Riak DT Map: A Composable, Convergent Replicated Dictionary

Russell Brown*, Sean Cribbs, Sam Elliot, Christopher Meiklejohn @basho.com

First Workshop on the Principles and Practice of Eventual Consistency April 13, 2014
Sorry

- Informal “experience report”
- Crap slides
Platform

- Riak
- Dynamo[1] Inspired KV Store
- Clustered and Replicated
- Eventually Consistent
Platform

- Consistent hashing[5]
- N replicas == Preference List
- R/W/PW/PR sloppy/strict quorums
Platform

- Version Vectors Per Key
- Partition is a Vnode is an Actor
- Opaque Values
- Syntactic Merge in Riak
- Semantic Merge at Client Application
History

- “Siblings” are a usability problem
- Application level semantic merges are “are ad-hoc and error-prone.” [Shapiro et al]
Problem?

• Focussed on a single use case
• Aimed for general usefulness
• Aware of CRDTs[2]
Use Case

- Mobile game progress data
- Game State
- Non trivial merge

Sample JSON

```json
{
"gold": 500,
"wood": 1250,
"stone": 100,
"buildings": [
"house",
"forge",
"farm"
]}
```
Desired Outcome

- Express updates as operations
- Apply related updates together
- Avoid “hand coded” resolution

**Conceptual**

**JSON Equivalent**

```
Build Tower ::
- subtract 250 gold
- subtract 500 wood
- subtract 100 stone
- add "tower" to buildings

{"gold_counter":
  {"decrement": 250},
"wood_counter":
  {"decrement": 500},
"stone_counter":
  {"decrement": 100},
"buildings_set":
  {"add": "tower"}}
```
Engineering Constraints

- Can’t build a new database from scratch
- Support Riak’s features
  - Anti Entropy
  - MDC Replication
-Hide CRDTs in Riak!
  - Riak Object, a sort of multi value register
How Hard Can It Be?

- Ignorance as an asset!
- Aim: Ad Hoc general composition of any CRDTs
- Result: Something less (but useful (we hope))
Structure

- ORSWOT[3] like
- DVV[4] like
- "Dot" is the actor, count pair for an event in version vector history
  - The event at time of creation
Structure

- `version_vector, [
  [{name, type}, [{dot, crdt}, ...}], ...], [deferred]`
Structure

- Version vector for Map
- Set of \{(name, type)\} pairs for field names map to
- Set of \{(dot, crdt)\} pairs
- List of \{(version_vector, [field_name])\} pairs for deferred operations
Structure

- e.g. \[
\begin{align*}
&\{[a, 3], [b, 2], [d, 1]\}, \\
&\{[\{“gold”, counter\}, [\{a, 1\}, \ldots]], \\
&\{[b, 1], \ldots]\}\], \\
&\{[\{“weapons”, set\}, [\{b, 2\}, \ldots]], \\
&\{[d, 1], \ldots]\}\], \\
&\{[\{[a, 5], [z, 25]\}, [\{“cars”, set\}]]
\end{align*}
\]

\[\text{Version vector} \quad \text{dots} \quad \text{CRDTs} \quad \text{Deferred}\]
Semantics

- Observed Remove (add wins)
- Updating embedded CRDT is a field Add
- Update merges set of dotted CRDTs and creates a new \{dot, crdt\}, discards merged.
Semantics

• Concurrent field remove + update: update persists

• Remove “dots” removed from remaining CRDT

• e.g. A with clock [{a,3}, {b,1}]
  removes {“weapons”, set} at {a,3} with
  with entries {a, 1} -> “sword”
      [{a, 2}, {b,1}] -> “dagger”
      {a, 3} -> “bazooka”
Semantics

- Merges with B which has clock [{b, 3}, {a, 2}]
  with entries {a, 1} -> “sword”
    [{a, 2}, {b,1}] -> “dagger”
    {b,2} -> “Halberd”
    {b,3} -> “ICBM”

- Result is [{b,3}, {a,3}] -> {“weapons”, set} -> {b, 3} -> [{b, 2} -> “Halberd”, {b,3} -> “ICBM”]
Semantics

- Shared causal context
- Map version vector is embedded set version vector
- Is embedded map version vector
- Dots are shared down through structure
Semantics

