Efficient State-based CRDTs by Decomposition

[Work in progress report]

Paulo Sérgio Almeida
HASLab, INESC TEC &
Universidade do Minho
Braga, Portugal
psa@di.uminho.pt

Ali Shoker
HASLab, INESC TEC &
Universidade do Minho
Braga, Portugal
shokerali@di.uminho.pt

Carlos Baquero
HASLab, INESC TEC &
Universidade do Minho
Braga, Portugal
cbm@di.uminho.pt

ABSTRACT

Eventual consistency is a relaxed consistency model used in large-scale distributed systems that seek better availability when consistency can be delayed. CRDTs are distributed data types that make eventual consistency of a distributed object possible and non ad-hoc. Specifically, state-based CRDTs achieve this through shipping the entire replica state that is, eventually, merged to other replicas ensuring convergence. This imposes a large communication overhead when the replica size or the number of replicas gets larger. In this work, we introduce a decomposable version of state-based CRDTs, called *Delta State-based CRDTs* (δ-CRDT). A δ-CRDT is viewed as a join of multiple fine-grained CRDTs of the same type, called deltas (δ). The deltas are produced by applying δ-mutators, on a replica state, which are modified versions of the original CRDT mutators. This makes it possible to ship small deltas (or batches) instead of shipping the entire state. The challenges are to make the join of deltas equivalent to the join of the entire object in classical state-based CRDTs, and to find a way to derive the δ-mutators. We address this challenge in this work, and we explore the minimal requirements that a communication algorithm must offer according to the guarantees provided by the underlying messaging middleware.

1. INTRODUCTION

Eventual consistency [12] has recently got the attention of both research community and industry [5, 1, 11, 6] due to the enormous growth of large-scale distributed systems, and at the same time, the need to ensure availability for users despite outages and partitioning. In fact, the practical experience of leading industry shows that daily server outages and network partitioning in large-scale distributed systems is a norm rather than an exception. Given that partitioning cannot be avoided, the limitations explained by the CAP theorem [7] requires some sacrifice in consistency (by delaying it) for the sake of higher availability only when immediate consistency is not a requirement; a *like/unlike* action in social networks is a concrete example. CRDTs [9, 10] are formal methods to make eventual convergence of distributed datatypes generic and easy. Although they are currently being used in industry [5], CRDTs are still not mature, and many enhancements are still needed on both levels: design and performance. This work addresses some design issues to achieve better performance.

Conflict-free Replicated Data Types (CRDTs) [9, 10] are formalized data types designed to ensure the convergence of different replicas of a distributed CRDT object. Traditionally, two types of CRDTs were defined: *operation-based* and *state-based*. In operation-based CRDTs [8, 10], once an operation is invoked on a replica, a *prepare* phase returns a payload message that comprises a derived operation of the original one and possibly other meta-data. The message is sent to other replicas that apply this message via the *effect* phase which, in its turn, makes use of the received meta-data to maintain the causal order of operations. To achieve eventual consistency, this approach assumes a middleware that provides causal delivery of operations and membership management. In state-based CRDTs [2, 10], an invoked operation is applied on the local object state that derives a new state. Occasionally, the new state is sent to other replicas that incorporate the received state with the local state though a *merge*. A merge is designed in such a way to achieve convergence from any two states, being commutative, associative, and idempotent. In mathematical terms, merge is defined as a *join*: a least upper bound over a join-semilattice [2, 10].

State-based CRDTs are preferred to operation-based when causal delivery is not guaranteed by the messaging middlware. However, the state-based approach has two main weaknesses: (1) shipping updates becomes expensive when the distributed object gets large, and (2) a sort of garbage collection is often required. Some recent works [4, 3] addressed the problem of garbage collection; however, to the best of our knowledge, no profound research dealt with reducing the overhead of data shipping as we propose in this work.

The communication overhead of shipping the entire state in state-based CRDTs often grows with the replica state size and the number of replicas. For instance, the state size of a *counter* CRDT increases with the number of replicas, whereas, in a *grow-only Set*, the state size grows as more operations are invoked. Other CRDTs, like the *OR-Set*, impose a similar overhead also (due to shipping the set and its tombstones); although garbage collection can reduce this overhead once used, this is only possible when the invoked

PaPEC'14 April 13-16 2014, Amsterdam, Netherlands

http://dx.doi.org/10.1145/2596631.259663.

operations that cancel each others are close in time; e.g., an *add* followed by *remove* of the same element must occur before the shipping time is due. These scalability issues limit the use of state-based CRDTs to data-types with conservative payloads (e.g. few megabytes in Dynamo [6]). Recently, calls in the industry started to show up asking for the possibility to consider larger state sizes (e.g., in RIAK [5]).

