Feel the Rush: CRDTs in Riak

Zeeshan Lakhani

Software Engineer at Basho Technologies,Inc | Founder/Organizer Papers We Love
@zeeshanlakhani

5-22-2015 (LambdaConf)

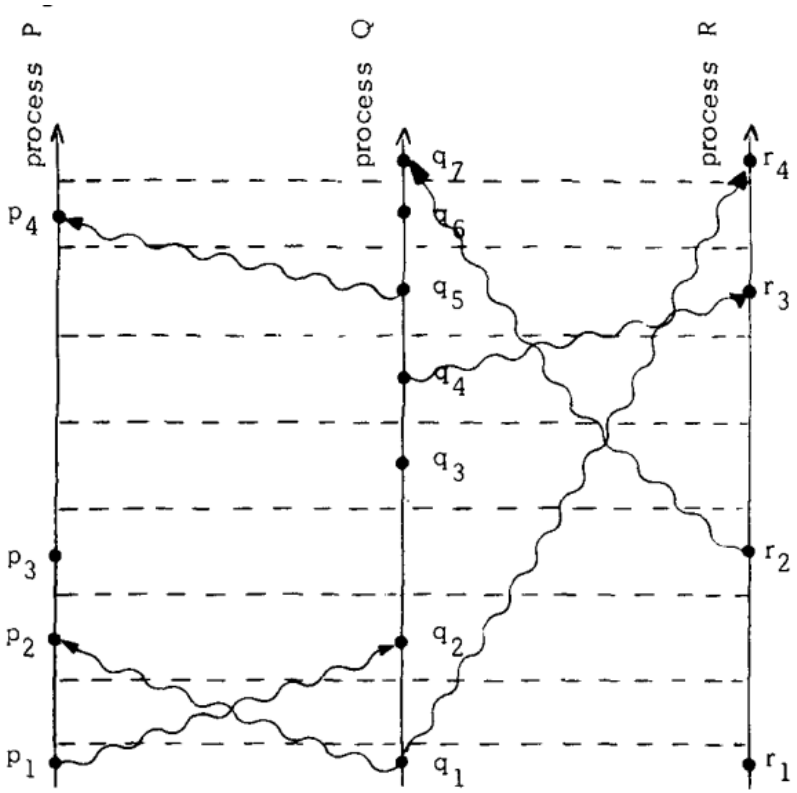
Partial Order of Events

- Time, Clocks, and the Ordering of Events never gets old
- Can only release say that something happened before[1]
- Events
  - q1 happens before p2 and before p3
  - p3 and q2 are concurrent - p3 || q2[2]
  - (figure on next slide)

[1] Time, Clocks, and the Ordering of Events in a Distributed System [Lamport] `http://bit.ly/1kQ3xhL`

[2] A Brief History of Time in Riak [Cribbs] `http://bit.ly/1Apto80`

Lamport's "space-time"/process diagram[3]



[3]Time, Clocks, and the Ordering of Events in a Distributed System [Lamport]
`http://bit.ly/1kQ3xhL`

Replication + Coordination (CRDTs)

- A kind of safety called Strong Eventual Consistency
- Data Structures reducing the need for coordination

State vs Operations-Based[6]

- CvRDT (convergent)
  - apply change locally; propagate entire state[4]
  - partial order to values
  - grows state monotonically
- CmRDT (commutative)
  - needs reliable broadcast to guarantee operations are delivered in partial order[5]
  - partial order to operations
  - replicas received updates

---

[4] Consistency and Riak [Meiklejohn] `http://bit.ly/1PBykxQ`

[5] CRDT Notes [pfraze] `http://bit.ly/1dpxMdn`

[6] A comprehensive study of Convergent and Commutative Replicated Data Types [Shapiro, et al.] `http://bit.ly/1PBC4zc`

"Hello World" of a distributed ordering/replication issue

- When two writes occur concurrently, the next read returns their union. Concurrent updates even on unrelated elements, a remove may be undone.[7]
- Using (semantic reconciliation) mechanism, an 'add to cart' operation is never lost. However, deleted items can resurface."[8]

[7] An Optimized Conflict-free Replicated Set [Bieniusa, et al.] `http://bit.ly/1ITUF48`

[8] Dynamo [DeCandia, et al.] `http://bit.ly/13QWj5Y`

Foundations - Lattice

A bounded join-semilattice[9] :

$$\langle S, \sqcup, \bot \rangle$$

smallest element in S:

