Distributed snapshots

Some terms

Often useful: states, executions, reachability
- A state is a global state \( S \) of the system: states at all nodes + channels
- An execution is a series of states \( S_i \) s.t. the system is allowed to transition from \( S_i \) to \( S_{i+1} \)
- A state \( S_i \) is reachable from \( S \) if, starting in \( S \), it's possible for the system to end up at \( S_i \)

Types of properties: stable properties, invariants
- A property \( P \) is stable if \( P(S) \to P(S_{i+1}) \)
- A property \( P \) is an invariant if it holds on all reachable states

Token conservation system

Invariant: token in at most one place
Stable property: no token

Why do we want snapshots?
- Checkpoint and restart
- Detect stable properties (e.g., deadlock)
- Distributed garbage collection
- Diagnostics (is invariant still true?)
Distributed snapshots

Record global state of the system
- Global state: state of every node, every channel
Challenges:
- Physical clocks have skew
- State can’t be an instantaneous global snapshot
- State must be consistent

Physical time algorithm

What if we could trust clocks?
Idea:
- Node: “hey, let’s take a snapshot @ noon”
- At noon, everyone records state
- How to handle channels?

Physical time algorithm

Channels:
- Timestamp all messages
- Receiver records channel state
- Channel state = messages received after noon but sent before noon
Example: is there <= 1 token in the system?

Physical time algorithm

11:59

Node 1
haveToken = true
Node 2
haveToken = false

token@11:59

Physical time algorithm

11:59

Node 1
haveToken = false
Node 2
haveToken = false

12:00

Node 1
haveToken = false
Node 2
haveToken = true

Snapshot:
- haveToken = false

Physical time algorithm

12:00

Node 1
haveToken = false
Node 2
haveToken = false

Snapshot:
- haveToken = false
Physical time algorithm

12:00

Node 1

haveToken = false

Node 2

haveToken = true

Snapshot: - haveToken = false

Snapshot: - haveToken = false

This seems like it works, right?
What could go wrong?

Physical time algorithm

11:59

Node 1

haveToken = true

Node 2

haveToken = false

11:58

Physical time algorithm

12:00

Node 1

haveToken = false

Node 2

haveToken = true

11:59

Physical time algorithm

12:00

Node 1

haveToken = false

Node 2

haveToken = false

11:59

Physical time algorithm

12:00

Node 1

haveToken = false

Node 2

haveToken = true

11:59
Physical time algorithm

12:01
Node 1 haveToken = false

Node 2 haveToken = true

Snapshot: - haveToken = true

Avoiding inconsistencies

As we've seen, physical clocks aren't accurate enough
Need to use messages to coordinate snapshot
=> make sure Node 2 takes snapshot before receiving any messages sent after Node 1 takes snapshot

Better algorithm

11:59
Node 1 haveToken = true

Node 2 haveToken = false

Better algorithm

12:00
Node 1 haveToken = true

Node 2 haveToken = false

Better algorithm

12:00
Node 1 haveToken = false

Node 2 haveToken = false

Better algorithm

11:59
Node 1 haveToken = false

Node 2 haveToken = true

Better algorithm

12:00
Node 1 haveToken = false

Node 2 haveToken = false

Better algorithm

12:00
Node 1 haveToken = false

Node 2 haveToken = false
Better algorithm

12:00
Node 1 haveToken = false
Node 2 haveToken = true
Snapshot:
- haveToken = true

11:59
Node 1 haveToken = false
Node 2 haveToken = true
Snapshot:
- haveToken = false

Chandy-Lamport Snapshots
At any time, a node can decide to snapshot
- Actually, multiple nodes can
That node:
- Records its current state
- Sends a “marker” message on all channels
When a node receives a marker, snapshot
- Record current state
- Send marker message on all channels
How to record channel state?

Channel state recorded by the receiver
Recorded when marker received on that channel
- Why do we know we’ll receive a marker on every channel?
When marker received on channel, record:
- Empty, if this is the first marker
- Messages received on channel since we snapped, otherwise

Chandy-Lamport Snapshots

B A M
D M C
E M F

Node 1 haveToken = true
Node 2 haveToken = false
What if multiple nodes initiate the snapshot?

- Follow same rules: send markers on all channels

Intuition:

- All initiators are concurrent
- Concurrent snapshots are ok, as long as we account for messages in flight
- If receive marker before initiating, must snapshot to be consistent with other nodes
**Chandy-Lamport Snapshots**

A cut is the set of events on each node in the system that are included in the snapshot.

A consistent cut is a cut that respects causality.

If an event is included by any node, all events that “happen before” the event are also included.

**Which state is snapshotted?**

Let’s say we have an execution $S_0, S_1, \ldots$

Some node starts the snapshot in $S_b$

The snapshot finishes in $S_e$

Which state did we snapshot?

**Which state is snapshotted?**

Node 1

counter = 4

Node 2

counter = 2

Which state is snapshotted?

Node 1

Counter = 4

Node 2

Counter = 2

Snapshot:

- counter = 4
Which state is snapshotted?

Node 1 snapshot Node 2

counter = 5 counter = 3

Snapshot:
- counter = 4

Which state is snapshotted?

Node 1 snapshot Node 2

counter = 5 counter = 3

Snapshot:
- counter = 4
- counter = 3

Which state is snapshotted?

What can we say about this snapshotted state?

Two things:
- Reachable from $S_b$
- Can reach $S_e$

Proof is in the paper
- Intuition: state is "consistent" with what actually happened

Stable Properties

Recall: a stable property is one that, once true, stays true

Snapshot represents a reachable state, but it may not represent any actual global state from $S_b$ to $S_e$.

However:
- If stable property is true in snapshot, we know it must still be true in $S_e$
- If stable property is false in snapshot, we know it must have been false in $S_b$