Hybrid Logical Clocks

Sandeep Kulkarni, Murat Demirbas, Deepak Madeppa, Bharadwaj Avva, Marcelo Leone

Vaibhav Bhagee

Logical Clocks and Vector Clocks

- Disregards the physical notion of time
- Progresses with the occurrence of events and not passage of time
- Captures the happens before relationship

Physical Clocks

- Leverage the system clocks at the nodes
- Issues of clock drift and clock jumps
- Periodic requirement of clock synchronisation
- Uncertainty intervals leading to inability to order events

True Time

- Proposed by google with Spanner
- Tight clock synchronisation using GPS clocks and Atomic Clocks
- Returns intervals [earliest, latest] instead of exact time
- Introduces delays if intervals overlap

Hybrid Logical Clocks

- Refines Physical and Logical Clock
- Preserves the causal ordering property of logical clocks

$$e \text{ hb } f \Rightarrow l.e < l.f,$$

- HLC timestamps are close to the NTP clock

$$|l.e - pt.e| \text{ is bounded.}$$

- Bounded space requirement
- Does not have clock jumps leading to non monotonic time updates

HLC Algorithm

Initially $l.j := 0; c.j := 0$

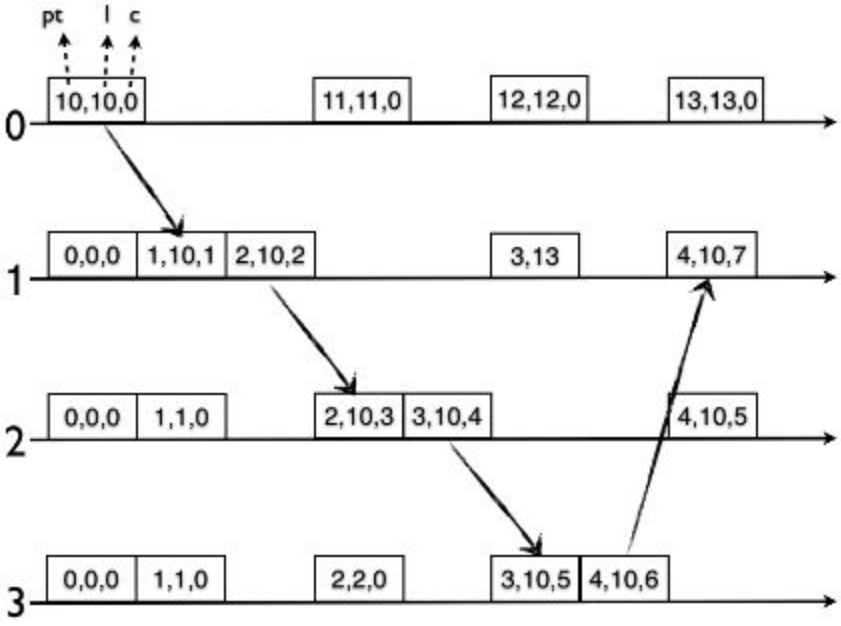
Send or local event

$l'.j := l.j;$
$l.j := max(l'.j, pt.j);$
If $(l.j = l'.j)$ then $c.j := c.j + 1$
Else $c.j := 0;$
Timestamp with $l.j, c.j$

Receive event of message m

$l'.j := l.j;$
$l.j := max(l'.j, l.m, pt.j);$
If $(l.j = l'j = l.m)$ then $c.j := max(c.j, c.m) + 1$
Elseif $(l.j = l'.j)$ then $c.j := c.j + 1$
Elseif $(l.j = l.m)$ then $c.j := c.m + 1$
Else $c.j := 0$
Timestamp with $l.j, c.j$

HLC Algorithm Example



HLC Properties

Theorem 1. *For any two events e and f, e hb $f \Rightarrow (l.e, c.e) < (l.f, c.f)$* $\square$

