Time and Snapshots
Today

Time
* logical clocks (CockroachDB, YugaDB)
* Vector clocks (Amazon's DynamoDB)

Event Ordering
* total, causal, FIFO
* linearizable, sequential, eventual
DNS

Edge cluster

Data Center

User session state:

Load Balancer

WebTier (front End Clusters)

Uses etcd to get server to Web Tier

Uses etcd to get user data

Today: ordering (consistent)

Storage Tier (back End Clusters)
Real Time

Bank Servers

$1

$2

behind by 5 mins

12:00

client

add (500)

OK (12:00)

12:01

transfer (600)

client

OK (11:56)

Server time

Server time
Global sequences

Before a $S_i$ processes a client event, the $S_i$ must get ID event.

$$e_i, s_i, 1$$
$$e_x, s_i, 2$$
$$e_x, s_i, 4$$
$$e_m, s_i, 5$$
$$e_m, s_i, 6$$
Ordering

Total order = all events can be sorted in the same order by every server in the system.

Partial order: servers only agree on a subset of events.

FIFO order

Causal ordering

only want to capture events that have a cause & effect relationship
**FIFO Ordering**

\[ e_1, e_2, e_3 \]

**FIFO:** event processed by a server are always in the same order but events by different servers can be in any order.

\[ e_5, e_6, e_7 \]
Real Time

People care about "time"

Distributed systems care about ordering:

- add → get? What is the right ordering

or

- get → add?
Types of events (in your system)

0 send msg
1 recv msg
2 local event

Diagram:

- State $s_1$ transitions to $s_2$ on event "local event"
rules for logical clocks

each server maintains a clock (c₀)

if event == (local | sendMsg)
  c++
  sendMsg(c)

if event == receive
  c = max(C, msg_c) ++

max(1, 2) ++
Vector clocks

vector of logical clocks
Vector clock

Rules for vector clocks

each server maintains a vector of clock \( VC[M] \)

initialize \( VC[i] = 0 \)

\( \text{initialize} \)

\( i = \text{me} \)

if (event == (Send | local))

\( VC[i]++ \)

\( \text{SendMsg}(VC) \)

else if (event == recv)

for \( j \) to \( N \)

\( VC[i] = \max(VC[j], \text{msg} \cdot VC[j]) \)

\( VC[i]++ \)
Vector Clock

rules for Vector Clocks

each server maintains a vector of clock \( VC(N) \)

initialize \( VC[i] = 0 \)
\( i = me \)

if (event == (Send | local))
\( VC[i]++ \)
\( sendMsg(VC) \)

else if (event == recv)
for \( j = 1 \) to \( N \)
\( VC[i] = \max(VC[j], msg.VC[j]) \)
\( VC[i]++ \)

Performance problems

1. maintain VC
2. send VC in each msg

How to determine \( N \) (when cluster is dynamic)

Kaboodle

1. Virtual nodes (issues with nodes leaving)
2. place a limit on your system (<1000 nodes)
3. fix the number of nodes (conflicting with all)

Dynamo
\[
\begin{align*}
\max & \quad \{ \max (3, 0) \} \quad \max (0, 0) \\
[0, 0, 1] & \quad [0, 0, 2]
\end{align*}
\]
Causal order

1. logical
2. vector clocks

Eventual

Causal ordering that allows to capture "happens before"
total FIFO Causal nothing (eventual)