Physical and Logical Time

> Super imcomplete :(

Time

- Time needs memory
- Time is local
- Time synchronization is harder at distance
- Time is a bad proxy for causality

Wall-clock time

Clock Drift

- Clock Drift: refers to drift between measured time and a reference time for thhe same measurement unit
- Quartz clocks: From 1e-6 to 1e-8 seconds per second.
- Atomic clocks: About 1e−13 seconds per second.

External vs. internal synchronization

External synchronization states a precision with respect to an authoritative reference. Internal synchronization states a precision among two nodes (if one is authoritative it is also external)

External	Internal
States a precision with respect to an authoritative reference	States a precision among two nodes (if one is authoritative it is also external)
For a band D > 0 and UTC source S we have mod(S(t) − Ci(t)) < D	For a band D > 0 we have mod(Cj(t) − Ci(t)) < D

Monotonicity

Correcting advanced clocks can be obtained by reducing the time rate until aimed synchrony is reached.

Synchronization

Synchronous System

$trans$ → transit time, ∈ $[tmin, tmax]$
t → origin time
u → uncertainty

- Using $t + tmin$ or $t + tmax$, $u = tmax - tmin$
- But, using $t + (tmin + tmax)/2$ the uncertainty becomes $u/2$

Asynchronous system

$trans$ → transit time, ∈ $[tmin, +\infty]$

1. Send a request mr that triggers response mt carrying time t
2. Measure the roud-trip-time of request and reply as tr
3. Set clock to $t + tr/2$ assuming a balanced round trip

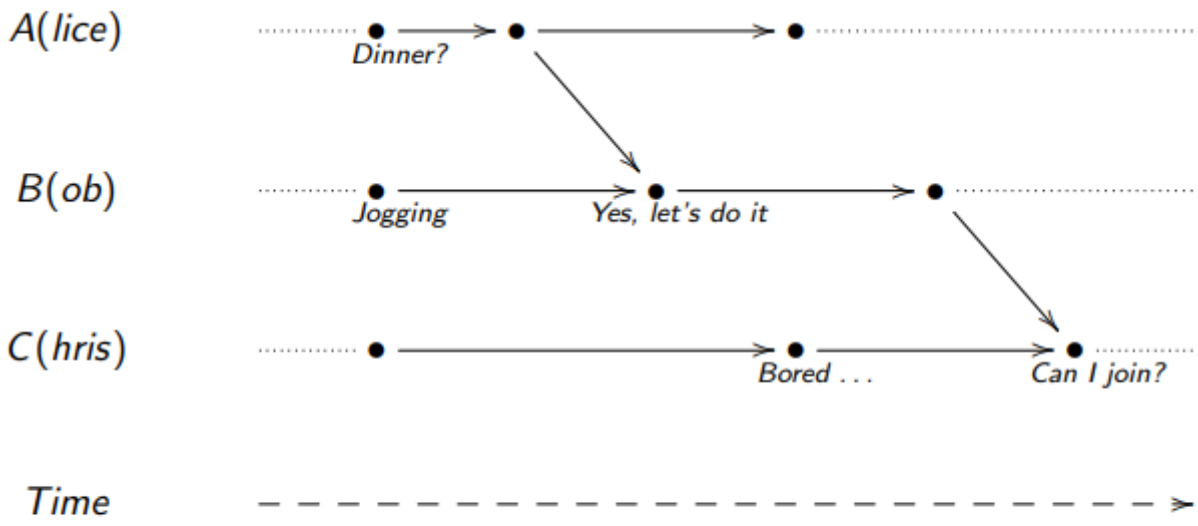
- Precision can be increased by repeating the protocol until a low tr occurs

Berkeley Algorithm

A coordinator measures RTT to the other nodes and sets target time to the average of times. New times for nodes to set are propagated as deltas to their local times, in order to be resilient to propagation delays.

