Dynamic Consistency

João Neto
Universitat Politècnica de Catalunya
jlneto@ac.upc.edu

Abstract

Distributed systems often rely on replicated databases, with different data consistency guarantees for different operations, affecting their performance. The topic of consistency has been studied thoroughly but in a static manner, focusing on strict correctness invariants of the applications, and assuming resources are constant and plentiful. This thesis aims to investigate the impact of dynamically-adjusted consistency, based either on the available resources or user-specifiable system goals, on the performance of a replicated database. We expect that such an approach can reduce costs and performance penalties in resource starvation scenarios.

1. Introduction and Background

Many of today's information systems rely on distributed databases in order to achieve availability and scalability. Typically, these databases maintain multiple replicas of shared data. The replicas exchange data between them, guaranteeing clients can access the updated data from any of the replicas.

Historically, replicated databases have provided strong consistency, which makes them behave as if all operations were handled by a single node. However, this strong model requires a high degree of synchronization between the replicas, resulting in significant performance penalties and making the system unavailable under network partitions, as stated by the CAP theorem [3].

To tackle extreme load, modern databases may forego the high degree of replica synchronization that programmers are familiar with, by using Optimistic Replication [5]. In this replication model, each replica performs the operations requested by a client locally, immediately returning a success to the client. The effect of this operation is *eventually* propagated to the other replicas, without any strict time constraints. However, this may lead to *anomalies*: unwanted behavior caused by concurrent access that would not exist in strong consistency databases.

While applications can often tolerate such anomalies, violations of their *invariants* are not acceptable. Weak consistency models that completely forego synchronization are often too weak to enforce some invariants. To address these issues, a few databases provide hybrid consistency models, allowing for certain operations to be synchronized between replicas (red operations), while others are committed locally and scheduled for eventual propagation (blue operations). Recent work [4] addresses how to compute the optimal set of operations to be synchronized.

2. Smarter Update Dissemination

To the best of our knowledge, the dissemination of updates in hybrid-consistency databases (e.g. Antidote [1]) hasn't been studied extensively: state of the art considers that updates of a replica are propagated periodically to other replicas without any information on network topology or congestion, relying on a general-purpose consistency-oblivious message-passing overlay. The work has also always considered the deployments have plenty of resources to handle the workload.

The claim here is that if databases consider the available finite resources, they can adapt their behavior to make more efficient use of them.

3. Dynamic Consistency

Previous work has only considered the criteria of correctness when it comes to selecting the degree of synchronization of the operations. These criteria are based on the static needs of the application and therefore assign a static level of consistency to a given operation.

For example, in a weakly-consistent file system, such as Dropbox [2] or GeoFS [6], writing to a file may be considered a *blue* operation, although it may result in unwanted behavior: if two users concurrently write to the same file, both systems will create additional *conflicting copy* files as a result. Although both file systems don't necessarily violate their correctness, this can be seen as an *anomaly* and should be avoided for the sake of user experience. One idea is to prioritize the updates queued for dissemination that are more anomaly-prone in order to reduce their vulnerability window, or to adapt the propagation frequency between the replicas.

To do this, we could employ machine-learning to analyze the workload and when anomalies happen and prioritize updates accordingly to their probability of causing a anomaly if delayed.

The opposite idea applies for non-critical invariants: operations that may have initially been flagged as *red*, can probably be relaxed to *blue* under extreme system load without sacrificing system correctness in a very critical sense. Some

low-priority messages (e.g. activity log for a given user of a social network) may even be withheld indefinitely, until the network has enough bandwidth for it to be transmitted without affecting more time-sensitive updates.

Although we focused exclusively on how to reduce anomalies, the methodology can be customized to any set of goals. Other potentially interesting goals are to reduce bandwidth costs (e.g. 95th percentile), and update latency (either to expedite causally-dependable updates in order to reduce visibility latency for dependent ones, or to expedite updates that are more likely to be requested when a client does a consistent read from a snapshot).

4. Plans for Future Work

The plan is to develop a smart message queuing system for Antidote [1] that optimizes the behavior of inter-replica communication with respect to user-defined goals, taking into account available resources and workload prediction.

Currently, as a first step we are evaluating existing machine learning algorithms to tackle a simplified version of the problem, focusing only on reducing the number of anomalies. We then plan to integrate the system from the previous step into Antidote as an initial prototype. Ultimately we would like to test the system in both datacenter-centric environments, and as well as more resource-constrained environments such as community networks[7], in order to identify the strengths and weaknesses of our approach, as well as identify and quantify possible tradeoffs.

References

[1] Antidote. `https://github.com/SyncFree/antidote`.

[2] Dropbox. `https://www.dropbox.com`.

[3] 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. ISSN 0163-5700. . URL `http://doi.acm.org/10.1145/564585.564601`.

[4] A. Gotsman, H. Yang, C. Ferreira, M. Najafzadeh, and M. Shapiro. 'Cause I'm Strong Enough: Reasoning About Consistency Choices in Distributed Systems. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '16, pages 371–384, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3549-2. . URL `http://doi.acm.org/10.1145/2837614.2837625`.

[5] Y. Saito and M. Shapiro. Optimistic Replication. *ACM Comput. Surv.*, 37(1):42–81, Mar. 2005. ISSN 0360-0300. . URL `http://doi.acm.org/10.1145/1057977.1057980`.

[6] V. Tao, V. Rancurel, and J. a. Neto. A Name Is Not A Name: The Implementation Of A Cloud Storage System. In *Proceedings of the 6th Asia-Pacific Workshop on Systems*, APSys '15, pages 4:1–4:8, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3554-6. . URL `http://doi.acm.org/10.1145/2797022.2797034`.

[7] D. Vega, R. Baig, L. Cerdà-Alabern, E. Medina, R. Meseguer, and L. Navarro. A technological overview of the guifi.net community network. *Computer Networks*, 93:260 – 278, 2015. ISSN 1389-1286. . URL `http://www.sciencedirect.com/science/article/pii/S1389128615003436`. Community Networks.