CSCI 1380 : Day 13
Global State / Replication
Last Class

1. Various approaches for global snapshots
   a. time synchronized
   b. continuous (periodic)
   c. manual (instantaneous)
   d. Chandy Lampert algorithm
2. Using Vector Clocks to detect inconsistencies

This Class

1. Chandy Lampert analyzed
2. Passive / Active replication
   a. Linearizability
   b. Tradeoffs
   c. Overheads
   d. Implications of failures
Consistent snapshot

if the rev event is recorded then the send must also be recorded

\[
\text{consistent} = [s_a, s_x], [s_b, s_2], [s_b, s_x]
\]

\[
\text{not consistent} = [s_a, s_2]
\]
1. Snapshot is triggered manually.
   - One server who creates a snapshot & sends out markers & starts recording events.
2. Rules for processing a Marker from Sk if server hasn’t processed any markers
   - Then create checkpoint of state
   - Start recording events from other servers (\( \backslash S_k \))
   - Else stop recording events from \( S_k \) & place recorded events in checkpoint.
3. Marker sending rules after creating initial checkpoint
   - Send markers to all other servers before sending other message.
4. Algorithm end after all servers have received markers from every other server.
   - Each server sends "checkpoint" to the initiator.
5. Only one snapshot at a time.
1. Server failure
   if a server fail it will not
   process or send out markers
   Protocol may never end if all
   markers are not sent
   This is key because each server
   waits for markers from ALL other servers

2. Packet reordering
   network can reorder but the protocol
   assumes that reordering will never happen

When the marker
is proceed then
store queue in
snapshot & stop
using uses
Send A, Send B, Send Marker

Snapshot will think you have A/B

all 4 msgS

÷.;¥e

÷÷÷÷÷

÷,

÷¥msgs

÷÷÷÷÷

÷)

(snapshot d-

dot goes it indie in

sofa newly

t*

Eon's slot

snapshot with reordering the marker is delivered after X & not before X hence a huge problem)
1. Snapshot is triggered manually.
2. Rules for processing a Marker from Sk:
   - If server hasn’t processed any markers:
     - then create checkpoint of state
     - start recording events from other servers (\{ Sk \})
   - else:
     - stop recording events from Sk
     - place recorded events in checkpoint
3. Markers sending rules:
   - after creating initial checkpoint
   - Send markers to all other servers before sending other message
4. Algorithm end after all servers have received markers from every other server:
   - each server sends checkpoint to
   - the initiator

\[ S_i \]
Analyzing Chandy-Lamport algorithm

Termination

1. Each server must receive markers from all other servers.
2. Provided no PHT loss & all servers are operating correctly then termination is guaranteed.

Agreement/Correctness

1. Each distributed snapshot has both recv/sent event.
2. As long as there's no reorder.

Send events are captured before send markers.
Reorder capture in snapshot or in queue before you get a marker.
Replication (storage layer)

FE = Front End Servers
(Web/Application servers)

Replica = store a copy of data

Properties of replication layer

1. Transparency = client is never aware of all the copies of the data; the illusion of just one copy
(1) FE always interacts with leader RM

(2) Leader RM applies changes locally then replicates to followers

(3) After all (or majority) of followers acknowledge then leader responds to FE

(1) FE interacts with all RMs (each msg has an ID assigned)

(2) Each RM lets the other RMs know of events received

(3) Each RM maintains an order of event & processes the first event with acknowledge from all (events are processed in order of FE assigned ID)
1. Complex failure scenario when leader fails you need a new leader before progress.

2. Only contact leader can overwhelm.

3. RMs need to wait for responses.

2. Lots of msg (expensive).

3. Hard to maintain consistent view.
passive

leader

replicate result

followers

Active/Active

Active

passive

raft/zookeeper/etc

Node/network is slow = then replication is slow

failure = stall

Consensus (agreement) = All or Majority

can tolerate some failures/slow nodes

for Active Replication each RM only needs multicast from majority

for Passive the leader needs a majority of followers
Passive

1. Leader fails the protocol slows down
2. Fewermsg
3. The leader does a lot of work & followers do almost no work
4. Can have non-deterministic code

Active

1. Failure of small # has almost no impact as long as majority is not affected
2. Lots of msg
3. Every node does a lot of work
4. Must be deterministic

Choices of Passive v. Active, this based on performance during failures
Summary

Replication
Terminology (FEs/RMs)
Properties (Transparency)

Replication Strategies
* Passive
* Active
* Tradeoffs between Active & Passive