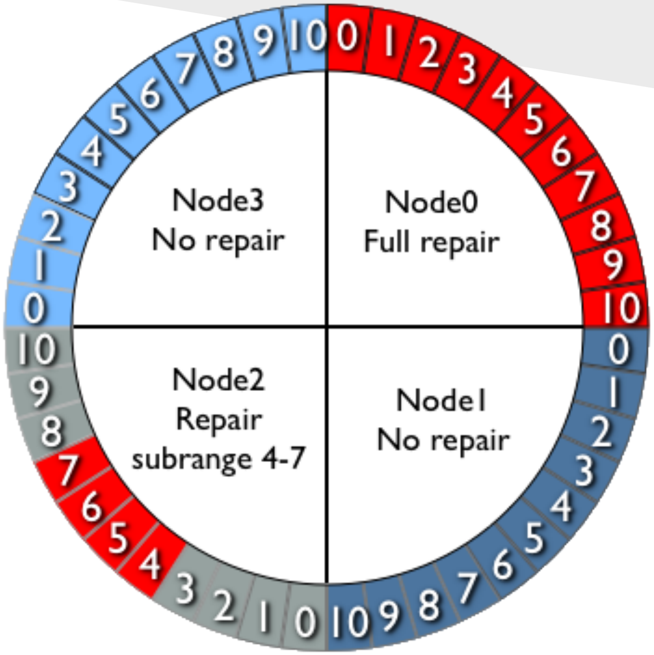
➡ Always use `-snapshot` to avoid overloading the targeted node and it’s replica.

- Since 2.0.2, sequential repair is the default behavior.
  - renamed to `-par` (for parallel)

Anti-entropy repair node

Options

- `-st` : start token
- `-et` : end token



➡ $ nodetool repair `-st` *‹start_token›* `-et` *‹end_token›*

➡ Completely eliminates the overstreaming (subrange repair)

Anti-entropy repair node

Key points

- Use nodetool repair as a maintenance task
- Use nodetool repair cautiously
- Use relevant options when needed

Thank you !

Q & A time

@ClemLardeur