Summary - Efficient Repairing and Measuring Replica Consistency in Distributed Databases

Bowen Song[1]

August 21, 2018

Abstract— **This report is a part of an independent study for set and string reconciliation problems in distributed systems. The paper [1] describes data consistency in a large-scale distributed peer-to-peer data storage system with multiple storage sites. By using set reconciliation primitives, the paper introduces an algorithm to measure and maintain inter-storage site data consistency. Due to the enormous scale of the system, the reconciliation process becomes asynchronous towards different parts of the database, thus, violating data's referential integrity. The algorithm, therefore, presents a metric for data *referential integrity* violations and a way to fix null referenced values.**

I. Introduction

The paper considers a distributed parallel database which requires content in all storage sites to be consistent replicas. Within each storage site, there are collections of databases. In each database, data are saved in multiple key-value tables that are connected by referencing keys. A key, if used as a reference in another table, is referred to as a foreign key, and it is unique within a storage site. The paper defines *Global Replica Consistency* as a metric for content consistency of all storage sites. The *Referential Integrity* indicates the existence of invalid foreign keys, of which may be a result of asynchronously modifying referencing and referenced tables. The proposed algorithm computes the aforementioned metrics based on symmetric differences between all pairs of storage sites and repairs all replica consistency under a small amount of communication cost.

A. Global Replica Consistency

The measurement of Global Replica Consistency, $gcur$, considers the consistency all of the database replicas. In a peer-to-peer distributed system setting, where the content of all nodes is treated equally, the $gcur$ is defined by Jaccard distance. The Jaccard distance for data on all hosts considers the collective intersections and unions according to $1 - \frac{[Intersection]}{[Union]}$. On the scale of 0 to 1, the closer the $gcur$ is to 0, the more consistent the replicas in the system.

II. Algorithm Overview

The algorithm considers measuring and repairing replica consistency and referential integrity for a distributed database.

[1]B. Song is with Department of Electrical and Computer Engineering, Boston University, Boston MA, sbowen@bu.edu

A. Replica Consistency

For replica consistency, the algorithm maintains a frequency table on each storage site for measurements, assuming data mutation is by insertion only. The replica consistency is achieved by broadcasting set reconciliation with CPI approach [2]. The broadcasting scheme of CPI approach requires an upper bound of symmetric differences on the reconciling sets. The algorithm treats all content of databases as sets of elements and uses the maximum number of symmetric differences between any two databases as the global upper bound.

The algorithm measures the global upper bound by treating one of the storage sites in the system as the master site. The algorithm uses the master site to reconcile with every other storage sites individually. Each reconciliation process may take several rounds to succeed [2]. Finally, the master site would be able to compute the union of all symmetrical differences between the master site and the rest of the system. The number of elements in the union is the global upper bound.

Due to the tremendous volume of each database, the storage sites may still undergo modifications during the reconciliation process. The frequency table is a summary of insertions that records the number of inserts of each key in the storage. If the reconciliation process fails for a storage site using the global upper bound, the master storage site can re-estimate the symmetric difference upper bound with that storage site by reconciling the frequency table, presumably with a lot smaller differences.

B. Referential Integrity

The referential integrity can be violated by asynchronously updating referencing and referenced tables, which is a typical case during reconciliation processes. Since, in most distributed databases, the primary network bandwidth is used to communicate with clients, the system often backup data under a limited communication budget; Therefore, not all data are backed up at the same time. In this case, a table with keys referencing values from another table might be pointing at values not backed up at the local storage site.

Within each storage site, the algorithm measures the referential integrity by creating a pair of referencing and referenced foreign key lists. The two lists contain all the foreign keys in the storage to make sure references are valid. The algorithm also considers the *frequency list* which keeps track of value updates of each key ensuring a key is referencing the correct value. Since the system assumes no deletion, a key's value on the frequency list should match the key's version or the number of values. The algorithm marks a referencing key an integrity violation if either of standards is not met. If a client requests for a locally null referenced key, the database would still want to supply the correct value. Therefore, on each reconciliation, the algorithm records the updating value's origin and respond to a local null reference inquiry with the referencing value form the origin site.

III. Algorithm Performance Analysis

The algorithm takes advantage of reduced communication cost of CPI approach to set reconciliation. The experimental evaluations demonstrate the computation and communication cost trade-off. However, the experiments are based on two clusters instead of multiple clusters to emulate multiple storage sites, which should have been the main point of the investigation.

The algorithm computes the maximum number of symmetric differences to use as a parameter to broadcast reconciling with other sites. However, with those efforts, the master site would have enough information to send other sites their missing data and save the computation cost while using the same amount of bandwidth utilization. Furthermore, the algorithm still relies on a master storage site to compute most of the calculations, which could disadvantage remote sites that suffer high-latency communication.

IV. Conclusion

This paper describes an efficient algorithm that maintains data consistency and reference integrity for a type of distributed parallel database. The algorithm performs best if the modification on each storage site is a lot smaller than the size of the entire data set, and it also prevents referencing incorrect or null data. For future reference, the algorithm may benefit from parallel reconciliation between multiple pairs of data storage sites. It may also benefit from enabling incremental set reconciliation.

References

[1] J. García-García, C. Ordonez, and P. T. Tosic, "Efficiently repairing and measuring replica consistency in distributed databases," *Distributed and Parallel Databases*, vol. 31, no. 3, pp. 377–411, 2013.

[2] Y. Minsky, A. Trachtenberg, and R. Zippel, "Set reconciliation with nearly optimal communication complexity," *IEEE Transactions on Information Theory*, vol. 49, no. 9, pp. 2213–2218, Sept 2003.