GSCI 1380 : Day 11

Time
Today

Ordering
1. Total ordering
2. FIFO ordering
3. Causal ordering
Clocks
1. Logical clocks (Cockroach DB/YugaDB)
2. Vector clock (Amazon’s Dynamo)

Ordering based on clocks

Global Snapshots
1. Identifying inconsistent snapshots with vector clocks
Real-time is bad

1. Hard to synchronize servers
2. Time is not monotonic

look up the "leap second" bug

We want a monotonic clock
Event types / Assumptions

Event Types
1. Receive
2. Local (timer)
3. Send msg

Assumption
4. All servers can dump events to external store (S3)
Type of order (total/FIFO/causal)

Total = every server can place events in some order

\[ S_1 \]
\[ A_1, B_1, A_2, B_2 \]
\[ S_2 \]
\[ A_1, B_1, A_2, B_2 \]

\[ S_1 \]
\[ B_2, A_1, A_2, B_1 \]
\[ S_2 \]
\[ B_2, A_1, A_2, B_1 \]

FIFO = every server orders event in order that it was received by the local server; they may not agree on ordering

\[ S_1 \]
\[ A_1, A_2, B_1, B_2 \]
\[ S_2 \]
\[ B_1, B_2, A_1, A_2 \]

\[ S_1 \]
\[ A_1, B_1, A_2, B_2 \]
\[ S_2 \]
\[ A_1, B_1, B_2, A_2 \]

Both are FIFO because \( A_2 \) is always after \( A_1 \) and \( B_2 \) is always after \( A_1 \).

The server which got them: \( A_1, A_2, C, S_1 \)
Global ID Generator

Global server

1. When event happens
2. Get ID from global server
3. Process event

Logical clocks

Logical address this

Time ++
1. Use local time
2. Find ways to break ties

This is not monotonic!!!
rules for logical clocks

Each node maintains a clock \( c = 0 \)

if event == (local | sendMsg)

\[ c++ \implies \text{include clock in msg} \]

if event == recv

\[ c = \max (c, \text{msg}, c) + 1 \]

Sort based on clock \\
break ties using server IPs
Vector Clock

Rules for Vector Clocks
- Each server maintains a vector of clock (VC[i])
- Initialize VC[1] = 0
- If (event == (Send | local))
  - VC[i]++
  - Send msg(VC)
- Else if (event == recv)
  - For j to N:
    - VC[j] = 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)
1. Virtual nodes (issues with nodes leaving)
2. Place a limit on your system (<1000 nodes)
3. Fix the number of nodes (config with all)
\[ [1,0,0] \quad [2,0,0] \quad [3,0,0] \quad [4,3,0] \]

\[ S_1 \]

\[ [0,1,0] \quad [1,2,0] \quad [1,3,0] \]

\[ S_2 \]

\[ [0,0,1] \quad [0,0,2] \]

\[ S_3 \]

\[ e_4 = \max(e_3, \text{msg}_{eb}) = \max([1,3,0], [3,0,0]) \]

\[ \downarrow \]

\[ \circ [3,3,0] \]

\[ e_4 = [4,3,0] \]

Observations

Without msgs you can't learn about another server's clock
Causal ordering (benefits of VC)

“happens before”

A happens before B \[ A \wedge B \text{ are events} \]

\[ V_{C_A} \leq V_{C_B} \]

all clocks in \( V_{C_B} \) must be equal/greater than \( V_{C_A} \)

& at least one of \( V_{C_B} \) elements must be greater than \( V_{C_A} \)

\[ [1, 0, 1] \leq [1, 0, 2] \]

happens before the other (greater)
Summary

real time is bad (not monotonic)

FIFO / CAUSAL / TOTAL

Logical / Vector Clocks
People care about "time"

Distributed systems care about ordering:

\[ \text{add} \rightarrow \text{get} \? \text{what is the right ordering} \]

\[ \text{get} \rightarrow \text{add} \]