The Mystery of Time in Distributed Systems

Part II

Logical Clocks

mechanism for capturing causal relationships.
(cause and effect)

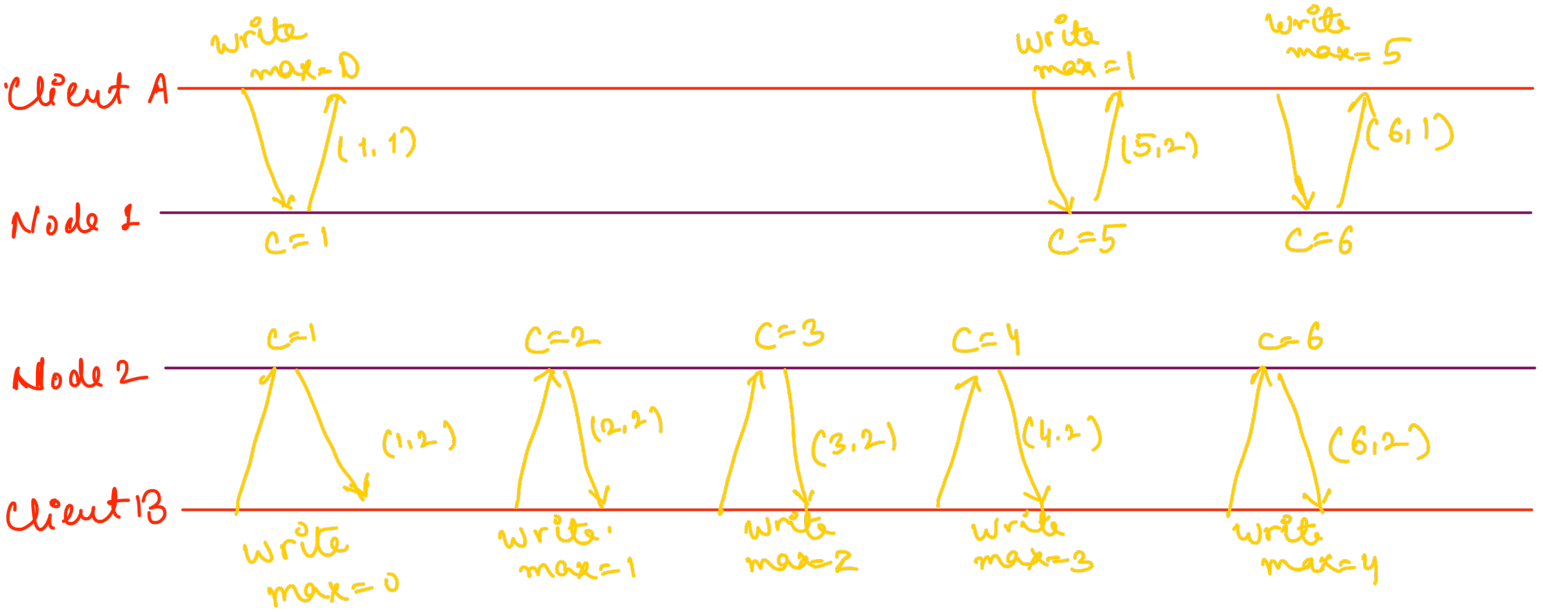
chronological and event ordering in order of event occurrence in time.

(1) Lamport Timestamps

Combination of (counter, node Id)

- counter: unique across single node
- (counter, node Id): unique across all nodes

For node A and node B
↳ ① Compare counters to decide order
↳ ② If counters are same, compare node Id.
(example lexicographic)



→ every node and every client keeps track of maximum counter value it has seen so far.

→ If node receives request / response greater than own counter value, it updates local counter value to that maximum.

Lamport timestamps provide **total ordering.**
you'll always be able to figure out which event occurred first.

Disadvantage — can't distinguish whether two operations are concurrent or casually dependent.

If $L(a) < L(b)$
can't say for sure $a \rightarrow b$ or $a \| b$
a occurred first — both concurrent

(2) Vector Clocks → solves above disadvantage.

