Join Decompositions for Efficient Synchronization of CRDTs after a Network Partition

[Work in progress report]

Vitor Enes Carlos Baquero Paulo Sérgio Almeida Ali Shoker
HASLab/INESC TEC and Universidade do Minho

Abstract

State-based CRDTs allow updates on local replicas without remote synchronization. Once these updates are propagated, possible conflicts are resolved deterministically across all replicas. δ-CRDTs bring significant advantages in terms of the size of messages exchanged between replicas during normal operation. However, when a replica joins the system after a network partition, it needs to receive the updates it missed and propagate the ones performed locally. Current systems solve this by exchanging the full state bidirectionally or by storing additional metadata along the CRDT. We introduce the concept of join-decomposition for state-based CRDTs, a technique orthogonal and complementary to delta-mutation, and propose two synchronization methods that reduce the amount of information exchanged, with no need to modify current CRDT definitions.

1. Introduction

The concept of *Conflict-free Replicated Data Type* (CRDT) was introduced in (Shapiro et al. 2011) and presents two flavors of CRDTs: state-based and operation-based. A state-based CRDT can be defined as a triple $(S, \sqsubseteq, \sqcup)$ where S is a join-semilattice, $\sqsubseteq$ its partial order, and $\sqcup$ is a binary join operator that derives the least upper bound for every two elements of S.

With δ-CRDTs (Almeida et al. 2016), every time a replica performs an update, it will only send the information needed to reflect this update in other replicas, with the anti-entropy algorithm keeping at each node metadata tracking which deltas still need to be propagated to current peers. However, after a long partition, such metadata is discarded. In this situation, when a replica goes online again, the other remote replicas typically send their full state so this replica sees the updates it missed.

(Linde et al. 2016) introduces the concept of Δ-CRDTs where replicas exchange metadata used to calculate a Δ that reflects the missed updates. As this metadata is typically smaller than the full state, less is demanded from the network. In this approach CRDTs need to be extended to maintain the additional metadata for Δ derivation, and if this metadata needs to be garbage collected the mechanism will fall-back to standard full state transmission.

In this paper we will present a mechanism that does not add additional metadata to standard state-based CRDTs, but instead is able to decompose the state into smaller states than can be selected and grouped in a Δ for efficient transmission.

1.1 Problem Statement

Consider replica A with state a and replica B with state b, which at some point stop disseminating updates but keep updating their local state. When these replicas go online, what should replica A send to replica B so that B sees the updates performed on a since they stopped communicating? We could try to find c such that:

$$a = b \sqcup c$$

but if both replicas performed updates while they were offline, their states are concurrent, and there's no such c. (We say two states a and b are concurrent if a is not less than b and b is not less than a in the partial order: $a \parallel b \iff a \not\sqsubseteq b \land b \not\sqsubseteq a$.) The trick is how to find c (Δ from now on) which reflects the updates in the join of a and b still missing in b:

$$a \sqcup b = b \sqcup \Delta$$

The trivial example would be $\Delta = a$, but we would like to send less information than the full state. So, how can replica A calculate a smaller Δ to be sent to replica B, reflecting the missed updates?

1.2 Contributions

Firstly, we introduce the concept of join-decomposition for state-based CRDTs, a technique orthogonal and complementary to delta-mutation. Then, we propose two synchronization techniques. *State Driven:* replica B sends its full state b to replica A and replica A is able to derive Δ. *Digest Driven:* replica B sends some information about its state b, smaller than b itself, but enough to allow replica A to compute Δ.

2. Join Decompositions

We now explain how the concept of join-decomposition (Birkhoff 1937) can be applied to state-based CRDTs. Given state $r \in S$, we say that $D \in \mathcal{P}(S)$ is a join-decomposition of r if:

$$\bigsqcup D = r \tag{i}$$

$$\forall s \in D \cdot \bigsqcup (D \setminus \{s\}) \sqsubset r \tag{ii}$$

Property (i) states that the join of all elements in a join-decomposition of r should be r. Property (ii) says that each element in a join-decomposition is not redundant: joining the remaining elements is not enough to produce r.

PMLDC '16 July 17, 2016, Rome, Italy

ACM [to be supplied]. . . $15.00

We are interested in decompositions made up of "basic" irreducible elements. An element s is join-irreducible if it cannot result from a join of two elements other than itself, i.e.:

$$t \sqcup u = s \Rightarrow t = s \vee u = s$$

We say D is a join-irreducible decomposition if D is a join-decomposition and:

$$\forall s \in D \cdot s \text{ is join-irreducible} \tag{iii}$$

States in common CRDTs typically have join-irreducible decompositions, and we now present some examples of decomposition functions, which take a state and return a join-irreducible decomposition.

2.1 Example Decompositions

A $\mathsf{GCounter}$ is a simple replicated counter where its value can only increase (Almeida et al. 2016). It is represented as a map from ids to naturals, i.e., $\mathsf{GCounter} = \mathbb{I} \hookrightarrow \mathbb{N}$, and each replica can only increase the value of the counter in its position of the map. The value of the counter is the sum of all increments. For example, $p = \{A \mapsto 3, B \mapsto 5\}$ means replica A has incremented the counter three times, replica B five times, hence the value is eight. For each state s, a join-irreducible decomposition can be obtained by function:

$$\mathsf{D}^{\mathsf{GCounter}}(s) = \{\{i \mapsto v\} \mid (i, v) \in s\}$$

The decomposition for the $\mathsf{GCounter}$ p above would be $\{\{A \mapsto 3\}, \{B \mapsto 5\}\}$.

