Distributed Databases on Top of SPNs: A case study with Cockroach DB

Yilong Geng and Balaji Prabhakar
Stanford University

Applications of Huygens -- a software clock synchronization system for data centers

Resource scheduling

- TDMA style scheduling in Ethernet
- Example: Fastpass (Sigcomm 2014)

Financial applications

Distributed databases

- Improve consistency and performance of databases (Spanner, CockroachDB)
- High throughput distributed ledgers (blockchains)

Applications of Huygens -- a software clock synchronization system for data centers

Resource scheduling

- TDMA style scheduling in Ethernet
- Example: Fastpass (Sigcomm 2014)

Financial applications

Distributed databases

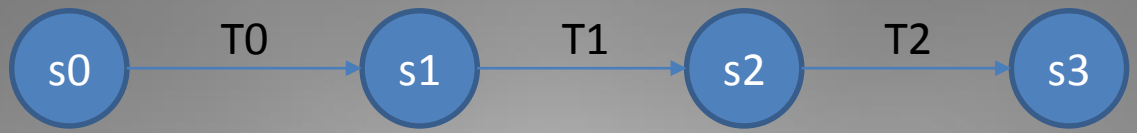
- Improve consistency and performance of databases (Spanner, CockroachDB)
- High throughput distributed ledgers (blockchains)

Overview

- Logical view of databases
- Realistic view of databases
- Bridging the two views using timestamps
- Case studies of Spanner and CockroachDB

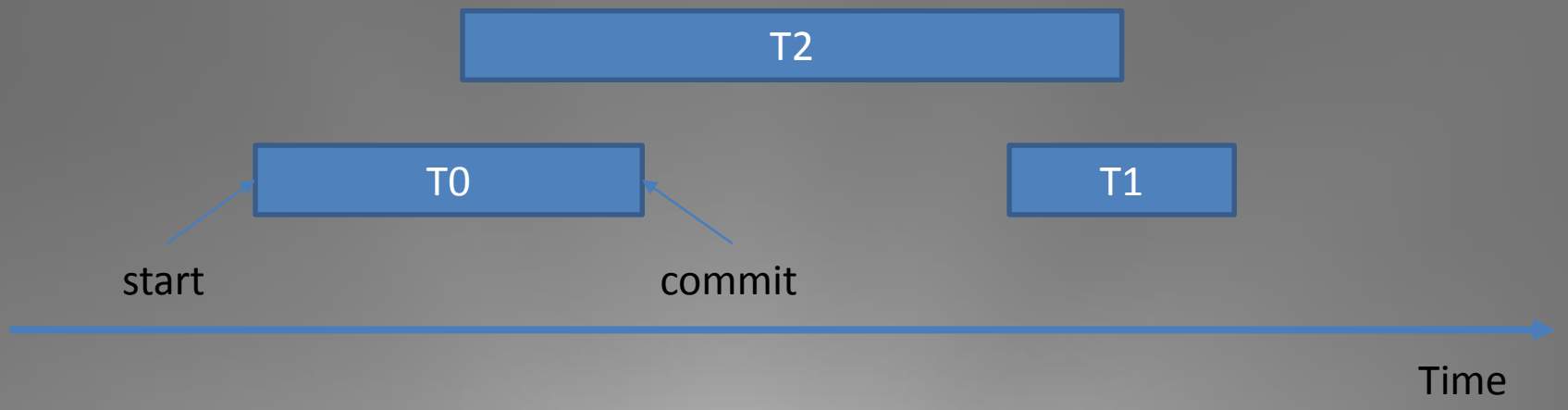
Logical view of database

- Database
  - A set of records
- Transaction
  - A sequence of reads and writes
  - Atomic transformation of the state of the database
- Snapshot read
  - Read out one of the states



Realistic view of database

- Data is distributed on multiple servers
- Despite the logical atomicity of a transaction, it takes time to finish
- Concurrent transactions

