WTF are CRDTs?

Adventures in distributed systems theory

Disclaimer!

- Zero practical experience
- It looked cool
- So I decided to try to talk about it
- Also at YAPC
- (hopefully better by then)

Overview

- AP systems
- The trouble with ^ (hint: missing ‘C’)
- Dealing with data conflicts
- Conflict free/convergent data types
- Example: Observed Remove Set
- ???

What flavor of distsys?

- Masterless
- Eventually consistent
- AP-ish

AP …ish?

- Consistency, availability, partition tolerance…
- Pick 2
- Or 0.7/0.5/0.3

Cassandra

What flavor of distsys?

AP


THE
GOOD
THE
BAD
AND THE
WOOKIEE

AP: The Good

- A(vailability)
- P(artition tolerance)
- (potentially) Linear write scalability

AP: The Bad

- Eventually consistent (I just PUT it there, where'd it go?)
- Network overhead (maybe)

AP: The Wookiee

Data conflicts

Resolving data conflicts

- Causality tracking (logical clocks, version vectors/vector clocks, interval tree clocks)
- Last write wins (bet the farm on NTP)
- Give up and ask the client (siblings)

Aside: time, LWW, and the farm

- "All objects with a lower timestamp will be silently deleted until GC removes the tombstone record–which means that a rogue client or node can cause the destruction of every write to a record for *days to weeks afterwards*"
  - https://aphyr.com/posts/299-the-trouble-with-timestamps

Give up and ask the client

- Merge, then tell database the new state

- "In all such systems, we find developers spend a significant fraction of their time building extremely complex and error-prone mechanisms to cope with eventual consistency and handle data that may be out of date."
  - The GOOG (F1 paper)

Can’t someone do that for me?

Merging data is hard, let’s not do it

CRDTs

Commutative/convergent/conflict-free

Replicated

Data

Types

CRDTs

Commutative/convergent/conflict-free

Replicated

Data

Types

CRDTs

Commutative/convergent/conflict-free

Replicated

Data

Types

CRDTs

Commutative/convergent/conflict-free

Replicated

Data

Types

operation based: SBR

CRDTs

Commutative/convergent/conflict-free

Replicated

Data

Types

state based: RBR

(with a monotonic/idempotent merge)

Aside: WAT

Monotonic:

- "only moves in one direction"
- e.g. entirely increasing, or entirely decreasing

Idempotent

- "just run it again, don't worry"
- e.g. add an item to a set

CRDTs

Commutative/convergent/conflict-free

Replicated

Data

Types

the whole gang

(they're theoretically equivalent)

Counter

- Replica A says to Replica B:
  - ‘yo, increment!’ (or decrement)
- Trivially commutative operation
- …not all that interesting
- (or I lack imagination)

Non-negative counter?

- E.g. gold in the bank in a game
- …global invariant! Noooooo!
- Requires synchronization

Observed-Removed Set

- Set of insertions, each with unique tag
- Set of deletions, each with unique tag
- On conflict, add > delete
- Merge: union of insertion, union of tombstones

Observed-Removed Set

```
{
  'type': 'or-set',
  'e': [
    ['foo', [1]],
    ['bar', [1], [1]],
    ['baz', [1, 2], [2, 3]]
  ]
}
```

