Optimistic Replicated Two-Phase Commit
Irene Zhang     Naveen Kr. Sharma     Adriana Szekeres
Arvind Krishnamurthy     Dan R. K. Ports
University of Washington

Distributed storage systems are a critical component for building scalable, reliable web applications. To meet the high performance and availability demands of web applications, these systems are increasingly built around in-memory replication [4], instead of traditional disk-based storage. At the same time, programmers increasingly prefer systems that support consistent replication and general-purpose, distributed transactions. A number of storage systems, most notably Google’s Spanner [2], have been developed recently that reflect this trend.

The standard design for these storage systems integrates several protocols that work together to provide scalable, replicated transactions. The protocols include concurrency control mechanisms like Strict Two-Phase Locking (S2PL), consensus-based replication algorithms like Paxos [3], and atomic commitment protocols like Two-Phase Commit (2PC). Modern storage systems implement these protocols in separate layers, with each layer providing a subset of the guarantees required for distributed ACID transactions.

In our work, we take a different approach. Instead of separating protocols, we argue that co-designing them provides opportunities for cross-layer optimizations that both improve performance and simplify the protocols. While Spanner and other systems now commonly implement these protocols together, they were originally developed to address separate issues in distributed systems. We analyze the interaction of atomic commitment, consistent replication and concurrency control based on their requirements on layers below and their guarantees for the layers above.

Our analysis leads to several key insights about the protocols implemented in distributed transactional storage systems.

- The concurrency control mechanism is the application for the replication layer, so its requirements impact the replication algorithm.
- Unlike disk-based durable storage, replication can separate ordering/consistency and fault-tolerance guarantees with different costs for each.
- Consensus-based replication algorithms like Paxos are not the most efficient way to provide ordering.
- Enforcing consistency at every layer is not necessary for maintaining global transactional consistency.

Based on these observations, we designed a new protocol, called Optimistic Replicated Two-Phase Commit or OR-2PC, to provide high-performance transactions in a replicated storage system. Our goal in designing OR-2PC is to take advantage of cross-layer optimizations to reduce the complexity and improve the performance of distributed transaction coordination. The protocol integrates:

- A new technique for providing ordering in a distributed environment, based on unique, client-proposed timestamps.
- A new optimistic concurrency control mechanism, similar to CLOCC [1], that enforces both ordering across replicas as well as ordering across transactions.
- A simple quorum protocol, providing replication, but not ordering.

We implement OR-2PC in a new transactional key-value store called Tapir. Tapir supports general-purpose transactions over a distributed set of keys. Applications execute multiple read and write operations within the scope of a transaction, then atomically commit those operations at a single point in time. Tapir also supports read-only transactions that execute at a single snapshot, without requiring the read set be known ahead of time.

In comparison to today’s distributed transactional storage systems, Tapir offers the following benefits:

- **Lower Latency**: Commit transactions in a single round-trip in the common case.
- **Higher Throughput**: No single-server serialization point.
- **Minimal Fault-tolerance Requirements**: Consensus with a simple majority out of 2f+1 replicas.

In our experiments, we found that Tapir out-performs conventional distributed transactional storage systems, including our implementation of the recent Spanner protocol, by more than 2x in latency and 3x in throughput.

References


