JCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Special Issue-10, May 2019 E-ISSN: 2347-2693

Bandwidth Saving Approach For De-duplication

Roshni Jaiswal[1*], Nagendra Kumar[2]

[1,2]Department of Computer Science & Engineering, SRIST, Jabalpur, India

[*]*Corresponding Author: roshanijaiswal9@gmail.com, Tel.: +91-6264202203*

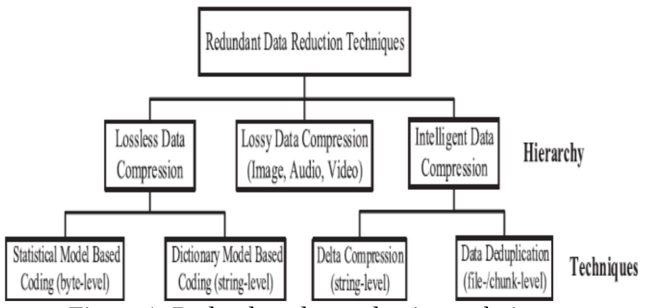
DOI: https://doi.org/10.26438/ijcse/v7si10.6871 | Available online at: www.ijcseonline.org

Abstract— The critical challenge of cloud storage or cloud computing is the organization of the continuously growing volume of data. Data de-duplication fundamentally submits to the exclusion of redundant data. However, indexing of all data is quiet maintained should that data ever be required. In general the data de-duplication eradicates the duplicate copies of duplicate data. Data De-duplication mechanism attain popularity from the academics and industrial as well, because the storage utilization in cloud storage is more efficient to store the data with the help of the cloud service providers. In this method redundant data is replaced with a pointer to the unique data copy. This reduces the hardware used to store data and the bandwidth costs required for transmitting and receiving purposes. This paper represents study of de-duplication as well as proposes a method for saving bandwidth and storage.

Keywords—Cloud computing, de-duplication, saving storage, bandwidth reduction

I. INTRODUCTION

The use of cloud for storing data by companies for backup and common people for sharing information among friends has increased drastically over the past few years. This has created a challenge to the cloud service providers to maintain all this massive data and to offer these services at lower price to the customers. In reality most of the data stored in the servers is often repeated. For example, a service may contain several instances of same data file, storing all these instances would require a large amount of storage space. This problem can be solved by using Data De-duplication technique [1]. Data de-duplication stores only one unique instance of the data type on the disk or tape. In this method redundant data is replaced with a pointer to the unique data copy. This reduces the hardware used to store data and the bandwidth costs required for transmitting and receiving purposes. De-duplication belongs to intelligent data compression technique for redundant data reduction [3].


Redundant Data Reduction Techniques
Lossless Data Compression
Lossy Data Compression (Image, Audio, Video)
Intelligent Data Compression
Hierarchy
Statistical Model Based Coding (byte-level)
Dictionary Model Based Coding (string-level)
Delta Compression (string-level)
Data Deduplication (file-/chunk-level)
Techniques
Figure 1: Redundant data reduction techniques.

But there are many challenges to be addressed when we are implementing data de-duplication. When we are uploading the data to a remote storage like cloud storage, preserving the confidentiality of the data is also important [2]. But data de-duplication is not compatible with the conventional encryption mechanisms that we use to convert the data into a secret form. In conventional encryption, the user first will encrypt the data with a key, that he chooses, and sends the cipher text to the cloud storage, but if several users try to upload the data using the conventional encryption mechanism, there will be so many different cipher texts uploaded to the cloud storage. so cloud storage provider will not able to identify whether two copies uploaded belonging to the same copy of the plain text or not. So if users are interested in making the data confidential, they have to scarify storage efficiency. If storage efficiency is to be given importance, users have to scarify on confidentiality of the data. So to ensure confidentiality of the data, while avoiding the redundancy of the data, i.e. maintaining the storage efficiency, a mechanism called convergent key encryption has been introduced. Convergent key encryption is a process of encrypting the data with a key generated by applying the data to a hash function; the resulting hash code will be used as the key to encrypt the data. The user will keep the hash code called convergent key with him and uploads the resulting cipher text to the cloud storage. Cloud storage provider will maintain the copies of the cipher text and a tag derived from the cipher text. The user who is trying to upload a file to the cloud storage will send the tag to the cloud storage provider. Cloud storage service provider will compare the tag with the tags available on the server. If there is a tag match then it is assumed that same copy of the file is already available on the cloud storage server and the user need not upload the file to the cloud storage server. The user will be added to the list of owners for that particular file.