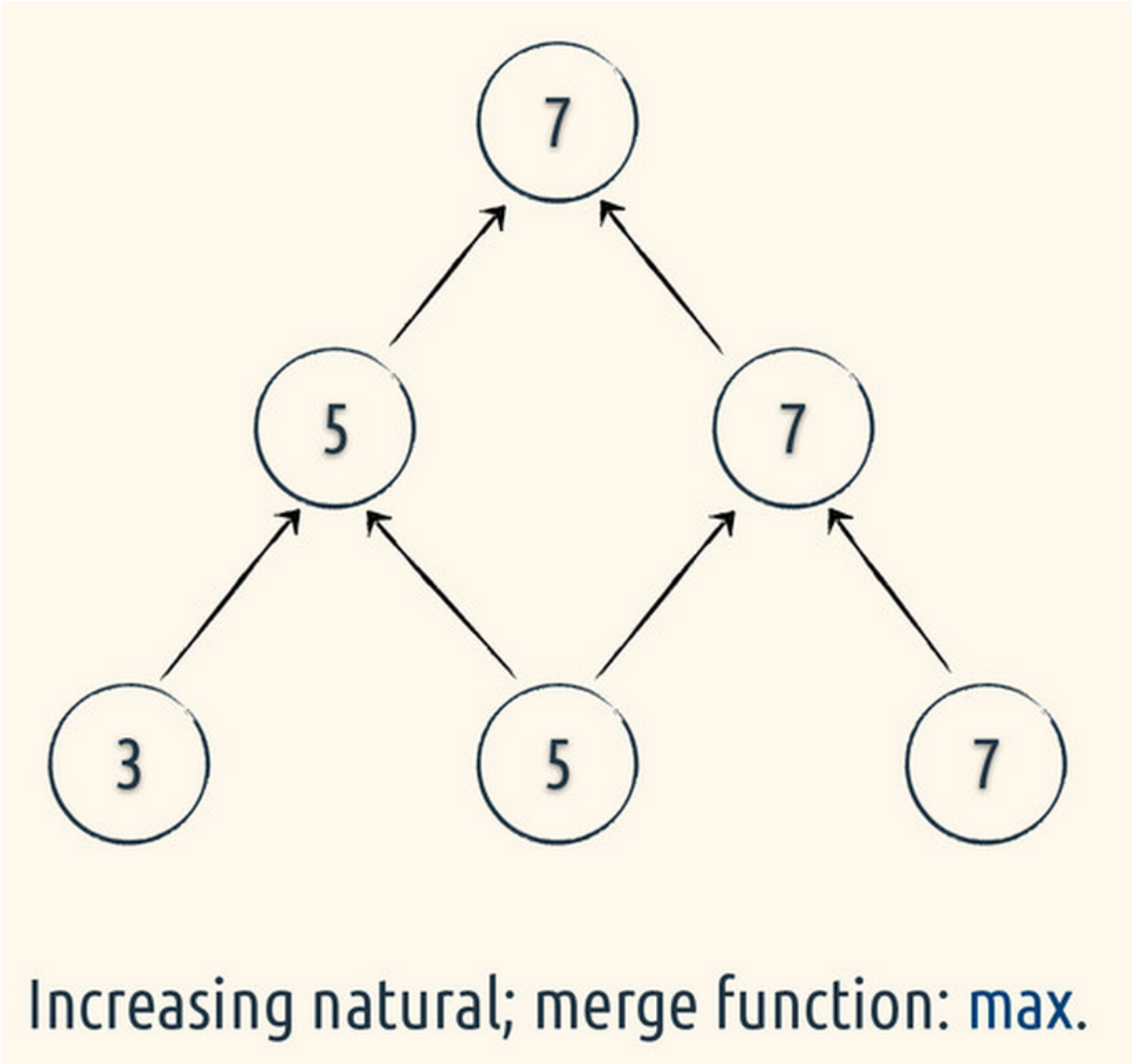
$$\bot$$

binary merge operator:

$$\sqcup$$

[9] Logic and Lattices for Distributed Programming [Conway, et al.]
`http://bit.ly/1IQ6ppV`

lmax[10, 11]



[10] Logic and Lattices for Distributed Programming [Conway, et al.]
http://bit.ly/1IQ6ppV

Foundations - LUB

- a merge function/operation that produces a Least Upper Bound (LUB) over a join-semilattice
  - Replica A has a Haskell book - insert into cart
  - Replica B has a Scheme book - insert into cart
  - LUB would be the smallest cart state that's greater than or equal to both elements in the ordering: {A, B}[12]
  - If LUB exists, then it's unique & conflict resolution is deterministic[12]

[12] What's the deal w/ lvars and crdts [Kuper] `http://bit.ly/1Kpz4Cq`

2P-Set Example[13]

```
{
  'type': '2p-set',
  'a': ['a', 'b'],
  'r': ['b']
}
```

