TLA+ Modeling of MongoDB Transactions

Murat Demirbas

Abstract

This talk will present a TLA+ model of MongoDB's distributed transactions, focusing on modeling and test case generation for the storage system. I will discuss the benefits of using TLA+ to validate correctness and isolation properties, detect subtle bugs, and improve confidence in implementation. To the best of our knowledge, this modeling effort is the first of its kind in terms of formal modeling and checking distributed transactions at a level close to the algorithm while also involving peripheral components at different system layers.

Motivation

MongoDB implements distributed transactions across shards using a two-phase commit protocol. These transactions interact with the underlying WiredTiger storage engine, which handles local concurrency control and persistence. Ensuring the correctness of this layered system is challenging due to subtle interactions between hybrid logical clocks (HLC) timestamps, transaction ordering, and storage engine guarantees.

Distributed systems introduce inherent challenges, such as network delays, node failures, and concurrency anomalies, making formal verification essential for ensuring correctness. Formal modeling with TLA+ provides a systematic way to verify the correctness of MongoDB's distributed transactions. Our TLA+ specifications capture the behavior of transactions, and enable us to model check for isolation guarantees under all potential execution paths. This ensures that the system behaves correctly under different conditions, including failures and concurrent executions.

MongoDB's Distributed Interactive Transactions

MongoDB supports interactive multi-document transactions that allow clients to execute multiple operations across different shards while maintaining ACID guarantees. When a transaction begins, MongoDB ensures that all reads and writes within the transaction see a consistent snapshot of the data. The system employs a two-phase commit protocol to ensure atomicity across multiple shards, with a transaction coordinator responsible for orchestrating the commit process. Transactions can span multiple statements and collections, making them well-suited for applications requiring complex multi-step updates. The use of cluster-wide HLC timestamps enables causally consistent reads, which ensures that operations respect dependencies across distributed shards. Transactions can be committed or aborted depending on whether all participant shards acknowledge the commit decision.

A significant challenge in MongoDB's transaction model is handling interactions with the underlying WiredTiger storage engine. The MVCC storage engine enforces snapshot isolation and manages concurrent access to data, requiring careful coordination between the transaction layer and the storage layer. Prepared transactions must be properly handled to ensure consistency, particularly in cases where transactions involve multiple storage operations that must be completed before commit.

Talk Content

The talk will begin with an overview of the TLA+ model of MongoDB transactions, including the specification of distributed transactions, the interaction between the transaction protocol and the storage layer, and the handling of prepared and pending transactions in WiredTiger. This section will highlight the importance of abstracting complex system behaviors into modular specifications that can be individually verified and composed. This section will also show how we model check the transaction protocol for snapshot isolation and read-committed isolation properties.

Next, it will cover the modeling of the storage system by abstracting sharded storage using a single-log model, explicitly defining the storage engine API and constraints, and addressing edge cases where transactions must wait for prepared operations.

A detailed discussion on test case generation will follow. The talk will explain how we utilize the TLC explicit state model checker to generate a set of test cases that exhaustively cover reachable states of our model,

and verify conformance to the behavior of the WiredTiger storage engine implementation.

Benefits of TLA+ Modeling

TLA+ modeling helps detect protocol bugs by revealing edge cases not easily covered by unit tests. It improves understanding of system behavior by providing a precise formal specification for developers. Automated test generation from the TLA+ model enhances verification coverage. Modular reasoning separates transaction logic from storage behavior, clarifying responsibilities and assumptions. By systematically verifying all possible execution paths, TLA+ ensures that the transaction protocol meets its intended guarantees in every scenario. A major benefit of formal verification is its ability to uncover subtle concurrency issues that may not be apparent through empirical testing alone.

The formal specification also serves as documentation for MongoDB's transaction semantics, providing a precise reference for developers working on implementation and optimizations. By maintaining a formal model alongside the implementation, engineers can more easily evaluate modifications and verify their correctness before deployment.

Acknowledgment

This is joint work with Will Schultz at MongoDB Research.