Theorem 2. *For any event f, $l.f \geq pt.f$* $\square$

HLC Properties

Theorem 3. *$l.f$ denotes the maximum clock value that f is aware of. In other words,*

$$l.f > pt.f \Rightarrow (\exists g : g \text{ hb } f \wedge pt.g = l.f)$$

- Proof by induction
- Case when **f** is a send event and **e** is the previous event
- Case when f is a receive event

$$l.e > pt.e \Rightarrow (\exists g : g \text{ hb } e \wedge pt.g = l.e)$$

- **e** is the previous event on the same node and **m** is the received message

$$l.f > pt.f \Rightarrow (\exists g : g \text{ hb } f \wedge pt.g = l.f).$$

HLC Properties

$$\text{For any event } f, |l.f - pt.f| \leq \epsilon$$

- Proof using clock synchronisation constraints
- We cannot have events e and f such that

$$e \text{ hb } f \text{ and } pt.e > pt.f + \epsilon$$

- By the previous theorem

$$l.f > pt.f \Rightarrow (\exists g : g \text{ hb } f \wedge pt.g = l.f).$$

HLC Properties

Theorem 4. *For any event f,*

$$
\begin{aligned}
& c.f = k \wedge k > 0 \\
\Rightarrow \quad & (\exists g_1, g_2, \cdots, g_k : \\
& \quad\quad (\forall j : 1 \leq j < k : g_i \text{ hb } g_{i+1}) \\
& \quad \wedge\ (\forall j : 1 \leq j \leq k : l.(g_i) = l.f) \\
& \quad \wedge\ g_k \text{ hb } f)
\end{aligned}
$$

- Proof by induction
- Cases: **f** being a send or a receive event
- Idea is similar to the property of logical clocks
- Number of events causally preceding the current event with same **pt**

HLC Properties

Corollary 2. *For any event f,*

$$c.f \leq |\{g : g \text{ hb } f \wedge l.g = l.f)\}|.$$

Corollary 3. *For any event f, $c.f \leq N * (\epsilon + 1)$*

- First corollary directly follows from the proof
- For proof of second, we use the fact that due to clock synchronisation