Causality

1. Alice decides to have dinner
2. She tells that to Bob and he agrees
3. Meanwhile Chris was bored
4. Bob tells Chris and he asks to join for dinner



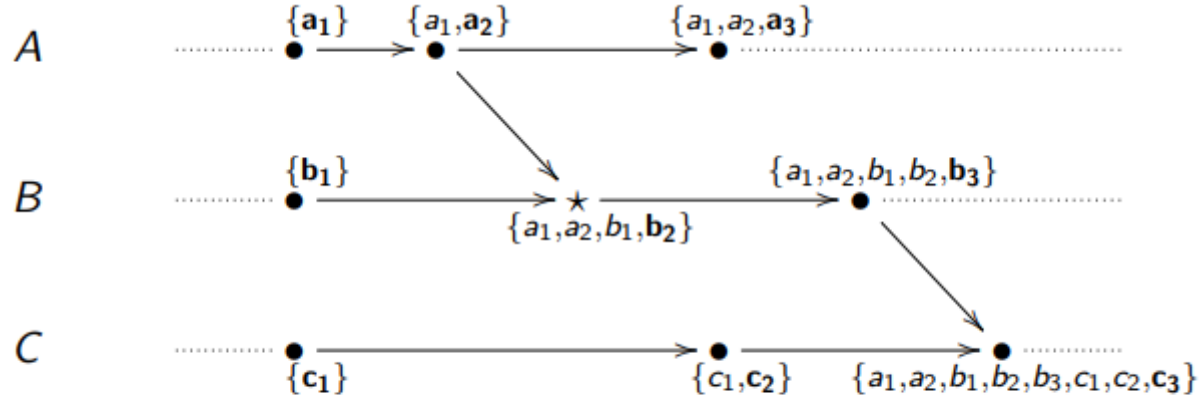
Causally: "Alice wants dinner" ∥ "Chris is bored"
Timeline: "Alice wants dinner" < "Chris is bored"

- Past statements only potentially influence future ones
  How to track it?

Causal Histories

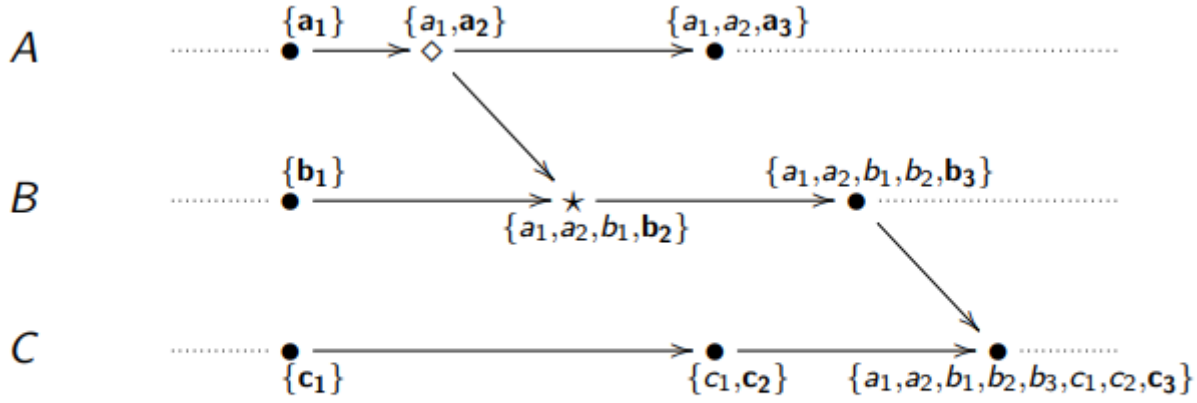
- Collect memories as sets of unique events
- Set inclusion explains causality → {a1, b1} ⊂ {a1, a2, b1}
- Properties:
  - You are in my past if I know your history
  - If we don't know each other's history, we are concurrent
  - If our histories are the same, we are the same