- only a exists

[13] meangirls [Kingsbury] http://bit.ly/1cSrWQQ

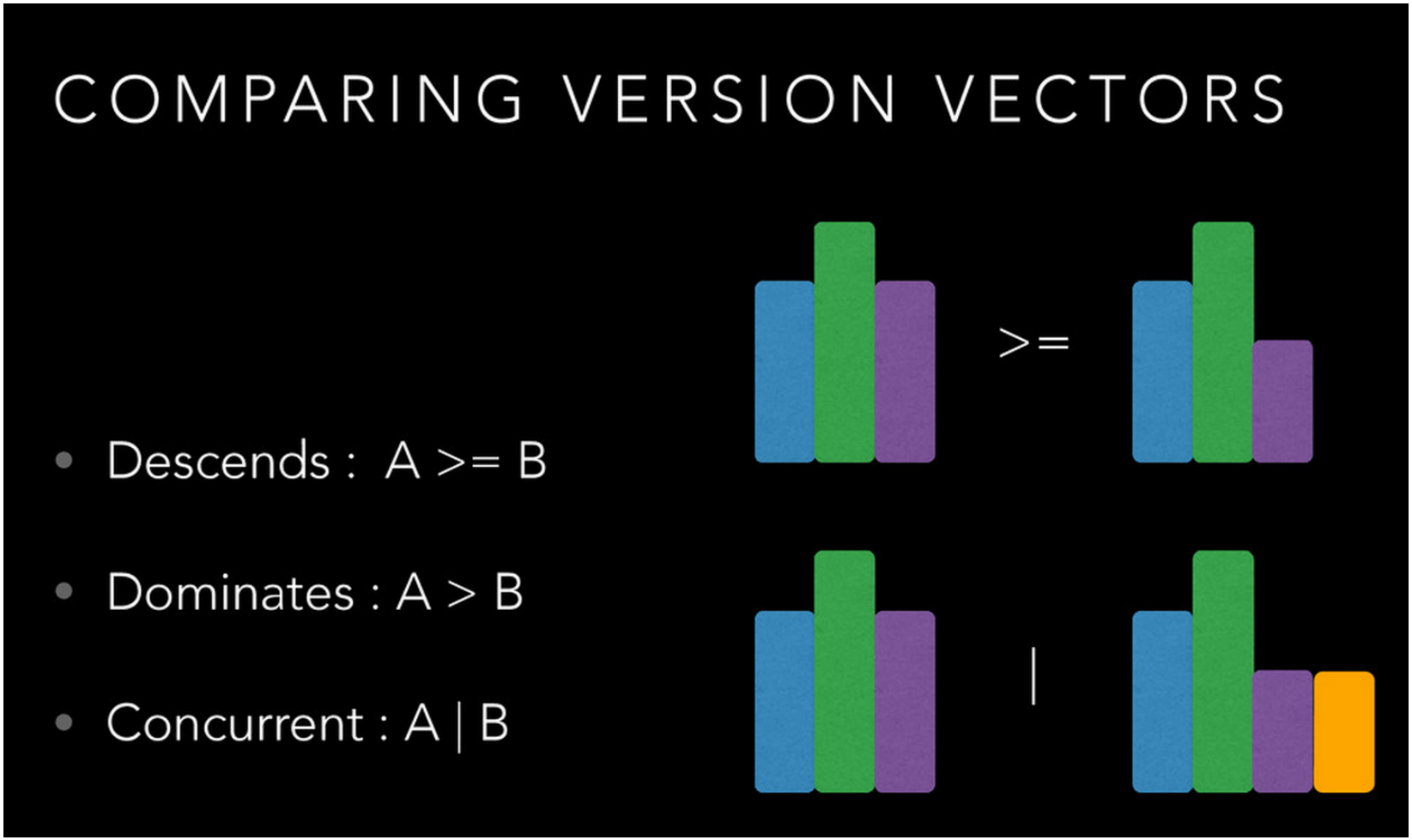
Regular Or-Set Example[14]

```
{
  'type': 'or-set',
  'e': [
    ['a', [1]],
    ['b', [1], [1]],
    ['c', [1, 2], [2, 3]]
  ]
}
```

- unique tags associted w/ each assertion
- a exists
- b != exist . . . insertion was deleted,
- c exists . . . two insertions, one deleted

[14] meangirls [Kingsbury] `http://bit.ly/1cSrWQQ`

Comparing Vector Versions - (VVs) track divergence



Descends, Dominates, Concurrent[15]

- descends A >= B
  - A summarizes at least the same history as B (as seen all the events in B)
- dominates A > B (good for discarding)
  - A is strictly greater than B (has seen all events B and at least 1 more event
- concurrent A || B
  - A contains at least 1 event unseen by B and B contains at least 1 event unseen by A

