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

Cassandra Transaction Processing

Lecturer : *Dr. Pavle Mogin*

SWEN 432
Advanced Database Design and Implementation

Cassandra The Fortune Teller

Plan for Transaction Processing

- Transactions and Concurrency Control
  - About the satisfaction of ACID properties
- Light Weight Transactions
  - Compare and Set (CAS) lightweight transactions in CQL3

Prolog

- Cassandra does not use RDBMS ACID transactions with rollback or locking mechanisms, but instead offers:
  - Atomic,
  - Isolated, and
  - Durable transactions with
  - Tunable consistency
- Cassandra supports atomicity and isolation at the row-level, but trades transactional isolation and atomicity in multi row transactions for high availability and fast write performance

Atomicity

- In Cassandra, a write is atomic at the partition-level, meaning inserting or updating columns in a row is treated as one write operation
- Cassandra does not support transactions in the sense of bundling multiple row updates into an all-or-nothing operation
- Nor does it roll back when a write succeeds on one replica, but fails on other replicas
- It is possible in Cassandra to have a write operation report a failure to the client, but still actually persist the write to a replica

Isolation and Durability

- Isolation
  - Cassandra supports full row-level isolation, which means that writes to a row are isolated to the client performing the write and are not visible to any other user until they are complete

- Durability
  - Writes in Cassandra are durable
  - All writes to a replica node are recorded both in memory and in a commit log on disk before they are acknowledged as a success
  - If a crash or server failure occurs before the memtables are flushed to disk, the commit log is replayed on restart to recover any lost writes
  - In addition to the local durability (data immediately written to disk), the replication of data on other nodes strengthens durability

Lineariazable Consistency

- Cassandra 2.0 offers two types of consistency:
  - Tunable consistency, and
  - Linearizable consistency
- Linearizable consistency is a serial isolation level for lightweight (LW), or compare-and-set (CAS) transactions when concurrently updating multiple rows or tables
- Linearizable consistency is used in rare cases when a strong version of tunable consistency is not enough
- Cassandra uses an extension of the Paxos consensus protocol to support linearizable consistency with quorum-based operations

When to Use Lineariazable Consistency

- A typical example that justifies use is when an application that registers new accounts needs to ensure that only one user can claim a given account
  - The challenge is handling a race condition when two users are attempting to make an insertion with the same row key
  - Checking for the existence of the account before the insert by the user A does not guarantee that the user B will not insert the account between the check time and A's insert
  - Linearizable consistency meets this challenge
- Use of LW or CAS transactions incurs an effective degradation to **one-third** of the normal performance

Lightweight Transactions in CQL3

- LW transactions can be used for both INSERT and UPDATE statements, using the `IF (NOT) EXISTS` clause

```
INSERT INTO users (email, name, password)
values ('pmogin@ecs.vuw.ac.nz', 'Pavle',
'1234') IF NOT EXISTS;
```

  - If the user `'Pavle'` already existed, he **would not** be overwritten (to protect other people's work)

```
UPDATE users SET password = '9999'
WHERE name = 'Pavle' IF password = '1234';
```

  - If the `password` **was not** `'1234'`, it **would not** be updated (to protect other people's work)
  - The columns updated do NOT have to be the same as the columns in the `IF` clause