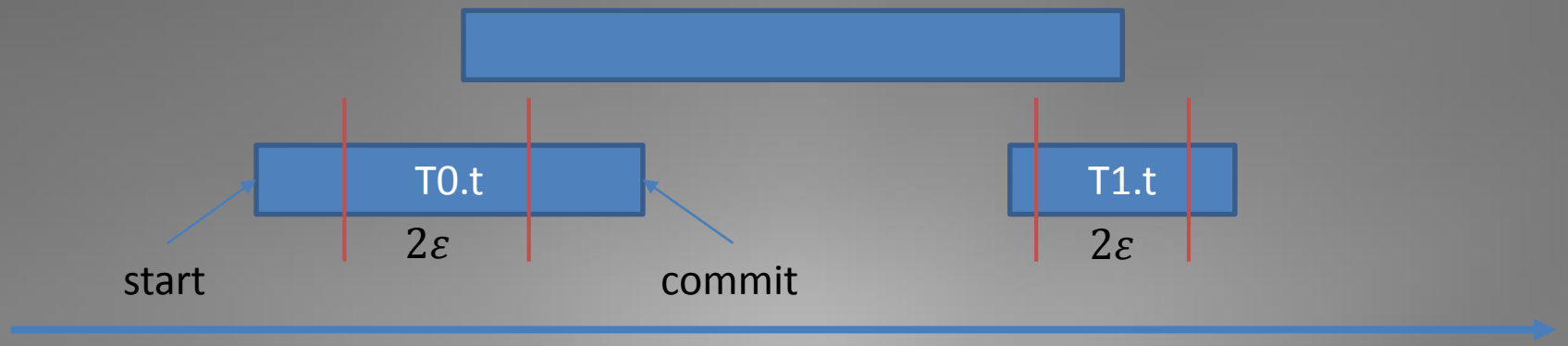


Bridging the two views using timestamps

- Partial order of transactions
  - Linearizability: A < B if A commits before B starts (Spanner)
  - Serializability: all transactions are parallel (CockroachDB)
    - Serializability is still difficult: Once serialized, history needs to be respected
- Clocks
  - Physical clocks synced to some precision (Spanner)
  - Lamport clocks
  - Hybrid logical clocks (CockroachDB)
- Execute transactions
  - Enforce partial order
  - Deal with transaction contentions
    - Read-write, write-read, write-write
  - Implementation and performance depends on partial order requirements and clock quality

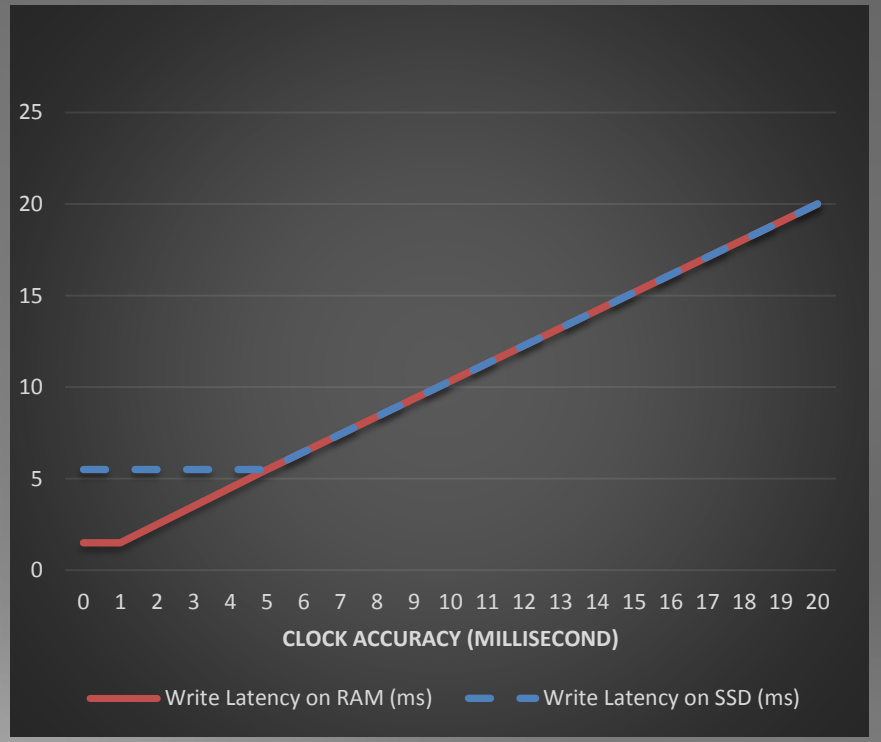
Spanner

- Globally synchronized clocks with 7 ms precision
- First DB to support linearizability at the global scale
- Good performance – short lock holding time



Linearizability in CockroachDB

- CockroachDB also has a linearizable mode that uses the same mechanism as Spanner
- Experiment
  - Three servers form a CockroachDB database
  - Data store is in either SSD or RAM
  - Repeatedly update a record under no contention
- The greater between clock accuracy and I/O latency bounds DB write latency
- Transaction support on RAMCloud?



Serializability in CockroachDB

- CockroachDB uses Hybrid logical clocks, and assumes 250ms accuracy on the physical clock (NTP)
- Cannot afford to wait out clock uncertainty when holding locks, thus runs on serializable mode by default
- Implements complex retry mechanisms to work around clock problems
- One problem involving the clock uncertainty is reading at the present time

Serializability in CockroachDB (cont')

- Experiment
  - 32 servers form a CockroachDB database
  - 128 data records, on average 4 records per server
  - 128 threads, each repeatedly update one record
  - Try to take a snapshot of the 128 records

Clock accuracy	1ms	10us	100ns
Retry rate	99.30%	4.74%	0.08%

Summary

- Precise clock sync in database helps with
  - Stronger ordering of transactions
  - Simpler implementation
  - Better performance
- Specifically to existing databases
  - Precise clocks can increase write speed and throughput in Spanner
  - Precise clocks can reduce read restarts, and thus reduce read latency and increase system throughput in CockroachDB

Welcome to our poster for more details and Q/A