Optimistic Concurrency Control for Distributed Transactions

May 31, 2021

Contents

Collection of notes and code for the project of the Distributed Systems 1 course, part of the Master in Computer Science at the Università di Trento (UNITN).

1 Concepts

1.1 TODO Strict serialization

1.2 TODO Optimistic concurrency control

1.3 TODO Two-phase commit (2PC)

2 Protocol

The goal is to allow **multiple clients** to make **concurrent transactions** (**TXN**) on a **distributed database**. A TXN is a sequence of R/W operations on one or more items that can be commited or aborted.

2.1 Assumptions

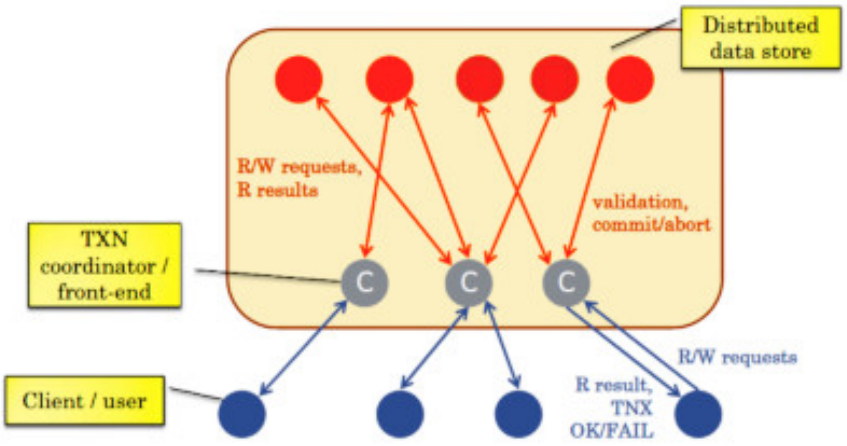
- Links are FIFO and reliable.
- Unicast transmission delays are simulated with small, random intervals.
- Clients don't crash, but coordinators and servers can crash at any time (and recover after a given time).

2.2 Requirements

- Crashes should be simulated on critical points of the execution, like validation/commit.
- After a number of transactions takes place, the system allows to check if its state is consistent.

2.3 Implementation

The distributed nature requires us to use atomic commitment to guarantee consistency (e.g. 2PC). We assume conflicts are rare, and in doing so we choose to use a private workspace instead of any form of lock (whose adoption in our scenario would be wasteful and non-trivial).



2.3.1 Client

Each client tries to start a transaction, one at a time, by sending a `TXN_BEGIN` to a randomly choosed coordinator. Once the coordinator confirms the transaction, the client will send a series of operations (ending with a `TXN_END`) and the TXN will eventually commit or abort.

The client implementation is given, and its operation consist in reading items two-by-two, then for a given value `x` subtracting it to the first item and adding it to the second (the sum should stay constant). Clients don't crash.

2.3.2 Coordinator

Coordinators are responsable for

- Uniquely identifying the TXNs
- Forwarding and replying to requests from the client
- Changing the values as requested by the client
- Orchestrating concurrent TXNs using distributed commits to ensure strict serializability

TXNs directly run on the private workspaces, and if valid they are later commited by the coordinator. The validation of a workspace is the only phase that would requirer the coordinator to lock (some) items, still for a very short window of time if compared with pessimistic locking.

2.3.3 Server

The server stores a part of the distributed data, that consists in a simple collection of key-value pairs (with integer values). A data store server S_i manages data with keys in the range $[10i \cdots 10i + 9]$.