Transactions in Couchbase

ANJU MUNOTH

Transactions

- Guarantee that multiple documents can be updated atomically.
- *Distributed ACID Transactions* are operations that ensure that when multiple documents are modified either all the modifications are completed successfully, or all the documents are rolled back to their previous state
- *Atomicity* supports *insert*, *update*, and *delete* operations, across any number of documents.

- Transactions make use of the Durability provided by *synchronous writes*.
- During execution, for each document, a transaction maintains a modified copy of the document in the Extended Attributes area of the document's meta data.
- Consequently, the changes that occur during the transaction prior to commitment — these constituting *uncommitted* (or *dirty*) data — are non-readable by any operation other than this transaction itself.
- Only following commit of the documents modified by the transaction — when the modified data-copy is removed from the meta data area, and is written over the main body of the document's data — do the corresponding changes become readable by other transactions and operations.
- This isolation-level is known as Read Committed.
- Use of a document's extended attributes area results in an approximate doubling of its pre-transaction size, while the transaction is underway.
- In consequence, transactions can only be used on documents whose maximum size is 10 MB (which is half of the maximum-permitted document-size, 20 MB).

- As well as making use of individual documents' extended attributes, transactions also create additional documents in each bucket that they access.

These include:

- Active Transaction Records. Multiple Active Transaction Records can exist per bucket.
  - Each of these documents can contain entries for multiple transactions.
  - Collectively, these documents record all currently active transactions for the bucket.
  - The name of each document is prefixed with _txn:atr-.
- Transaction Client Records. One of these exists per bucket, covering all clients currently accessing the bucket transactionally.
  - Document is named _txn:client-record.
- These documents, which may persist indefinitely, are automatically maintained by Couchbase Server, and should not be modified by any application.

- A transaction can be performed across multiple buckets.
- Only nodes that contain data to be updated are affected by a transaction.
- Multiple transactions can read the same document at the same time.
- If two transactions simultaneously attempt to write to the same document, one is allowed to proceed, while the other is obliged to retry.

Services and Transactions

- The indexes provided by the Index, Search, and Analytics Services are *not* atomically updated with the commits performed by transactions
  - Updated with Eventual Consistency.
- Neither the Query Service nor the Search Service sees uncommitted data.
  - Snapshot Isolation is *not* provided: consequently, if a transaction performs a commit while a query or search is ongoing, then the query or search may return data from both prior to and subsequent to the commit.
- Note that the Query Service provides the At_Plus feature, which allows queries to wait for indexes to be appropriately updated, following a transaction.

Limitations

- Only documents whose initial size is 10 MB or less can be included in a transaction.
- Non-transactional updates should not be made to any document involved in a transaction while the transaction is itself in progress: this prevents the non-transactional update from being overwritten.
- The number of writes required by a transactional update is greater than the number required for a non-transactional update
  - Transactional update requires writes in order to stage and commit data, and also to maintain transaction records.
  - Consequently, transactional updates may be less performant than non-transactional updates.
- Data within a single document is *always* updated atomically: therefore, whenever multiple key-value pairs consistently require atomic update, their co-location within a single document may best ensure high performance.

Limitations

- Cross Data Center Replication (XDCR) supports *eventual consistency*: however, it does *not* support atomicity — nor does XDCR Conflict Resolution.
    - Consequently, transactionally modified documents should only be replicated across clusters if no transactions involving the same documents can occur on those clusters simultaneously.
- Since transactions make use of the Durability provided by *synchronous writes*, in order to use transactions in development on a *single-node cluster*, the number of replicas assigned to any bucket to be used transactionally must be established as *zero*.
    - If the number is greater than zero, the required durability level is unattainable, and the write fails.
- If, on a single-node cluster, a bucket's replica-assignment was previously greater than zero, and the number is reduced to zero in order to support synchronous writes, a rebalance must be performed prior to any synchronous write.

Creating and Using Transactions