Correctness of a New Distributed Database Synthesis Protocol Using SMT Solvers

Kiarash Rahmani

Computer-Aided Program Reasoning Course Project - Dr. Samanta
rahmank@purdue.edu
https://github.com/Kiarahmani/CAPR2016

Abstract. As a result of the CAP theorem, today's distributed databases offer various levels of *weak consistency*. Application developers must pick the correct consistency level for each operation to preserve the correctness of their application. Recent works[1] successfully take this burden off the developer's shoulders. They correctly choose the weakest store offered consistency, that preserves application-level requirements. We are now planning to extend these works by actually **implementing** required consistency guarantees.

Keywords: Eventual Consistency, SMT solvers, Synthesis, Weak Consistency, Quelea, Cassandra

1 Proposal

Modern geo-replicated database systems, are widely used by major web services such as Google, Facebook and Amazon. They offer replication over intercontinent data centers that results in a high application throughput and partition tolerance. However, replication and reduced user-latency, come with a price: strong consistency cannot be achieved anymore[2]. As a result, application developers usually deal with a spectrum of weak consistency levels offered by the store and must pick the right one for each operation in their application.
This task is by no mean trivial for inexperienced coders and they usually end up choosing guarantees stronger than what they need (which will cause in reduced performance) or weaker ones (which will lead to inconsistent data and unacceptable behaviors).

Quelea[1] is a tool developed here at Purdue, that takes this burden off the shoulders of developers. They can express their application level requirements, and Quelea maps them to the *weakest, strong enough* store-offered consistency guarantee. The tool successfully implements widely used store-level consistencies, namely eventual, causal and strong consistencies, and each operation is executed in the right environment.

The main limitation with Quelea is the set of pre-defined, hard-coded store offered consistency levels. The application is analyzed statically and each operation is labeled with one consistency level: EC, CC and SC. Later, each operation is executed on a shim layer offered by quelea, that shows behavior defined by its own label.

We want to address this limitation by synthesizing a shim layer for each operation based on its needs. We have developed an algorithm that makes sure that a shim layer only shows behaviors acceptable by the operations requirements. In other words, instead of *picking* the right consistency level, we want to actually *implement* the exact level. Our algorithm statically analyzes the application and implements a shim for each operation.

Our algorithm must pass a number of safety and liveness features; the result shim layer must not show behavior that falsifies the operations consistency requirements. It must also present the maximal set of updates to each operation (we do not want the trivial extreme solution of not showing any updates to the current operation and thus preserving the consistency; we want to show as many as possible to make a sensible system). All these guarantees seem to be true at the moment but we want to formally prove them. Consequently, as the project for this course, I will be exploring different possibilities in model checking and SMT/SAT solvers to create a framework for correctness specification and proof. Ideally, I will be able to show the correctness of our algorithm and make use of the results from SMT solvers to fix the possible problems and/or improve its performance. we can compare results for different design options and pick the best one for the final implementation.

References

1. KC Sivaramakrishnan, G. Kaki, S. Jagannathan: Declarative Programming over Eventually Consistent Data Stores. PLDI '15.
2. S. Gilbert and N. Lynch: Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. SIGACT News, 33(2):51-59, June 2002.