Message Reception



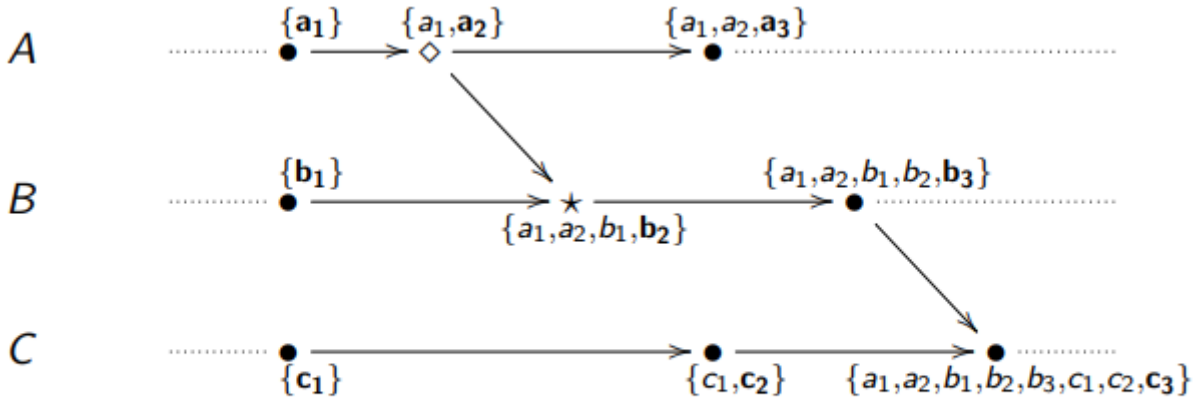
$\star$ receive $\{a_1, a_2\}$ at node b with $\{b_1\}$ yields $\{b_1\} \cup \{a_1, a_2\} \cup \{b_2\}$

Causality Check



Check $\diamond \rightarrow \star$ iff $\{a_1, \mathbf{a_2}\} \subset \{a_1, a_2, b_1, \mathbf{b_2}\}$

Faster causality check



Check $\diamond \rightarrow \star$ iff $\mathbf{a_2} \in \{a_1, a_2, b_1, \mathbf{b_2}\}$

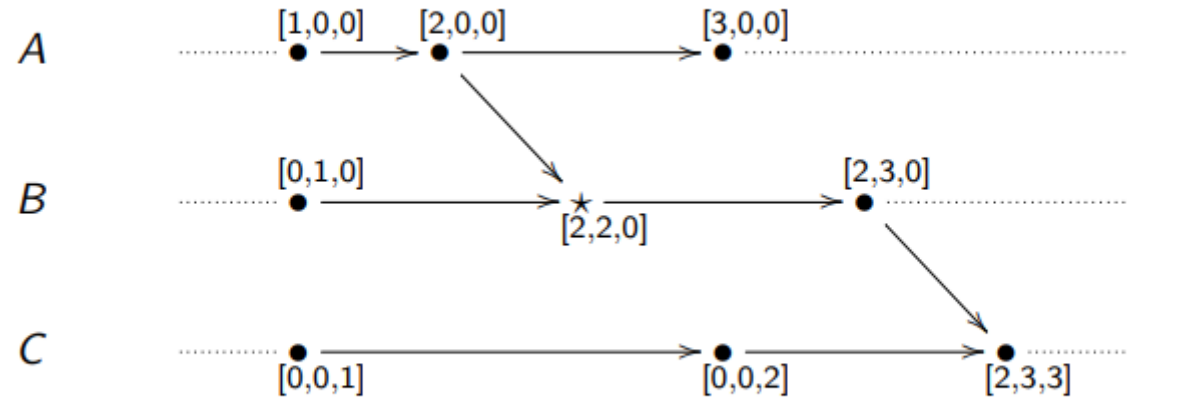
Note: {en} ⊂ Cx ⇒ {e1, ..., en} ⊂ Cx

Vector clocks

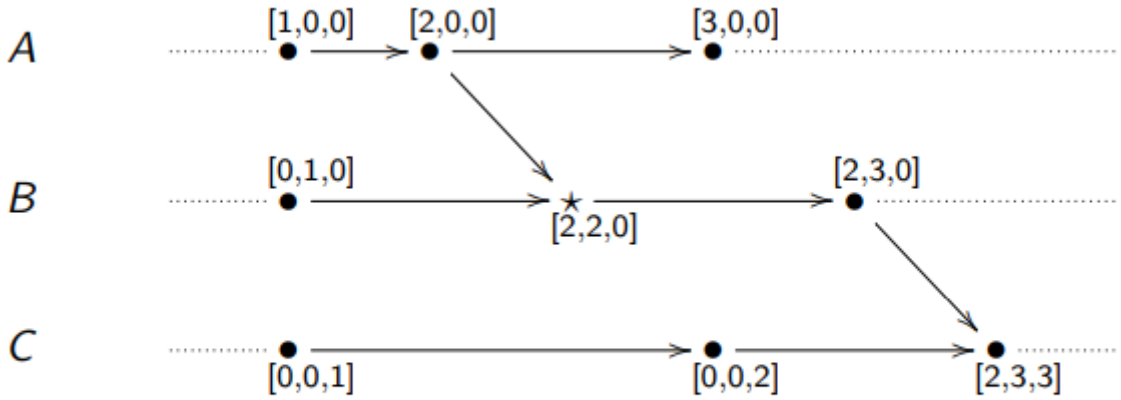
- {a1, a2, b1, b2, b3, c1, c2, c3}
- {a 7→ 2, b 7→ 3, c 7→ 3}
  Finally a vector, assuming a fixed number of processes with totally ordered identifiers
- [2, 3, 3]

