Challenges and Choices in building a Replicated, High-Throughput, Distributed ACID Transactions Management System

Having implemented and benchmarked a non-distributed and replicated database system with replicas executing distinct ACID transactions in parallel for performance, we are now implementing a distributed but unreplicated system with the end goal of incorporating replication and parallel processing. This talk will present some of the design and implementation challenges that we faced and addressed in our long journey of successfully combining functional requirements of ACID with non-functional aspects of availability and throughput.

An overarching design challenge is to ensure that design solutions chosen for distinct problems work well together without increasing overhead or even better take advantage of each other. For example, replication requires consensus which in turn (freely) offers order on transactions and hence the management of transactions access conflict (e.g., write-read) can be based on this order and not on timestamps derived from any global time base (e.g. as in CockroachDB) that can be done away with; further, 2-phase commit, essential for atomic commit, is a centralised consensus and hence a throughput killer when used for managing replication; we need a decentralised consensus at replication level that builds a 1-server abstraction and works seamlessly well with 2PC at transactions commit level.

First, we selected the best set of precedence-based conflict management solutions for concurrency control which offered low-overhead and low transaction abort rates. This selection was done by modelling a large system and running simulations, and those choices were then verified using implementations. (PhD thesis, 2023)

Next, we built a replicated centralised system which used a decentralised order protocol for merging distinct incoming transaction streams that were processed simultaneously by replicas, into one global, conflict-free transaction stream ready to be committed. Our order protocol is a ring-based protocol as the ring structure is proven to offer the highest possible throughput by Guerraoui et. al. (ACM TOCS, 2010). (PhD thesis, 2025).

RAFT is a well-known, centralised order protocol suite with recovery mechanisms in case a crash occurs. We also established that with some minor tweaks RAFT recovery mechanisms can be used within a decentralised, Ring-based protocol to recover from crashes. (PhD, 2024)

Our nearly completed implementation of a distributed transaction system unearthed some surprising challenges. When a transaction needs to access objects on a remote shard, the local thread executing that transaction, can block on timeout (as in Arjuna) or work on another job; in the latter case, the new job can be one of three types: a partially executed transaction, a fully executed transaction in/awaiting validation, or a new transaction. The job selection must be according to a specific priority; otherwise, aborts will increase when conflict resolution is precedence-based and wait-free; in the extreme, only conflict-free transactions will commit. Another challenge is in the garbage collection of meta-data about completed (aborted/committed) transactions. The latter do not (in fact, cannot) track their successors in the precedence relation imposed and their meta data must be maintained until all their successors complete. We leverage the *Atomic Reference Count* (ARC) facility supported by RUST language to efficiently accomplish this.