Distributed snapshots

Doug Woos
Logistics notes

Problem Set 1 due Friday

- questions on Piazza—private, please!
Today

Chandy-Lamport 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 $S_j$ is reachable from $S_i$ if, starting in $S_i$, it’s possible for the system to end up at $S_j$

Types of properties: stable properties, invariants

- A property $P$ is stable if

  $P(S_i) \rightarrow P(S_{i+1})$

- A property $P$ is an invariant if it holds on all reachable states
Token conservation system

Node 1 \(\rightarrow\) token \(\rightarrow\) Node 2

haveToken: bool

In \(S_o\)

- No messages
- Node 1 has haveToken = true
- Node 2 has haveToken = false

Nodes can send each other the token or “discard” the token
Token conservation system

Node 1 — token — Node 2

haveToken: bool haveToken: bool

Invariant: token in at most one place

Stable property: no token
Token conservation system

Node 1  token  Node 2

haveToken: bool

How can we check the invariant at runtime?

How can we check the stable property at runtime?
Distributed snapshots

Why do we want snapshots?

- Detect stable properties (deadlock)
- Phased computations (decentralized MapReduce)
- Diagnostics
- Invariant maintenance
- Other things?
Distributed snapshots

Problem: 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 snapshot, but
- 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 $\leq 1$ token in the system?
Physical time algorithm

11:59

Node 1

haveToken = true

Node 2

haveToken = false
Physical time algorithm

11:59

token@11:59

Node 1 \rightarrow Node 2

haveToken = false

haveToken = false
**Physical time algorithm**

12:00

![Diagram showing two nodes, Node 1 and Node 2, connected by an arrow labeled 'token@11:59'.](image)

- **Node 1**
  - haveToken = false
  - **Snapshot**: - haveToken = false

- **Node 2**
  - haveToken = false
  - **Snapshot**: - haveToken = false
Physical time algorithm

12:00

Node 1

haveToken = false

Snapshot:
- haveToken = false

Node 2

haveToken = true

Snapshot:
- haveToken = false
Physical time algorithm

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

Node 1

Node 2

11:59

haveToken = true

11:58

haveToken = false
Physical time algorithm

12:00  

Node 1  →  Node 2

haveToken = true  

11:59

haveToken = false

Snapshot:  
- haveToken = true
Physical time algorithm

12:00

Node 1

haveToken = false

Snapshot:
- haveToken = true

token@12:00

Node 2

11:59

haveToken = false
Physical time algorithm

12:00

Node 1

haveToken = false

Node 2

haveToken = true

Snapshot:
- haveToken = true
Physical time algorithm

Node 1
	haveToken = false

Node 2
	haveToken = true

12:00

Snapshot:
- haveToken = true

12:01

Snapshot:
- haveToken = true
Avoiding inconsistencies

As we’ve seen, clocks won’t help us

Need to use messages to coordinate snapshot

Idea: what if we could make sure Node 2 took a snapshot before receiving the token?
Better algorithm

Node 1

11:59

haveToken = true

Node 2

11:58

haveToken = false
Better algorithm

Node 1 | snapshot@12:00 | Node 2

12:00 | 11:59

haveToken = true

haveToken = false

Snapshot:
- haveToken = true
Better algorithm

Node 1

haveToken = false

Node 2

haveToken = false

12:00

11:59

token@12:00

snapshot@12:00

Snapshot:
- haveToken = true
Better algorithm

12:00

Node 1

haveToken = false

Snapshot:
- haveToken = true

Node 2

haveToken = false

Snapshot:
- haveToken = false

11:59
token@12:00
Better algorithm

12:00

Node 1

haveToken = false

Snapshot:
- haveToken = true

11:59

Node 2

haveToken = true

Snapshot:
- haveToken = false
Better algorithm

Node 1  →  Node 2

```
haveToken = false
Snapshot:
  - haveToken = true
```

```
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?
Chandy-Lamport Snapshots

Channel state recorded by the receiver
Recorded when marker received on that channel

When marker received on channel, record:
- Empty, if this is the first marker
- Messages received on channel since we snapshotted, otherwise
Chandy-Lamport Snapshots
Chandy-Lamport Snapshots

Node 1

haveToken = true

Node 2

haveToken = false
Chandy-Lamport Snapshots

Node 1
haveToken = false

Node 2
haveToken = false
Chandy-Lamport Snapshots

Node 1

haveToken = false

Node 2

haveToken = false

Snapshot:
- haveToken = false

marker

token
Chandy-Lamport Snapshots

Node 1

Node 2

haveToken = false

marker

token

Snapshot:
- haveToken = false

haveToken = false

Snapshot:
- haveToken = false
Chandy-Lamport Snapshots

Node 1

haveToken = false

Snapshot:
- haveToken = false

Node 2

haveToken = true

Snapshot:
- haveToken = false

In-flight:
- token
Chandy-Lamport Snapshots

Node 1
haveToken = false
Snapshot:
- haveToken = false

Snapshot:
- token

Node 2
haveToken = true
Snapshot:
- haveToken = false
Chandy-Lamport Snapshots

Multiple nodes can initiate the snapshot

- Follow same rules: send markers on all channels
- Intuition: either initiate and include in-flight messages, or don’t and exclude them
Chandy-Lamport Snapshots

D M C

B A M

E M F
Which state is snapshotted?

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

Node 2

snapshot

counter = 5

Snapshot:
- counter = 4

counter = 2
Which state is snapshotted?

Node 1: counter = 5

Node 2: counter = 3

Snapshot:
- counter = 4
Which state is snapshotted?

Node 1
- counter = 5
  Snapshot:
  - counter = 4

Node 2
- counter = 3
  Snapshot:
  - 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

So, why does this make sense for stable properties?
Discussion

What’s cool about this?
What’s the deal with phases?
What about dropped markers?