Message Reception

Set union becomes **join** ⊔ by point-wise maximum in vectors

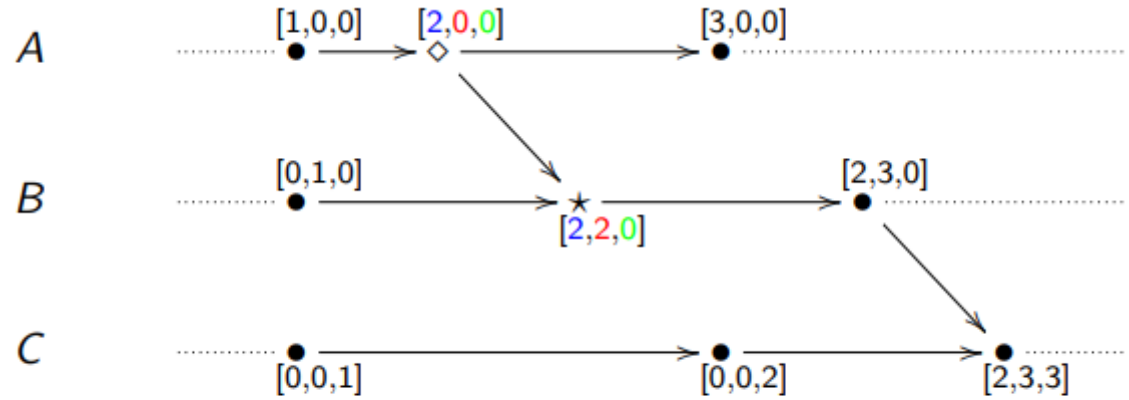


$\star$ receive $[2, 0, 0]$ at b with $[0, 1, 0]$ yields $\mathrm{inc}_b(\sqcup([2, 0, 0], [0, 1, 0]))$



$$\text{inc}_b(\sqcup([2,0,0],[0,1,0])) \equiv \text{inc}_b([2,1,0]) \equiv [2,2,0]$$

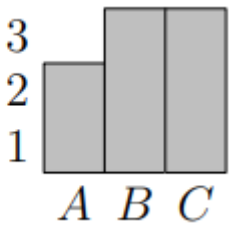
Causality Check



Check $\diamond \rightarrow \star$ iff point-wise check $2 \leq 2, 0 \leq 2, 0 \leq 0$

