Summary - Synchronization and Deduplication in Coded Distributed Storage Networks

Bowen Song[1]
August 14, 2018

Abstract— **This report is a part of an independent study for set and string reconciliation problems in distributed systems. The paper [1] considers the problem of encoded data deduplication in distributed storage systems, where the system requires low internal communication cost and budgeted storage space. The system assumes all data modification details are known to the local host.**

I. Introduction

Encoding data in distributed storage systems (DSSs) targets reconstruction properties to achieve fault tolerance. Files are backed up in multiple storage nodes in shared spaces. For efficient use of space, the DSSs maintain data in both data storage nodes and parity storage nodes. Each data storage node contains a collection of coded data blocks which belongs to one user. The data in parity node comprises at least two merged users' data blocks as backup duplicates. The parity node merges data by adding multiple user data blocks together bit-wise in F_q, and it extracts any single user data block by subtracting bit-wise information from other user data blocks. Under this distributed structure, synchronizing encoded data in DSSs considers updating data for more than one storage node. The problem considers minimizing inter-nodes communication cost, which includes reducing total message size, communication rounds, and the number of affected nodes per update. The primary approach is to use an intermediary coding scheme to change the structure of the code on parity nodes per update while maintaining the property for using parity data to reconstruct files. The proposed schemes use Maximum Distance Separable (MDS) array codes as system structure and permutation and Vandermonde matrices to keep track of content structure.

II. Distributed Storage System for Coded Data

Structured by $[n, k; l]$ MDS array codes over Finite Field q [2], the DSS uses n number of storage nodes for every k number of users each with l length data blocks with $q \geq l$. As an example, a DSS maintained by [3,2;5] MDS array codes uses one data node per user and a parity node per two data nodes for bit-wise parity-check. In case one of the nodes fails, the system can self-repair by using information from two other nodes. In practice, systems with terabytes of data should choose larger parameters such as $[12, 9; 64000]$ MDS array codes to maintain millions of data blocks per node.

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

III. Algorithm Overview

To retain data consistency upon insertion and deletion edits, a DSS has to update more than one storage node to maintain the reconstruction property. As a base scheme, the DSS considers sending entire data block from one data node to every parity node that added this data node.

A. Using Permutation Matrix

To trade-off the extensive use of bandwidth in the base scheme, the Scheme P saves two types of matrices on parity nodes for each involving data nodes. After each synchronization process, the parity node records all latest operation positions and k individual $l \times l$ permutation matrices indicating data order for each data node. For each edition, instead of sending the entire block of data, Scheme P is only sending the edit position and the value to update its parity nodes. Unfortunately, the Scheme P cannot change block size for either deletion or insertion operations; it sends deleted values to the back of the block as zeros and insertions are only allowed in vacant positions within a block.

B. Using Vandermonde Matrix

To reduce storage requirement, the Scheme V stores invertible Vandermonde Matrices for each involving data nodes to allow dynamic block size. The position of last operation position is computable from the Vandermonde Matrices, and the scheme keeps the size of each matrix at $l' \times l'$ with l' changes for each operation. However, this scheme is only feasible for deletion edits. The Vandermonde matrices are encoding matrices for each parity node entry. For each deletion operation, the parity node deletes its corresponding row and column of encoding matrix and element from parity array. The dynamic size of data blocks and encoding matrices saves the redundant allocated space for each data blocks.

Fortunately, the DSS system-level data deduplication is an important and frequent operation [3], which tracks down and removes all but a fixed number of copies of the same data blocks within the system. For each deduplication process, the system could request a large number of deletions of symbols. In this case, the transmitting message can be the complement of deleted data.

IV. Algorithm Performance Analysis

The communication cost analysis considers the number of transmitted bits from a user to a parity node ignoring the number of parity and data nodes affected in each transmission. The base scheme requires users to transmit their entire data block which results in average communication cost of $O(l\ log(q))$, where $log(q)$ is the size of a symbol from F_q. At a cost of extra storage, Scheme P and V requires $O(log(l) + log(q))$ communication cost per edit where $O(log(l))$ is the size of index.

The necessary storage complexity for all schemes requires $l'\ log(q)$ per block since parity blocks are saved as the bit-wise sum in F_q of different users' data blocks, where l' is the size of data block after edition. The Scheme P, however, cannot change the value of l and, therefore, cannot reduce storage for deleting data from a block. Scheme V and P avoid saving any encoding matrices by spontaneously calculate them from recorded block editions and the unified description of the initial matrices. Therefore, the extra storage overhead for Scheme P and V to minimize communication cost only requires $log(l)$ per edit for storing the location of deletion.

V. Conclusion

The synchronization protocol while considers the general case of maintaining data consistency between storage nodes for distributed storage systems focuses on data deduplication and trade in storage overhead for reducing communication cost. The use of encoding matrices for tracking parity data modification reduced the size of the transmitted message for the synchronization and added d as a constant multiplicative factor to computation cost for recovering a file, where d is the total number of block editions in a parity node.

References

[1] S. El Rouayheb, S. Goparaju, H. M. Kiah, and O. Milenkovic, "Synchronization and deduplication in coded distributed storage networks," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 3056–3069, 2016.

[2] F. J. MacWilliams and N. J. A. Sloane, *The theory of error-correcting codes*. Elsevier, 1977.

[3] P. Strzelczak, E. Adamczyk, U. Herman-Izycka, J. Sakowicz, L. Slusarczyk, J. Wrona, and C. Dubnicki, "Concurrent deletion in a distributed content-addressable storage system with global deduplication." in *FAST*, 2013, pp. 161–174.