- “Future” Context for remove ops are “deferred”
  - Remove dots contained in context
  - Defer removal of unseen dots.
Merge

- Merge Map Version Vectors

- For each field, keep dots on both sides, drop dominated dots, keep frontier dots

- If a field is on one side, but not on the other, merge “empty” CRDT with absent side version vector (drops internal dots)

- Union deferred operations

- Apply deferred operations that are no longer from “the future”
Op Based Interface

- Send batch of operations
- Execute atomically at one replica
- Replicate new state to preflist
- Send map vv as context (for removes!)
Failure

• Not arbitrary composition

• Embedded ORSWOT /= ORSWOT

• Embedded Cntr /= PN-counter

• Ask me about the counter I made
Global Clock

- Map clock is the clock for all embedded CRDTs
- Dots must be passed down through structure
- e.g. embedded ORSWOT
Generality?

- Can't just "bang together" CRDTs
- Must share some causal relationship (?) verify (?)
Embedded ORSWOT - bad

- ORSWOT has a VV
- Update ORSWOT field at A
  - [{a, 1}] -> "sword"
- replicate to B
- remove field at A
- Update ORSWOT field
  - [{a, 1}] -> "bazooka"
- BUSTED!
Some mistakes I mad

- Merging Field CRDTs in Merge - need event
- Not dropping dominated dots
  - Can be in both sides, but removed after merge
- Not sharing causal context down the Map
Map “design” iterations

- OR-Set[2] of Fields
- Lattice of Types
What's in a name?

- {"gold", counter} -> counter
- {"gold", set} -> set
- Sam Elliot - intern extraordinaire
Next Iteration

- OR-Sets get BIG
- Update a CRDT means add the field (again!)
- ORSWOT[3] to the rescue
ORSWOT

• Didn’t understand the paper
• Didn’t grok CvRDT - CmRDT equivalence
• Learned about DVV[4]
• (re)Invented the ORSWOT!
[{a, 1}]

{a, 1}  Shelly
Shelly

{a, 1}  Shelly

{a, 1}, {b, 3}

{a, 1}

{b, 1}  Bob

{b, 2}  Phil

{b, 3}  Pete
\[\{\{a, 1\}, \{b, 3\}\}\]

- \(\{a, 1\}\) - Shelly
- \(\{b, 1\}\) - Bob
- \(\{b, 2\}\) - Phil
- \(\{b, 3\}\) - Pete

\[\{\{a, 1\}, \{b, 3\}\}\]

- \(\{a, 1\}\) - Shelly
- \(\{b, 1\}\) - Bob
- \(\{b, 2\}\) - Phil
- \(\{b, 3\}\) - Pete
One diagram contains the set \([\{a, 2\}, \{b, 3\}]\) with the following elements:
- \(\{a, 1\}\): Shelly
- \(\{b, 1\}\): Bob
- \(\{b, 3\}\): Pete
- \(\{a, 2\}\): Anna

The other diagram contains the set \([\{a, 1\}, \{b, 3\}]\) with:
- \(\{a, 1\}\): Shelly
- \(\{b, 1\}\): Bob
- \(\{b, 2\}\): Phil
- \(\{b, 3\}\): Pete
Flag (an aside)

- Observed Remove Boolean Flag
- Is a new thing?
Next Iteration

- ORSWOT like

- \{vv, \{{{\text{name}, \text{type}}, [{\text{dot}, \text{crdt}}]}}\}\}

- on remove, simply discard

- on merge, keep the un-dominated and common dots
On Going Work

- More compact representation
- More efficient merge implementation
- Replicate “fragments” not whole state
  - depend on Active Entropy more
Production Report

• Not in Production Yet :(  
• Basho’s difficult 2nd Album - Riak 2.0  
• Some ML posts - positive experiences  
• Some customer use cases
  • TapJoy - Maps of Counters for Stats
Testing

- Verification is hard
- No general tools for jobbing engineers
- Property based testing reasonable compromise
QuickCheck

• Model the system as set of actors
• Model the state
• Generate millions of interleavings of operations
  • create, add field, remove field, update field
QuickCheck Lessons

• Bugs slipped through
  • State space too large
• Smaller state space
• Focus on concurrent operations
Lets fight Talk

• Questions?