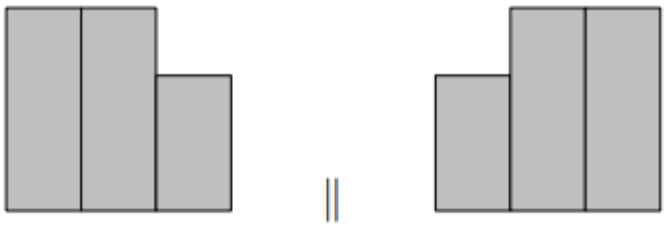
Graphically

$[2,3,3] \equiv \{A \mapsto 2, B \mapsto 3, C \mapsto 3\} \equiv$



Graphically Comparing

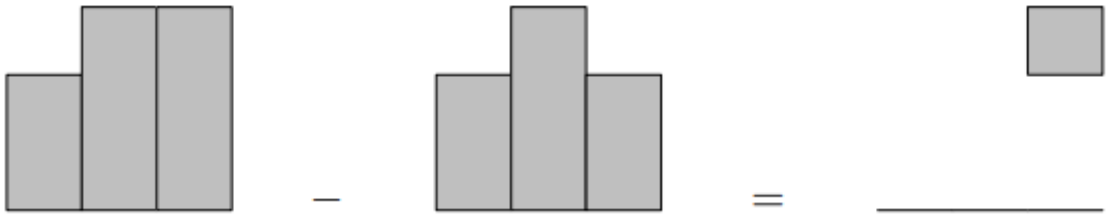
$$[3,3,2] \parallel [2,3,3]$$



$$[2,3,2] < [2,3,3]$$

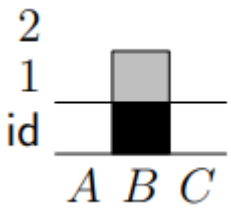
Graphical Difference

$$[2,3,3] - [2,3,2] = \{c3\}$$

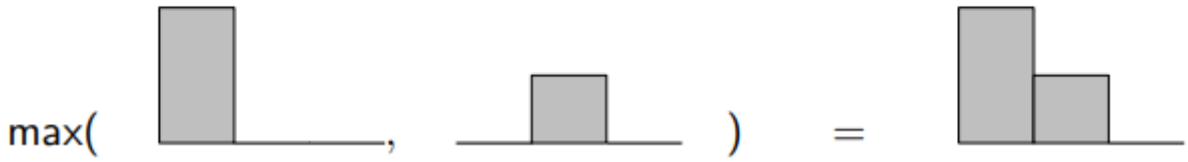


Graphical message reception

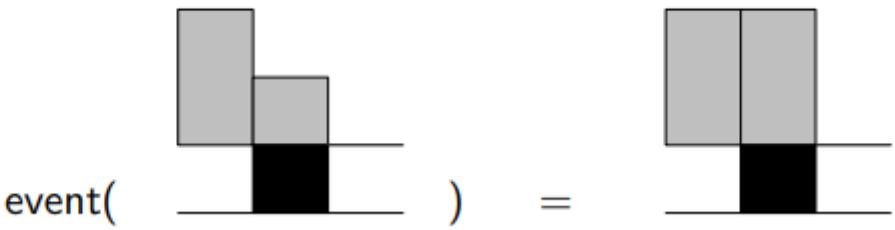
Node b, with [0, 1, 0] is receiving a message with [2, 0, 0]
We need to combine the two vectors and update b entry
Node b, with [0, 1, 0]:



Point-wise maximum of $[2,0,0]$ and $[0,1,0]$



Register a new event on the result

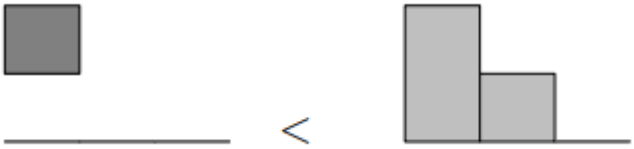


Faster Causality Check

Comparing vectors is linear on the vector size
This can be improved by tracking the last event

Vector clocks with dots

- [2, 0, 0] becomes [1, 0, 0]a2
- [2, 2, 0] becomes [2, 1, 0]b2
- The causal past excludes the event itself



Registering Relevant Events

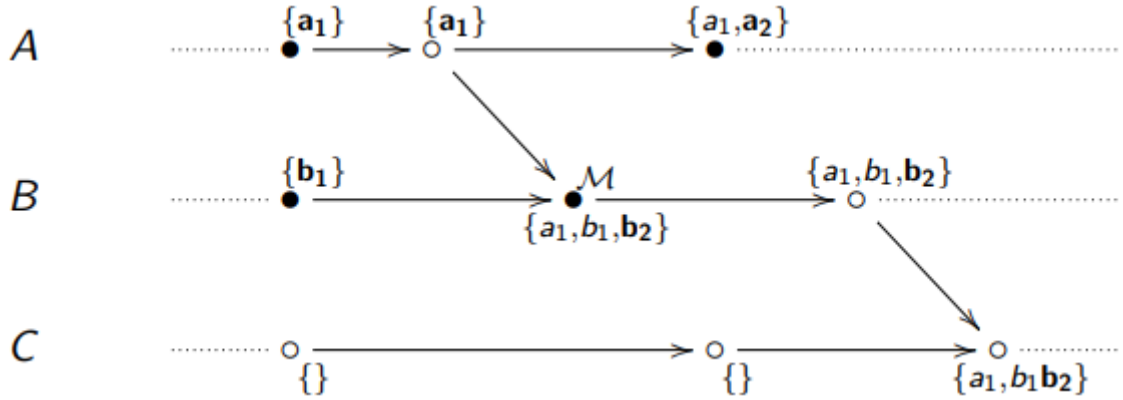
- Track only update events in data replicas
- Applications in:
  - File-Systems
  - Databases
  - Version Control