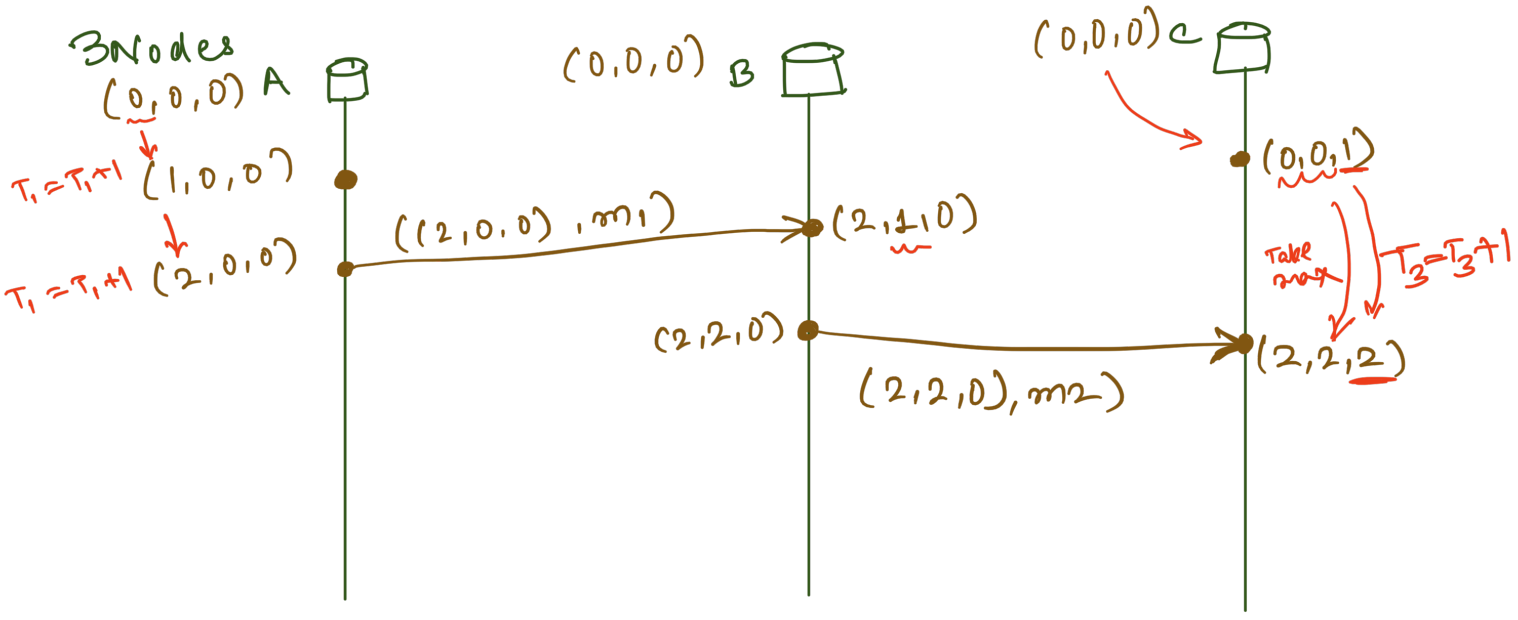
n nodes in system $\{N_1, N_2, N_3 \ldots N_n\}$

Timestamp of Event 'a', $V(a) = \{t_1, t_2, t_3 \ldots t_n\}$

t_i = no of events by node i

on event at N_i, $T_i = T_i + 1$

↑ vector timestamp is attached to every msg.



Comparing event occurrence T and T'

$T = T'$ if $T_i = T'_i$ for $i \in \{i \dots n\}$

$T \leq T'$ if $T_i <= T'_i$

$T < T'$ if $T <= T'$ and $T \neq T'$

$T \| T'$ if $T \not< T'$ and $T' \not< T$

↳ events are incomparable ↑ partial order

for concurrent events, DB should keep both values and return both on next read.

client can decide which data to keep.

Downsides

↳ The vector data needs to be exchanged with every message along with multiple versions.

everything need to be stored.

↳ Cassandra don't use vector clocks.

↳ we will discuss this in upcoming videos.

See you again in next video

Subscribe for more

YT - MsDeep Singh

Happy Learning ☺