In this work, we rethink the way that state-based CRDTs should be designed, having in mind the useless redundant shipping of the entire state. Our idea is to decompose a state-based CRDT in such a way to only ship recent updates rather than the whole state. To achieve this goal, we introduce *Delta State-based CRDTs* (δ-CRDT). A δ-CRDT is roughly a union of multiple fine-grained δ-CRDTs of the same type, which is built through multiple invocations of *δ-mutators* which are then merged. A δ-mutator is a derived version of a CRDT mutator that produces a δ which only comprises the new changes that the original mutator induced on the state. This way, we can retain the deltas, and join these deltas together into batches, to be shipped later instead of shipping the entire object. Once these batches of deltas arrive at the receiving replica, they are joined with its local state.

The challenge in our approach is to make sure that decomposing a CRDT into deltas and then joining them into another replica state (after shipping) produces the same effect as if the entire state had been shipped and merged. In particular, the challenge involves how to derive the δ-mutators from the original CRDT mutators.

In this work, we discuss these challenges, and explore possible solutions. In addition, we discuss the benefits of this approach given the guarantees provided by the messaging middleware, and we propose the basic requirements a distributed algorithm must satisfy towards this goal.

Acknowledgments.

Project Norte-01-0124-FEDER-000058 is co-financed by the North Portugal Regional Operational Program (ON.2 -- O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF). Funding from the European Union Seventh Framework Program (FP7/2007-2013) with grant agreement 609551, SyncFree project.

2. REFERENCES

[1] P. Bailis and A. Ghodsi. Eventual consistency today: Limitations, extensions, and beyond. *Queue*, 11(3):20:20–20:32, Mar. 2013.

[2] C. Baquero and F. Moura. Using structural characteristics for autonomous operation. *Operating Systems Review*, 33(4):90–96, 1999.

[3] A. Bieniusa, M. Zawirski, N. Preguiça, M. Shapiro, C. Baquero, V. Balegas, and S. Duarte. An optimized conflict-free replicated set. Rapp. Rech. RR-8083, Institut National de la Recherche en Informatique et Automatique (INRIA), Rocquencourt, France, Oct. 2012.

[4] S. Burckhardt, A. Gotsman, H. Yang, and M. Zawirski. Replicated data types: specification, verification, optimality. In S. Jagannathan and P. Sewell, editors, *POPL*, pages 271–284. ACM, 2014.

[5] S. Cribbs and R. Brown. Data structures in Riak. In *Riak Conference (RICON)*, San Francisco, CA, USA, oct 2012.

[6] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's highly available key-value store. In *Symp. on Op. Sys. Principles (SOSP)*, volume 41 of *Operating Systems Review*, pages 205–220, Stevenson, Washington, USA, Oct. 2007. Assoc. for Computing Machinery.

[7] S. Gilbert and N. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, 2002.

[8] M. Letia, N. Preguiça, and M. Shapiro. CRDTs: Consistency without concurrency control. Rapp. Rech. RR-6956, Institut National de la Recherche en Informatique et Automatique (INRIA), Rocquencourt, France, June 2009.

[9] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski. A comprehensive study of Convergent and Commutative Replicated Data Types. Rapp. Rech. 7506, Institut National de la Recherche en Informatique et Automatique (INRIA), Rocquencourt, France, Jan. 2011.

[10] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski. Conflict-free replicated data types. In X. Défago, F. Petit, and V. Villain, editors, *Int. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS)*, volume 6976 of *Lecture Notes in Comp. Sc.*, pages 386–400, Grenoble, France, Oct. 2011. Springer-Verlag.

[11] D. B. Terry, M. M. Theimer, K. Petersen, A. J. Demers, M. J. Spreitzer, and C. H. Hauser. Managing update conflicts in Bayou, a weakly connected replicated storage system. In *Symp. on Op. Sys. Principles (SOSP)*, pages 172–182, Copper Mountain, CO, USA, Dec. 1995. ACM SIGOPS, ACM Press.

[12] W. Vogels. Eventually consistent. *ACM Queue*, 6(6):14–19, Oct. 2008.