Dynamo

Causally tracking of write/put operations

Causal histories with only some relevant events

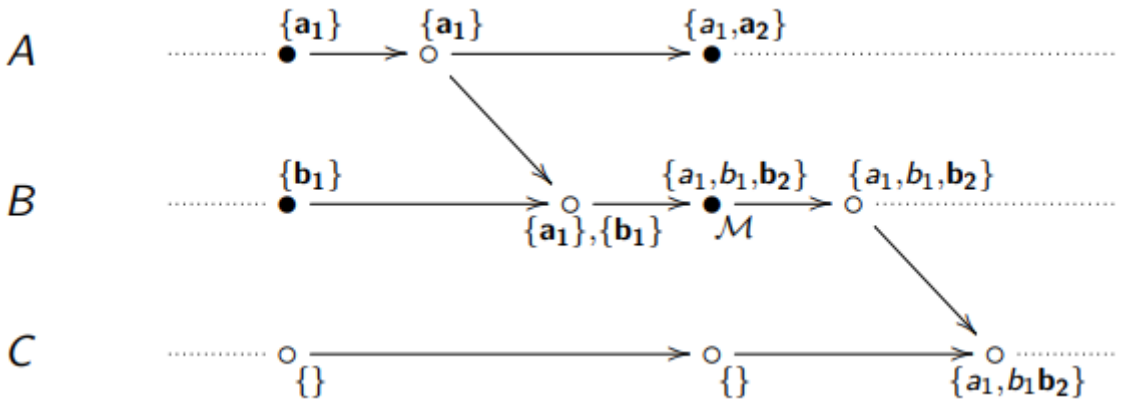
Relevant events are marked with a • and get an unique tag/dot
Other events get a ◦ and don't add to history



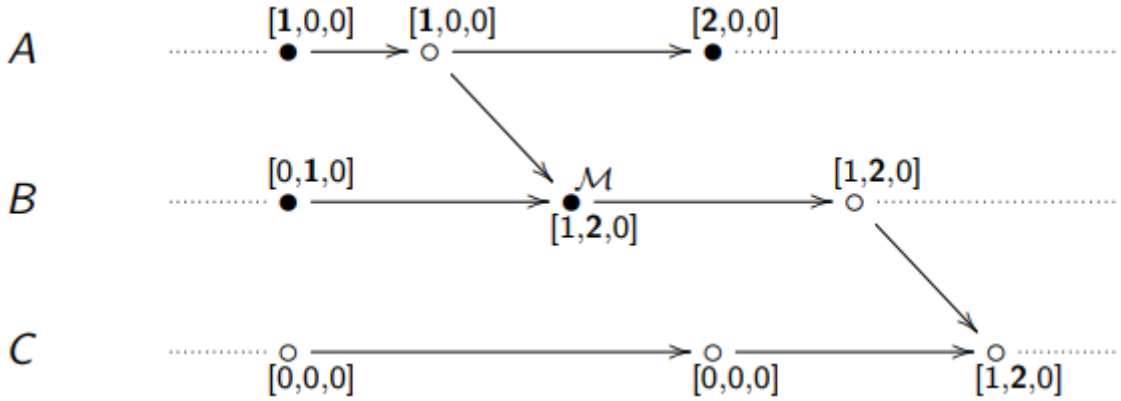
Concurrent states {a1} k {b1} lead to a • marked merge M
Causally dominated state {} → {a1, b1, b2} is simply replaced

Causal histories with versions not immediately merged

Versions can be collected and merge deferred
Causal histories are only merged upon version merging in a new •