Observed-Removed Set

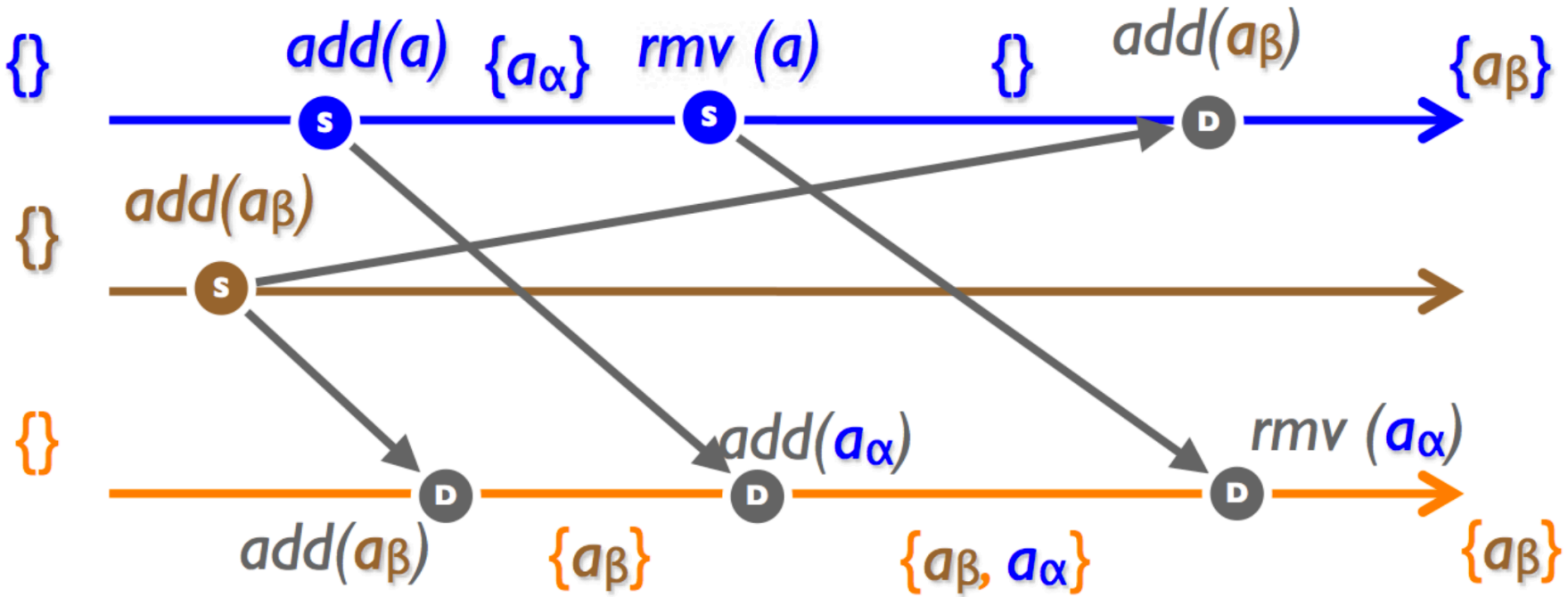
```
{
  'type': 'or-set',
  'e': [
    ['foo', [1]], // foo exists
    ['bar', [1], [1]],
    ['baz', [1, 2], [2, 3]]
  ]
}
```

Observed-Removed Set

```
{
  'type': 'or-set',
  'e': [
    ['foo', [1]],
    ['bar', [1], [1]], // no bar
    ['baz', [1, 2], [2, 3]]
  ]
}
```

Observed-Removed Set

```
{
  'type': 'or-set',
  'e': [
    ['foo', [1]],
    ['bar', [1], [1]],
    ['baz', [1, 2], [2, 3]]
  ] //baz exists
}
```


{}
add(a)
{aα}
rmv (a)
{}
add(aβ)
{aβ}
S
S
D
{}
add(aβ)
S
{}
add(aα)
rmv (aα)
D
D
D
add(aβ)
{aβ}
{aβ, aα}
{aβ}

Caveats

- Still eventual consistency (but stronger)
- Not all client operations will be respected; applications need to be aware.
- Garbage collection can be tricky (potential for unbounded garbage growth)

In the real world…

- Riak data types (Counters, Flags, Sets, Maps, etc.)
- Soundcloud stream: LWW element set
  - https://github.com/soundcloud/roshi
- League of Legends chat: friend list
  - Friends in general
  - Online/offline friends

Bonus! DAG

- Directed Acyclic Graph (e.g. tree)
- Global invariant??
- Locally enforced: only add edges in existing directions

The End