CSCI 1380: Day 12
Time + Global State
Lost Class

Real time is bad (not monotonic & hard to sync)
Systems want ordering that is monotonic

Ordering
- Total $\rightarrow$ logical clocks
- FIFO
- Causal $\rightarrow$ VC

Today

Global state
- Consistent snapshots
- Chandy Lamport

Replication (Passive/Active)
- Quorum & Consensus
- Raft = passive
Global state (global snapshot)

global state (distributed snapshot) = distributed backup of the cluster

Time-based snapshot

@ some time T (2pm) all servers make backup (snapshot)
Potential problems
① 2pm all servers stop running
② 2pm is different on every server

Manual snapshot

manually log into each server to create a snapshot
Problems
① takes too long
② inconsistency

Continuous snapshot

every N hours each server takes a snapshot
Don't pause during snapshot & don't try to coordinate

The trick is when you rollback (restore)

You look across servers for a consistent snapshot
① Pick a time that you want to roll back to
② figure all snapshots within that time
③ is latest snapshot from s; consistent with s2, if yes => done
ever go back a snapshot

Problems
① still have some notion of real time
② cascading rollback = the only consistent snapshot is @ T=0 when system starts

no guarantee to find a consistent snapshot
Consistent snapshots

only way for causality or to intertwine state on different servers is via msg

consistent = recv event also have then send event in the snapshot

Pick random snapshot (e₅, e²)

S₂ [6, 1] Versus [2, 0]

if pick eᵢ from Sᵢ

$V_{C_i} [i] > V_{C_j} [j]$

j = i

Detecting Inconsistent Snapshot

if an event at Sᵢ has a vector clock $V_{C_i}$

and an event at Sⱼ (Sᵢ ≠ Sⱼ) has a vector clock $V_{C_j}$

then the snapshot is inconsistent if $V_{C_i} [i] > V_{C_j} [j]$

Thus Sᵢ knows about more events of Sⱼ than Sⱼ does hence inconsistency
logical clocks don't capture causality

& can't help infer msg b/t two servers

so they can't help with consistent snapshot
1. Any node can initiate a snapshot
   * Sends a marker to all other nodes
   * Start watching for markers for all other nodes
   * Create a snapshot of state & maintain queues for each server

2. When you get your first marker
   * Checkpoint of state
   * Send a new to everyone but the person that sent you a marker
   * Maintain queues for every one but the person that sent you a marker

3. Each server considers the checkpoint complete after receiving markers for everyone else
Assumptions

① servers never crash (if they do the system stops)

② external entity starts the snapshot process

③ msg are deliver in order (FIFO order)

④ also msg don't get lost

out of order introduces inconsistency

if marker is lost then protocol never ends
Today

Global state (distribute checkpoint / snapshot)

* approaches for capturing global state
* determining consistent snapshot
  > using vector clock
* Chandy-Compos algorithm for distributed snapshot