Version Vectors



Scaling Causality

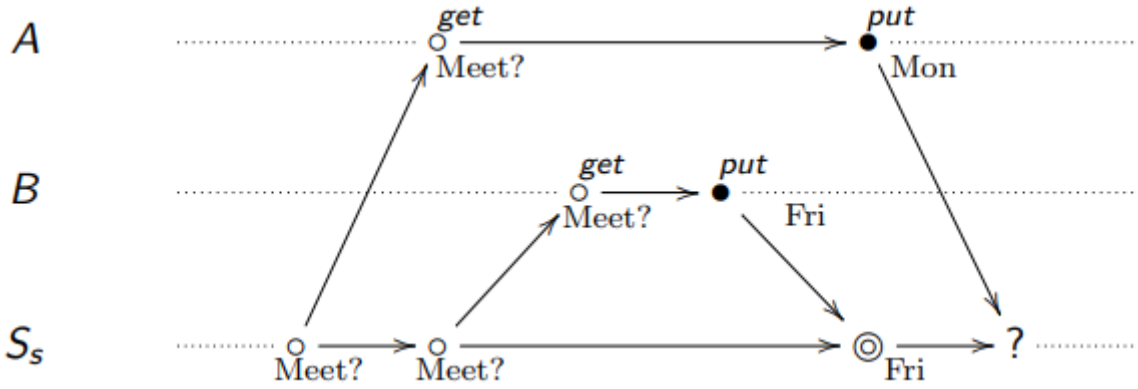
Scaling at the edge (DVVs)

- Mediate interaction by DC proxies
- Support failover and DC switching
- One entry per proxy (not per client)

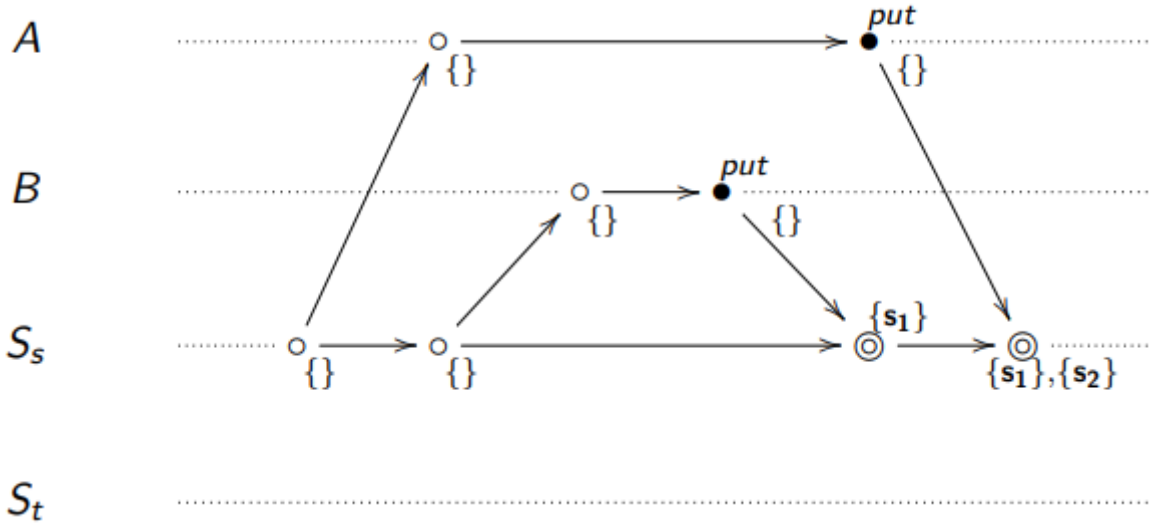
Dynamic concurrency degree (ITCs)

- Creation and retirement of active entities
- Transparent tracking with minimal coordination
- Causality tracing under concurrency across services

Dynamo (get/put interface)

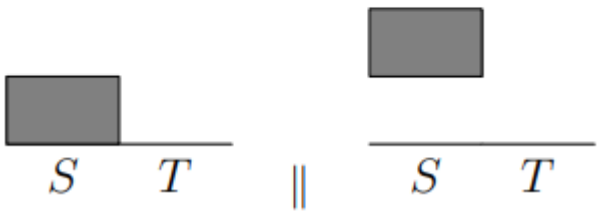


- Conditional writes
- Overwrite first value
- Multi-Value ([1, 0] || [0, 1], one entry per client!)

