Supporting Savepoints in YSQL

Rob Sami & Karthik Ranganathan

Friday, April 29, 2022

YFTT
YUGABYTEDB
FRIDAY
TECH TALKS

yugabyteDB

What are Savepoints?

From Postgres Docs…

“A savepoint is a special mark inside a transaction that allows all commands that are executed after it was established to be rolled back, restoring the transaction state to what it was at the time of the savepoint"

Source: https://www.postgresql.org/docs/current/sql-savepoint.html

Why Savepoints?

Graceful Failure of Operations

- Some operations in your transaction fail with low probability, but can be retried or avoided without losing the rest of your transactions progress.

Why Savepoints?

PL/pgSQL

- Some operations in your transaction fail with low probability, but can be retried or avoided without losing the rest of your transactions progress.
- PL/pgSQL stored procedures support exception handling using savepoints. With savepoints support in YSQL, we can support exceptions in stored procedures, etc.

Why Savepoints?

Ecosystem Integration

- Some operations in your transaction fail with low probability, but can be retried or avoided without losing the rest of your transactions progress.
- PL/pgSQL stored procedures support exception handling using savepoints. With savepoints support in YSQL, we can support exceptions in stored procedures, etc.
- Many frameworks (especially ORMs) which sit on top of postgres depend on savepoints: JDBC, SQLAlchemy, and others.

Why Savepoints?

...and Postgres tests!

- Some operations in your transaction fail with low probability, but can be retried or avoided without losing the rest of your transactions progress.
- PL/pgSQL stored procedures support exception handling using savepoints. With savepoints support in YSQL, we can support exceptions in stored procedures, etc.
- Many frameworks (especially ORMs) which sit on top of postgres depend on savepoints: JDBC, SQLAlchemy, and others.
- We can expand the set of Postgres integration tests which we can run against YSQL as we continue to narrow the gap in feature parity.

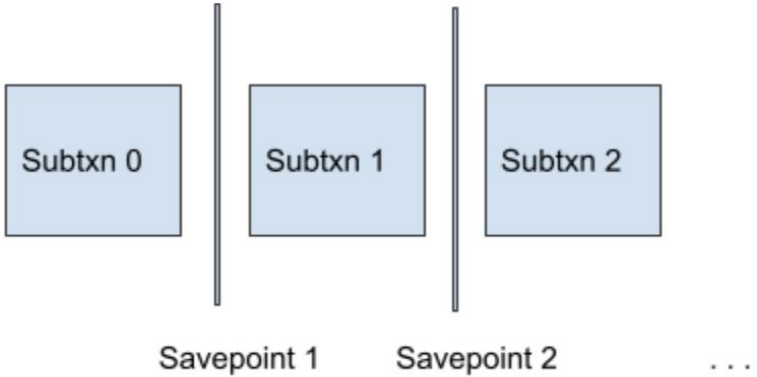
How to Use Savepoints

Demo

How We Built It

Data model

Subtransactions



Provisional Records

intent_key -> {txn_id}{subtxn_id}{value}

Protocol changes

- Tracks currently active subtransaction_id.
- Tracks a set of which subtransactions are live vs. rolled back.
- Sends both of these pieces of state on every read/write. Mask provisional records from rolled back subtransactions.
- Send live subtransaction set on commit -> only apply live provisional records.

A caveat

Conflict Resolution Updates Asynchronously

- Explicit locks acquired during subtransaction are not released when savepoint is rolled back
- Provisional writes to a key may continue to cause conflict with other transactions even if part of a rolled back subtransaction
- TL;DR – you may see spurious conflicts with operations which were part of a rolled-back subtransaction

- Coming soon – pessimistic locking, which should help alleviate this problem

Thank You

Join us on Slack: yugabyte.com/slack (#yftt channel)

Star us on Github: github.com/yugabyte/yugabyte-db

YFTT YUGABYTEDB FRIDAY TECH TALKS

yugabyteDB