II. RELATED WORK

Figure 2 shows an iconic visualization of data de-duplication, in which multiple files are fed into the de-duplication system. Each of them consists of a series of coloured data blocks. In data de-duplication, these data blocks are called "chunks" by convention. Chunks of the same colour have the same content. The data de-duplication system processes the files so that only one chunk of each colour is emitted. One important application domain for data de-duplication is backup storage. The prevailing storage media for backup systems has been tape. Over time, disk storage became less costly, but still tape storage has been considered cheaper. Disk-2-disk (D2D) backup became more cost-effective by using data de-duplication techniques. Data de-duplication reduces the storage requirements by a factor of around 10 and more, as large parts of a backup is identical to the previous backup [4]. Therefore, backup data has an exceptionally high data redundancy, which can be exploited by data de-duplication.

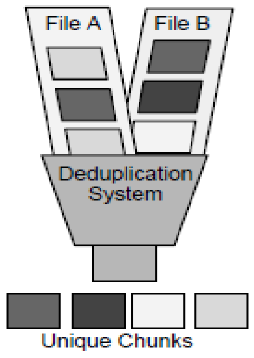


Figure 2: Iconic visualization of data de-duplication[9]

Besides its commercial success, data de-duplication has also been a hot topic in the research community. Most storage conferences in the last years have featured de-duplication sessions. In the recent years, researchers focused on different aspects of data de-duplication:

Throughput: A major research focus is approaches to achieve a high write throughput.
Especially the problem of the "chunk lookup disk bottleneck" proved to be important with Zhu et al.'s work an early breakthrough [5].

Clustering: Researchers are working on improving the scalability by using a cluster of cooperating nodes for data de-duplication [6] [7] to overcome the throughput and fault tolerance limitations of single-node systems.

Other types of storage workload: Beyond the usage of data de-duplication in backup and archival storage systems, researchers explore how to use data de-duplication for other types of storage workloads, e.g., primary storage and storage and migration of virtual machine images.

Advanced chunking: The primary methods for splitting blocks, files, or a data stream into smaller chunks, which are then used as the unit of redundancy detection, are static chunking and content-defined chunking. Researchers work on advanced chunking approaches to improve on these primary methods [8][9].

Haonan Su et al [11] proposes an efficient and secure data de-duplication scheme, which allows cloud storage servers to perform de-duplication based on fingerprints of data blocks before these blocks are encrypted by users. This can significantly improve the computation and communication efficiency considering the case of many duplicated data. File-level de-duplication is the easiest but inefficient method. Variable-size block-level de-duplication is difficult to deal with the situation of inserting data in the file.

P. Minisha [10] proposes a performance analysis of Rabin based chunking and Rapid Asymmetric Maximum (RAM) chunking using throughput. The Chunk is a method of breaking data into multiple pieces and each chunk has the unique hash identifier for identification. To check data duplication the hash identifier of a chunk is checked with the previously stored chunk. It increases the efficiency of cloud storage. There are two limitations with this approach. First, there are more number of strings and less number of hash values, so some different strings may have the same hash value. If these hash values match, the pattern and the substring may not match and vice-versa. Consequently, it reduces the throughput of data de-duplication.

Huijun Wu[12] proposed a sampling-based chunking method and develops a tool named SmartChunker to estimate the optimal chunking configuration for de-duplication systems. Our evaluations on real-world datasets demonstrate the efficacy and efficiency of SmartChunker. SmartChunker only needed to use a small number of samples to learn the potential storage capacity saving under different chunk configurations. However, the individual estimations on different chunk sizes do not work well. The key reason is that not all chunk sizes are equal in the estimation.

III. PROPOSED METHODOLOGY

A two layer approach has been proposed for capturing redundancy. Proposed system architecture is shown in figure 1. For any outbound traffic from the server, system first detects redundancy by the first layer module. If no redundancy is found, it turns to the second-layer module to search for short-term redundancy at finer granularity. The first-layer detects redundancy by a prediction-based matching approach. In the second layer, both the server and client maintain a temporary small local chunk cache to store most recently transmitted and received chunks during their communication respectively.