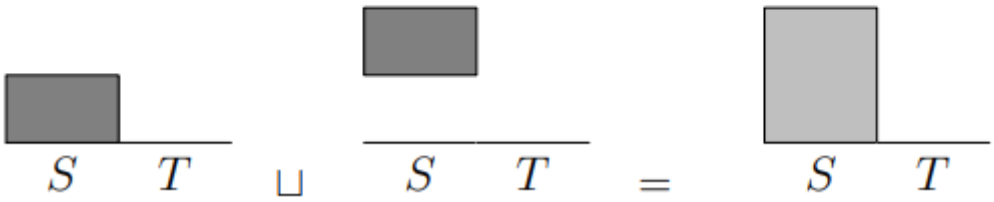


Dotted Version Vectors

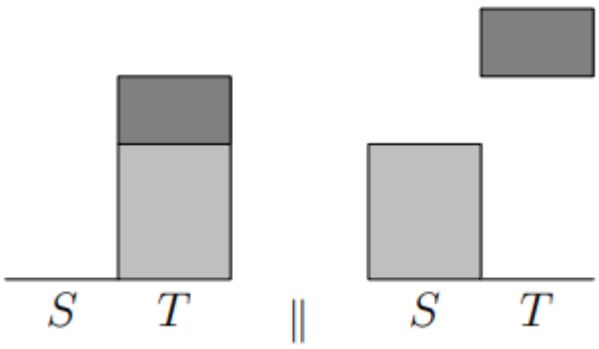
$$[0,0]s_1 \parallel [0,0]s_2$$



$$[0,0]s_1 \sqcup [0,0]s_2 = [2,0]$$

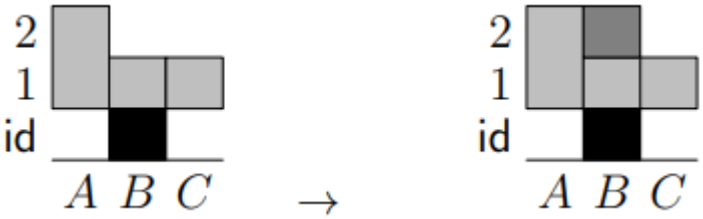


$$[0,2]t_3 \parallel [2,0]t_4$$



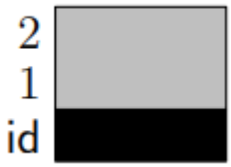
Dynamic Causality

Tracking causality requires exclusive access to identities
To avoid preconfiguring identities, id space can be split and joined



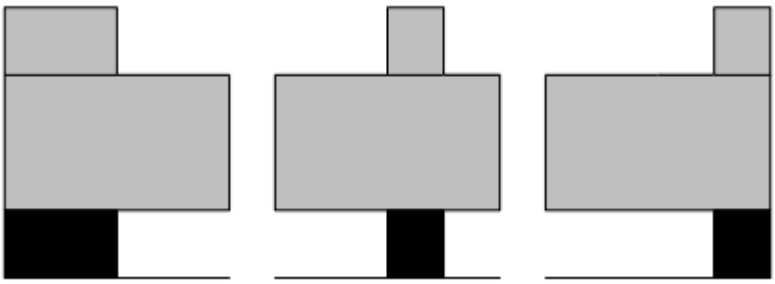
Interval Tree Clocks

A seed node controls the initial id space
Registering events can use any portion above the controlled id

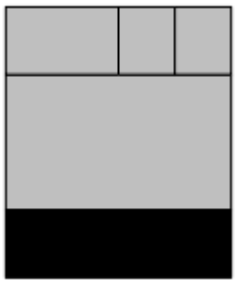


Ids can be split from any available entity

Entities can register new events and become concurrent



Any two entries can merge together eventually collecting the whole id space



Global Invariants on IDs

Causality characterization condition

Each entity has a portion of its identity that is exclusive to it.
Entity events must use at least a part of the exclusive portion.

Disjoint condition

A less general but more practical condition is that all identities are kept disjoint.

Summary

- Causality is important because time is limited
- Causality is about memory of relevant events
- Causal histories are very simple encodings of causality
- VC, DVV, ITC do efficient encoding of causal histories
- All mechanisms are only encoding of causal histories