To allow both increments and decrements we can compose two $\mathsf{GCounter}$ by pairing them (Baquero et al. 2015) and we have a $\mathsf{PNCounter} = (\mathbb{I} \hookrightarrow \mathbb{N}) \times (\mathbb{I} \hookrightarrow \mathbb{N})$. Join-irreducible decompositions can be obtained through:

$$\begin{aligned}\mathsf{D}^{\mathsf{PNCounter}}((p, n)) = \{(\{i \mapsto v\}, \{\}) \mid (i, v) \in p\} \\ \cup \{(\{\}, \{i \mapsto v\}) \mid (i, v) \in n\}\end{aligned}$$

As a final example, an Add-Wins set has state $\mathsf{AWSet} = (E \hookrightarrow \mathcal{P}(D)) \times \mathcal{P}(D)$. This CRDT is a pair where the first component is a map (from element, in E, to a set of supporting *dots* (unique event identifiers), in $\mathcal{P}(D)$) and the second component is a causal context represented as a set of dots $\mathcal{P}(D)$ (Almeida et al. 2016). When an element is added to the set, a new entry in the map is created, if needed, mapping this element to a new dot, and current dots for the element, if any, are discarded. This new dot is also added to the causal context. To remove an element, we remove its entry from the map. An example for this data type where two elements (x and y) were added and another (initially marked with unique dot $a2$) was removed is $s = (\{x \mapsto \{a1\}, y \mapsto \{b1, c1\}\}, \{a1, a2, b1, c1\})$. (The *range* function rng returns all sets of supporting dots in the mapping.) The join-irreducible decomposition of state (m, c) can be obtained through function:

$$\begin{aligned}\mathsf{D}^{\mathsf{AWSet}}((m, c)) = \{(\{e \mapsto \{d\}\}, \{d\}) \mid (e, s) \in m, d \in s\} \\ \cup \{(\{\}, \{d\}) \mid d \in c \setminus \bigcup \mathsf{rng}\, m\}\end{aligned}$$

The join-irreducible decomposition for the state s above is:

$$\begin{aligned}\{(\{x \mapsto \{a1\}\}, \{a1\}), \\ (\{y \mapsto \{b1\}\}, \{b1\}), \\ (\{y \mapsto \{c1\}\}, \{c1\}), \\ (\{\}, \{a2\})\}\end{aligned}$$

3. Efficient Synchronization

State Driven The State Driven approach can be applied to all state-based CRDTs as long as we have a corresponding join-decomposition. We define $\mathsf{min}^{\Delta} : S \times S \to S$ as a function that given two states (the local state a and the remote replica state b) will produce a Δ. Join-irreducible decompositions will in general produce smaller Δs. Let $\mathsf{D} : S \to \mathcal{P}(S)$ be a function that produces a join-decomposition.

$$\mathsf{min}^{\Delta}(a, b) = \bigsqcup\{s \mid s \in \mathsf{D}(a) \wedge b \sqsubset b \sqcup s\}$$

This min^{Δ} function joins all s in the local state join-decomposition that strictly inflate the remote state. If the local replica ships the resulting Δ, to be joined to the remote replica, and joins the state received from the remote replica to its local state, both these replicas will reach convergence (if in the meantime no new update was performed).

Digest Driven With the Digest Driven approach we achieve the same results of State Driven but by exchanging less information. We re-define $\mathsf{min}^{\Delta} : S \times M \to S$ as a function that given the local state a and some digest m related to the remote state will produce a Δ.

$$\mathsf{min}^{\Delta}(a, m) = \bigsqcup\{s \mid s \in \mathsf{D}(a) \wedge \mathsf{inf}(s, m)\}$$

This digest will be data-type specific, which means that min^{Δ} will use a type-specific function $\mathsf{inf}(s, m)$ to check if s inflates the remote state summarized by the received digest m.

A digest extraction function $\mathsf{digest} : S \to M$ and the inflation test $\mathsf{inf} : S \times M \to \mathbb{B}$ for the causal AWSet CRDT can be defined as:

$$\mathsf{digest}^{\mathsf{AWSet}}((m, c)) = (\bigcup \mathsf{rng}\, m, c)$$

$$\mathsf{inf}^{\mathsf{AWSet}}((e, \{d\}), (a, c)) = \begin{cases} T & \text{if } d \notin c \vee (e = \{\} \wedge d \in a) \\ F & \text{otherwise} \end{cases}$$

The function $\mathsf{digest}^{\mathsf{AWSet}}$ returns a pair where the first component is the set of active dots (the supporting dots of elements that were added and not yet removed) and the second component is the full causal context. The inflation check $\mathsf{inf}^{\mathsf{AWSet}}$ will return T for $s \in \mathsf{D}(a)$ if the dot in s has not been seen in the other replica or s represents a removed element (i.e., $(\{\}, \{d\})$) that has been added and not yet removed in the other replica (d is still in the active dots).

If the Digest Driven technique is performed bidirectionally and no updates occurred, both replicas will converge (otherwise, they can still be collected separately in a dedicated buffer for further transmission).

References

P. S. Almeida, A. Shoker, and C. Baquero. Delta State Replicated Data Types. *CoRR*, abs/1603.01529, 2016. URL `http://arxiv.org/abs/1603.01529`.

C. Baquero, P. S. Almeida, A. Cunha, and C. Ferreira. Composition of State-based CRDTs. 2015.

G. Birkhoff. Rings of sets. *Duke Math. J.*, 3(3):443–454, 1937.

A. Linde, J. Leitão, and N. Preguiça. Δ-CRDTs: Making δ-CRDTs Delta-Based. *PaPoc 2016*, 2016.

M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski. Conflict-free Replicated Data Types. Technical Report RR-7687, July 2011. URL `http://hal.inria.fr/inria-00609399/en/`.