Improving Scalability by Self-Archiving

Zhiwu Xie[1, 2], Jinyang Liu[3], Herbert Van de Sompel[4], Johann van Reenen[1], Ramiro Jordan[1]

[1]University of New Mexico
Albuquerque, NM 87131
{zxie, jreenen, rjordan}@unm.edu

[2]George Mason University
Fairfax, VA 22030
zxie2@gmu.edu

[3]Howard Hughes Medical Institute
Ashburn, VA 20147
liuj@janelia.hhmi.org

[4]Los Alamos National Laboratory
Los Alamos, NM 87544
herbertv@lanl.gov

Digital repository is but one form of data driven, dynamic web applications, therefore is not immune to the scalability problem. The increased usage, especially when coupled with personalization, context aware services, and social networking functionalities, brings in faster and more complicated repository state changes, which in turn aggravate distributing the dynamic contents in a timely and consistent manner.

In contrast, the motion picture takes on a much simpler but still practical distribution model. It only archives snapshots taken at the predefined time instants. State variations in between are presumed insignificant and unworthy of knowing, therefore are never distributed to the end users. Still, the dynamics are sufficiently preserved by replaying the temporalized snapshots according to their original timelines.

Inspired by this model, we proposed a novel web application framework that combines the server-side temporal database state archiving and the client-side database replication and querying.

We start by temporalizing the data model and require the database query issued explicitly with a time predicate, therefore is conceptually against a past database state. This forms a database query parallel to the Memento framework. We then limit the time predicates in these queries to a predefined set by establishing a TimeMap that all the web clients must obtain upon initialization and possibly update later. In this way, the same database snapshot can be reused by all the queries issued by different web clients that although not issued at the same instant, fall in the same time period.

We also take advantage of the recently popular client side database to replicate the archived database snapshot to the clients and then execute the timed queries on the clients. Since multiple clients would need the same database snapshot in approximately the same time period, its materialized view can be efficiently cached by the web intermediaries and reused, presenting a good temporal locality.

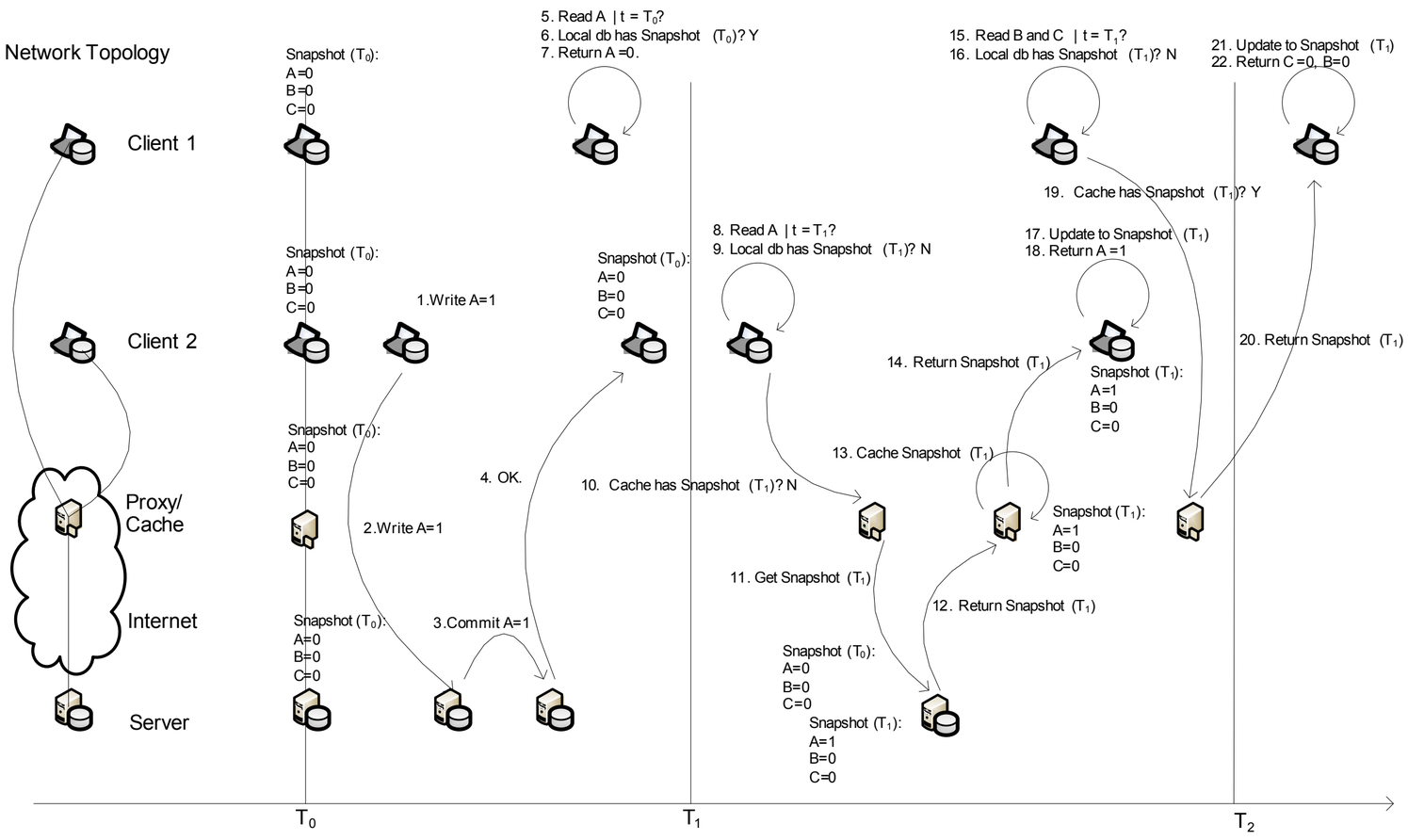

Figure 1. Improving scalability by self-archiving.

Unlike queries, database updates don't need time predicates, and are always executed on the server side database and timestamped with their respective commit times. This forms a lazy-master style database replication control algorithm. We can easily prove its correctness. The distributed execution of this algorithm is 1-copy serializable. It presents a single-node view equivalence to the web clients, an important correctness property not present in the popular "eventual consistency" model, widely used by the NoSQL style database . We leave the details and the proof to a separate paper.

In Figure 1 we schematically show how the server side temporal database, its snapshot archives, the web proxy/cache, and the clients and their local databases are concerted by our framework to read and write data over the time. We note a few interesting phenomenon. First, although step 3 commits before step 5, Client 1 is not aware of the updated value of A until after T1. Second, in step 13 Snapshot(T1) is brought in to the cache by Client 2, but is shared with subsequent queries from Client 1. Since the cache is closer to the clients than the origin server, Client 1 may get faster responses, potentially compensating the prolonged staleness.

In this poster we present an archive based web applications framework. It uses the client-side databases as the replicas of the master database on the server, ensures 1-copy serializability and improves scalability. We will report the experimental verification and quantitative measurements in the follow-up work.