Use of Time in Distributed Databases —don't fall behind the times

Murat Demirbas

MongoDB Research
http://mongodb.com

January 31, 2025

Alignment Problem in Distributed Systems

- Nodes execute concurrently with no shared state and no common clock
- Time provides a shared reference frame for ordering events across nodes

Findings

1. Modern systems increasingly leverage synchronized clocks for time
2. Sync clocks improve performance by obviating communication for ordering
3. Trend toward more sophisticated time usage continues
4. Cloud-based time services are getting very precise & widely available

5-part survey: `https://muratbuffalo.blogspot.com/2024/12/use-of-time-in-distributed-databases.html`

Logical Clocks

- Captures causality (Lamport'78) & enables serving consistent snapshots
- Limited by being disconnected from real time

- Vector clocks (Dynamo'07, ORBE'13)
- Dependency graphs (COPS'11, Kronos'14)
- Epoch services (Chardonnay'23)

Physical Clocks

- Sync clocks create a global time reference & allows coordination with less communication

- Tightly synchronized (Spanner'12, Aurora Limitless/DSQL'24)
- Loosely synchronized (most production distributed databases)
- Hybrid logical clocks (most production distributed databases)

Clock Synchronization

Network Time Protocol (NTP, 1985)

- 10s of milliseconds precision; vulnerable to network delays

Google TrueTime API (2012)

- GPS and atomic clocks as time source
- 6ms clock uncertainty

Precision Time Protocol (PTP)

- Hardware timestamping; 50μ uncertainty (AWS TimeSync'23)

Time-based Alignment (most distributed databases)

Consistent Snapshots

- Effortless MVCC implementation; Lock-free strong-consistency reads

Conflict Detection

- Compare timestamps for concurrent updates; Efficient OCC implementation

Fencing Mechanism

- Prevent stale operations through time-based leases

Time-Based Speculation

- Deadline-Ordered Multicast (Nezha'23)
- Predictive commit timing (Cassandra Accord'23, DSQL'24)
- Speculative execution with fallback paths

- Better common-case performance; Maintained consistency guarantees
- Significantly reduced coordination overhead

Spanner et al.

Spanner (Google)

- External consistency via commit-wait of clock uncertainty
- Lock-free snapshot reads

CockroachDB

- NTP + Hybrid Logical Clocks (HLCs)
- No commit-wait due to NTP uncertainty; dynamic timestamp adjustment
- https://muratbuffalo.blogspot.com/2024/12/utilizing-highly-synchronized-clocks-in.html

AWS offerings

DSQL'24

- Snapshot Isolation: transactions read from storage with T_{start}
- OCC validation by adjudicators using T_{commit}
- Adjudicator responsibility over key-ranges fenced by using time-range leases
- AWS timesync provides 50μ clock uncertainty

DynamoDB'18

- Timestamp-based OCC (VLDB'80)
- One-shot transactions; two-phase lock-free read txns

Use time for performance, not correctness

Key Requirement: Monotonic clock!

- Hybrid Logical Clocks (HLC) act as insurance

Clock bound APIs provides safeguards:

- Multiple failure requirements for correctness violation

Trends

- Accelerating adoption of synchronized clocks
- More sophisticated time usage
- Cloud-based time services

Future Research Areas

- Isolation-performance tradeoffs
- Smoother degradation modes in time-based speculation

https://muratbuffalo.blogspot.com/2025/01/use-of-time-in-distributed-databases_14.html