Clock & Data Synchronization
from High Frequency Trading to Distributed Ledgers

Victor Yodaiken | CEO | sales@fsmtime.com
A Different Take on “Distributed Ledger”

- Often associated with things like
  - “bitcoin”,
  - blockchain,
  - and untrusted participants

- There is a much wider application domain – particularly in financial trading and related enterprise areas.
A Different Take on “Distributed Ledger”

- **Wikipedia**: “*The distributed ledger database is spread across several nodes (devices) on a peer-to-peer network,*
- *where each replicates and saves an identical copy of the ledger,
- *and updates itself independently.*
- **The primary advantage is the lack of central authority.**”

The “blockchain” is a method for getting the result of a peer-to-peer system with no central authority.
Traditional Distributed Database Systems

Are centralized via “data synchronization”

- Each modification of the database (“transaction”) involves one participant obtaining locks from all the other participants to ensure data integrity

Server

Request and obtain locks and then commit transaction and release locks

Server

The “lock manager” is a central authority
Basic Clock Sync in Finance

- **Problem:** multiple computers trading at the same time each producing transactions

- Require an “in order” consolidated ledger

Consolidated Trading Ledger: the more analytics that is done on these the more that it is necessary for them to be in order
Basic Clock Sync in Finance

• Traditional “computer science” solution is centralized authority: lock manager. Only one server gets to carry out a transaction at any time. (scales poorly)

Trading Venue or Partner <-> Server <-> Lock Manager <-> Trading Venue or Partner

Trading Records: arrive in order
Basic Clock Sync in Finance

- **Synchronized clocks**: servers advance in parallel and “timestamp” transactions using synchronized clocks.

Trading Records:
- arrive in any order but can be sorted by time
Basic Clock Sync in Finance

- Same solution can be extended to have multiple ledgers that update in parallel and can be ordered independently and data synchronized using timestamps.

Each ledger orders by timestamp and can locate missing records on peer copies of the ledger.
Basic Clock Sync in Finance

- This is a **distributed ledger problem** that can be solved using synchronized clocks.

Trading Records: arrive in any order but can be sorted by time.
Basic Clock Sync in Finance

- This is a "distributed ledger" solution that requires trust (the timestamps have to be accurate) but no central authority.

Each ledger orders by timestamp and can locate missing records on peer copies of the ledger.
Market Developments

Drive a need for higher precision & higher reliability

- Faster transactions - to order them requires more precise timestamps
- More venues - increase total transaction rate so ...
- Global trading - even more venues
- Regulations - regulators want to see “consolidated” ordering of transactions across firms.
- Failures in clock sync bring whole systems down so reliability and fault tolerance becomes essential
Clock Sync for Networked Computers

• NTP and NTPd 1980s –
  • millisecond accuracy,
  • Ubiquitous

• PTP early 2000s,
  • tens of microseconds
  • Hardware assist

• Post 2008 advanced methods:
  • both NTP/PTP below 1 microsecond
  • Variety of fault tolerance methods
  • Instrumentation - detection of errors
New Opportunities

• Need for more precision and more resilience to operate basic trading

• On the other hand, better clock sync opens up new opportunities to use high speed, clock enabled, distributed technologies.

• Better clock synchronization tools open up new applications in the enterprise.
Clocks for Decentralized Data Synchronization

- Academic paper: Liskov 1993
- Relying on NTPd
- A series of methods for solving distributed computing problems using clock sync.
- Limited by slow networks, millisecond accuracy.
- Later: common use for “coordinator” selection in distributed consensus (e.g. Paxos)
- Widespread use in finance since 2000s
- Google Spanner 2015
Sketch of an Application Example

Distributed Key-Data Database

• Distributing reads is no problem

• How to handle writes (modifications) is the hard part

• System:
  • Multiple (distributed) servers
  • Assume: system fits the common pattern of many more reads than writes
  • Read and write requests go to any server

• Solution: time domain multiplexing
  • One distinguished (possibly revolving) server to control writes
  • Each read server expects a write update every X time units. Executes writes before returning to service reads. Needs clock sync to work.
Simple Algorithm (for illustration)

- Each server starts with a complete copy of the data
- All servers begin to process requests at the same time
- Every $T$ time units all servers look for write commands from the next write controller and process them during a window
- The write controller duty is cycled among the servers
- Reads can be executed at any time outside the window if they have an acceptable timestamp
- Guarantee: Once a writer has received an ack for a write, no reads will get stale data
No Data Locking Needed

- One writer at a time
- All servers wait for write data periodically synchronized by the clock
- Reads operate in parallel in read cycles
- Each write cycle goes to next writer
- Missing a write cycle, a server stops serving reads until it catches up
Option: Add Blockchain & Encryption

- Then distributed ledgers that are data synchronized with no central control in peer to peer network.
Tradeoff

• Higher speed flexible database operations “without central authority” (but in trusted environment)

• Failures in clock sync accuracy can cause problems in data integrity.

• Clock synchronization systems need to provide resilience and instrument to detect and warn of problems that cannot be automatically fixed.

• Need automatic failover, alerts, comprehensive logging (for forensics).
Time and Money | Jan 28, 2019 | NYC

THANKS!

Victor Yodaiken | CEO | sales@fsmtime.com