[15] A Brief History of Time in Riak [Cribbs] `http://bit.ly/1Apto80`

Riak's implementation - ORSWOT

- Optimized Observe-Remove Set[16, 17]
- Two-Way Comparison
- E.g. 2 Replicas Merge. . . one contains element in the set, other does not

[16] An Optimized Conflict-free Replicated Set [Bieniusa, et al.] `http://bit.ly/1ITUF48`

[17] riak_dt_orswot.erl [Russell Brown] `http://bit.ly/1ca8Wgb`

Causality to the Rescue[18]

Sets as (dotted) version vectors

[{actor1, count}=Dot1, {actor2, count}=Dot2,...] - minimal clock

We take all the elements that are in Set A and not in Set B and compare their minimal clocks to Set B's set version vector. Every element whose minimal clock is dominated has been removed from Set B, does not make it into the merged set. Also drop dots dominated by Set B's clock[a]

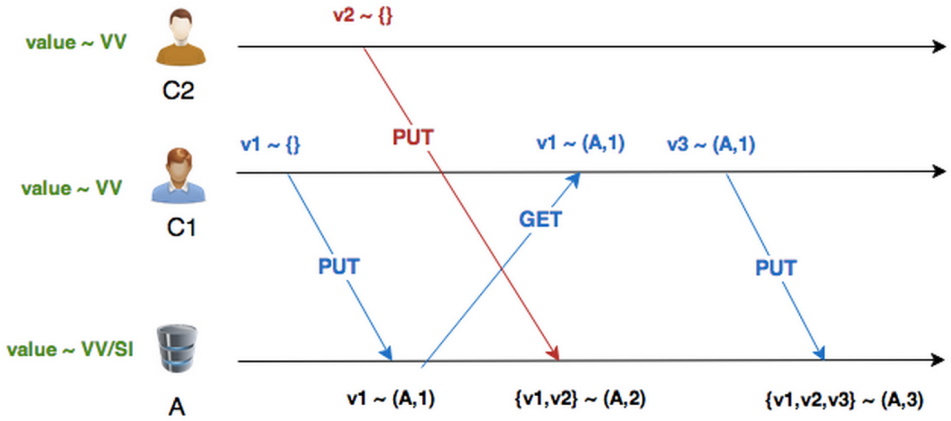
- Then repeat the other way (compare to Set A's VV)

[a] [Russell Brown] `http://bit.ly/1AptLzI`

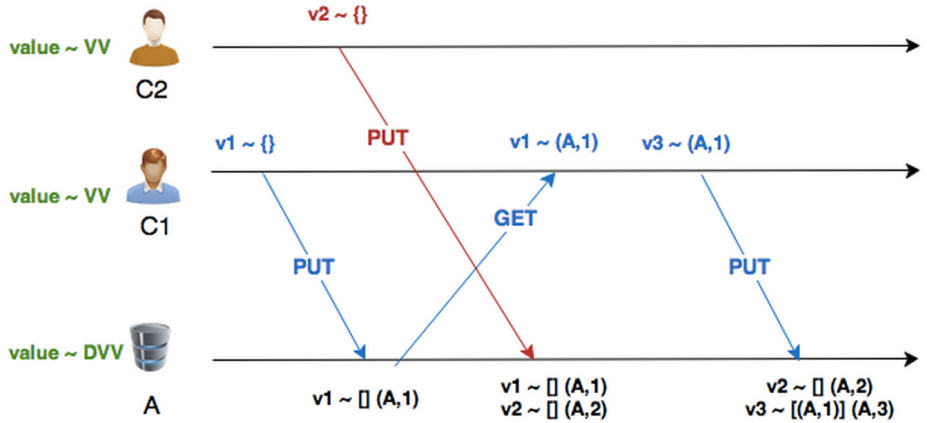
[18] [Russell Brown] `http://bit.ly/1AptLzI`

DVVs[19]

- VVs merge



- DVV



[19] Dotted Version Vector Sets [Gonçalves] `http://bit.ly/1Akedxs`

Riak Data Types[20, 21]

- Maps - riak_dt_map
- Sets - riak_dt_orswot
- Registers - riak_dt_lwwreg
- Flags - riak_dt_od_flag
- Counters - riak_dt_emcntr

[20] Riak DT [Russell Brown] `http://bit.ly/1Eoj0J4`
[21] Riak Data Types `http://bit.ly/1Q1S7RG`

Erlang client-side Example

22

```
Map4 = riakc_map:update({<<"interests">>, set},
          fun(S) ->
            riakc_set:add_element(<<"robots">>, S)
          end,
          Map3),
Map5 = riakc_map:update({<<"interests">>, set},
          fun(S) ->
            riakc_set:add_element(<<"opera">>, S)
          end,
          Map4).
```

[22] Riak Data Types Map `http://bit.ly/1BhfnEI`