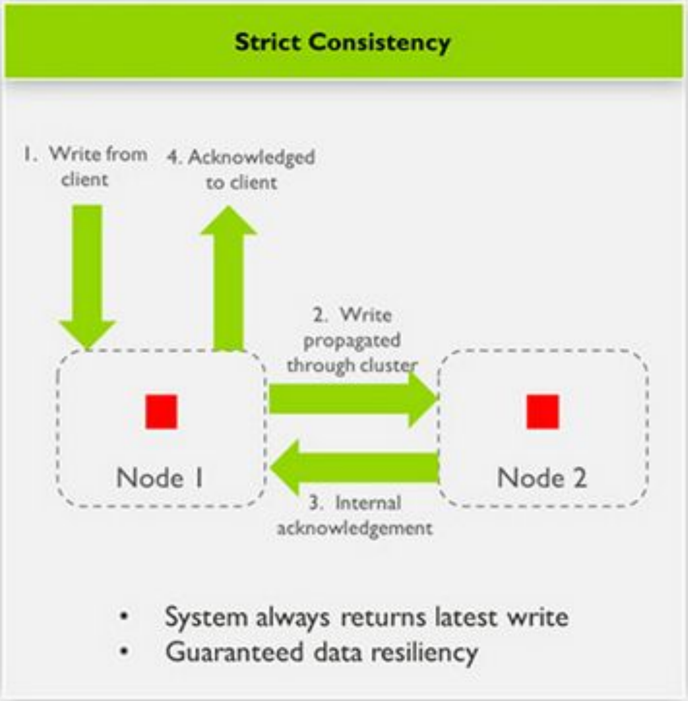
$$[l.f, l.f + \epsilon]$$

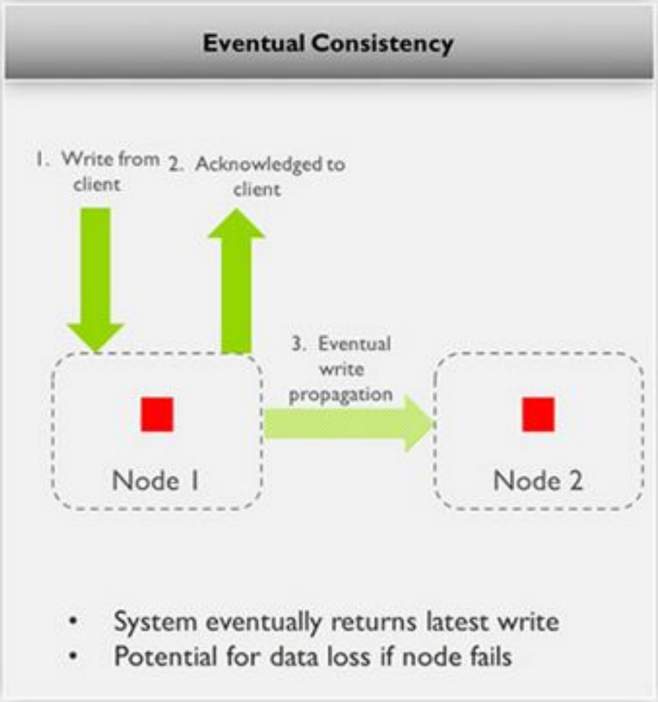
- Physical time gets incremented by at least 1 between two events on a node

Applications of HLC: Database Consistency

- **Atomicity:** Transactions are a single unit, either going to completion or not.
- **Consistency:** Database should go from one consistent state to another.
- **Isolation:** Concurrent transaction execution mapped to a serial order
- **Durability:** Once a commit happens, the changes should be persistent.

Applications of HLC: Database Consistency





Applications of HLC: Achieving strict consistency

- Consensus on the operation log and the written value: RAFT
- Concurrency control: contention issues with reader-writer locks
- Need for wait free transaction ordering
- Idea: Multi Version Concurrency Control (MVCC) with HLC timestamps

Operation: R(A, 350)

A@350

Key	Timestamp	Value
A	400	“current_value”
A	322	“old_value”
A	50	“original_value”
B	100	“value_of_b”

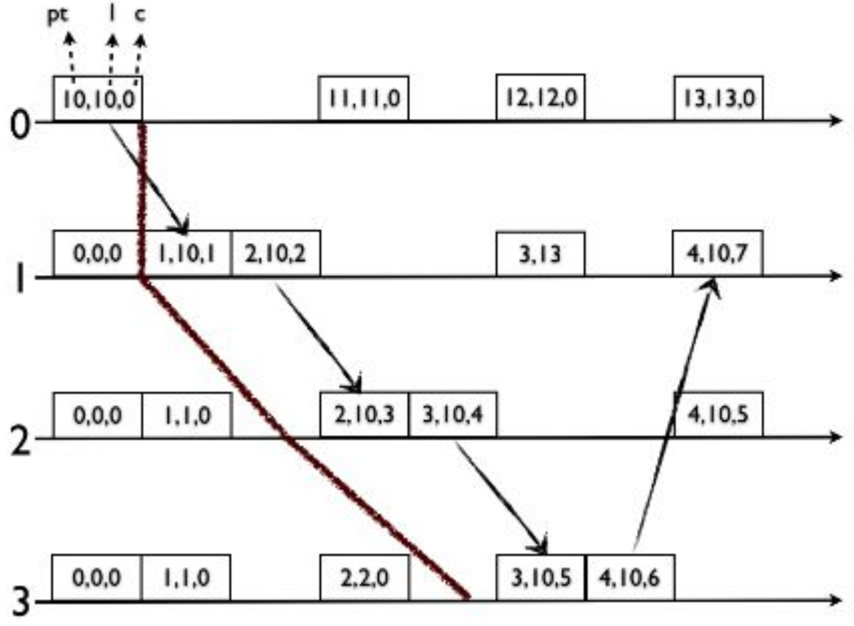
Result = “old_value”

Applications of HLC: Global Snapshots

- Request to obtain a snapshot of the data at a time **t**
- Idea of introducing dummy events between events **e** and **f** on the same node
- Every event has **l** between **l.e** and **l.f** with **c** as 0.
- Take snapshot at **l = t** and **c = 0**.

Applications of HLC: Global Snapshots

- Consistent snapshot for t = 10



Applications of HLC: Event Ordering Service

- **Kronos:** EuroSys 2014 paper
- Can use HLC timestamps to schedule events and register with Kronos
- HLC properties ensure consistent cuts in the event trace
- Snapshots, Predicate detection etc. become local operations