Once a client receives a data chunk that already exists in its local cache, it is expected that the future incoming data would also be duplicate. Thus, the client predicts the future coming data chunks and notifies the cloud server with the signatures of the predicted chunks. The server compares the predicted signatures received from the client with the signatures of outgoing chunks and confirms the correctly predicted chunks, which then do not need to be transferred without maintaining the client's status at the server.

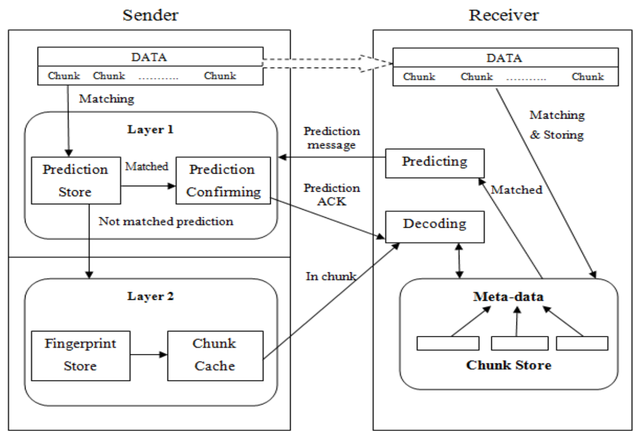


Figure 3: Proposed System

The sender divides the outgoing data into chunks in the same way as the receiver, and compares the signatures of outgoing chunks with all chunk predictions recently received from the receiver regardless of their expected positions in the data stream. For each chunk, the sender computes its signature (e.g., its SHA-1 hash value) and looks up the signature in the prediction store that keeps all predictions recently received from the receiver. If a matching signature is found, the sender sends a prediction confirmation acknowledgement message to the receiver instead of the outgoing chunk, no matter whether the chunk has the expected offset in data stream which is specified in the prediction.

In the second-layer, both the server and client maintain a temporary small local chunk cache to store most recently transmitted and received chunks during their communication respectively. In particular, the sender stores its recently transferred chunks in its local chunk cache. For every chunk in the cache, the sender computes a set of representative fingerprints, each of which is the hash value of a data window of size w in the chunk. Every representative fingerprint (along with a pointer to the corresponding chunk in the cache) is stored into a fingerprint store. To detect the redundancy inside an outgoing chunk, the sender performs match to identify maximal sub-strings in the chunk which have duplicates in the chunk cache. Specifically, the sender compares each representative fingerprint of the outgoing chunk against the fingerprint store to check whether a matching fingerprint exists. A matching fingerprint indicates that the outgoing chunk has a data window of size w which also appears in a previously transmitted chunk in the cache. If a matching fingerprint is found in the fingerprint store, the in-cache chunk which it points to is retrieved. The data window corresponding to the matching fingerprint in the in-cache chunk is expanded byte-by-byte in both directions and compared with the outgoing chunk, in order to identify the maximal overlap substring between the in-cache chunk and the outgoing chunk. Then, the sender encodes the maximal matched substring in the outgoing chunk with an in-chunk shim which contains the signature of the corresponding in-cache chunk, the offset and length of the matched substring.

For any incoming packet, the receiver first decodes in-chunk shims if any. To decode an in-chunk shim, the receiver retrieves the chunk in its local cache which has the same signature as that in the shim, and replaces the shim by the substring of the in cache chunk according to the offset and length specified by the shim. If the packet is an ACK message, the receiver checks the confirmed prediction in its prediction store and retrieves the corresponding predicted chunk in its chunk store.

IV. RESULTS AND DISCUSSION

Implementation of proposed system consist of three modules- Admin, Server and User. GUI of all these modules are shown below:

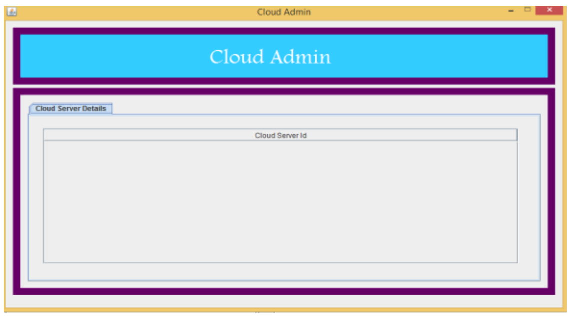


Figure 4: Admin GUI

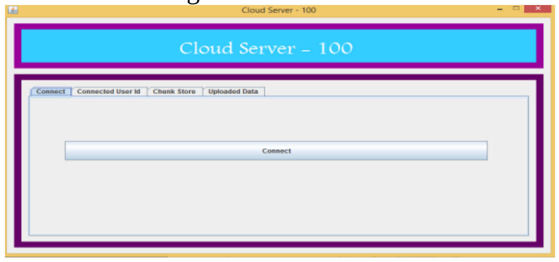


Figure 5: Server GUI

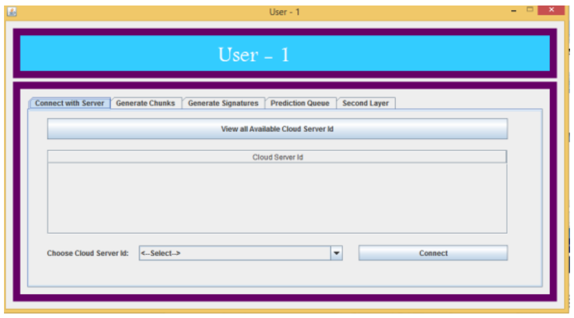


Figure 6: User GUI

At the receiver, the incoming data stream is divided into chunks. The chunks are linked in sequence, which forms a chain, and stored into a local chunk store. The receiver compares each incoming chunk to the chunk store. Once finding a matching chunk on a chain, it retrieves a number of subsequent chunks along the chain as the predicted chunks in the future incoming data. The signatures of the retrieved chunks and their expected offsets in the incoming data stream are sent in a PRED message to the sender as a prediction for the sender's subsequent outgoing data. To match data with a prediction, the sender computes SHA-1 over the outgoing data at the expected offset with the length given by the prediction, and compares the result with the signature in the prediction. Upon a signature match, the sender sends a PRED-ACK message to the receiver to tell it to copy the matched data from its local storage.
This implementation avoids the additional computational and storage costs.

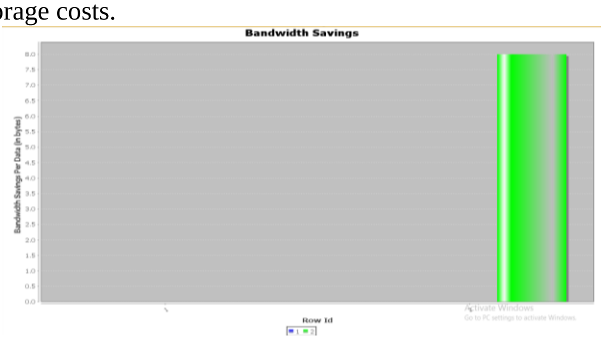


Figure 7: Saving of bandwidth

V. Conclusion

The pay-as-you-go service model impels cloud customers to reduce the usage cost of bandwidth. This technique is used to improve storage utilization and can also be applied to network data transfers to reduce the number of bytes that must be sent. Redundancy elimination has been shown to be an effective solution for reducing bandwidth costs, and thus has recently captured significant attention in the cloud environment. The work proposes a solution which can detect and remove redundancy through a two-layer design with cooperative operations between layers.

References

[1] Yukun Zhou, Dan Feng, Wen Xia, Min Fu, Fangting Huang, Yucheng Zhang, Chunguang Li,"SecDep: A User-Aware Efficient Fine-Grained Secure Deduplication Scheme with Multi-Level Key Management", IEEE Mass Storage Systems and Technologies (MSST) 2015 31st Symposium, Year – 2013.

[2] B.Tirapathi Reddy, U.Ramya, Dr.M.V.P.Chandra Sekhara Rao, "A comparative study on data deduplication techniques in cloud storage", IJPT| Sep-2016 | Vol. 8 | Issue No.3 | 18521-18530.

[3] Wen Xia, Hong Jiang, Dan Fen, "A Comprehensive Study of the Past, Present, and Future of Data De-duplication", Vol. 104, No. 9, IEEE 2016

[4] Heidi Biggar. Experiencing data de-duplication: Improving efficiency and reducing capacity requirements. White paper February, The Enterprise Strategy Group, 2007

[5] Heidi Biggar. Experiencing data de-duplication: Improving efficiency and reducing capacity requirements. White paper February, The Enterprise Strategy Group, 2007.

[6] Benjamin. Zhu, Kai Li, and Hugo Patterson. Avoiding the disk bottleneck in the Data Domain deduplication file system. In Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST). USENIX, 2008.

[7] Xiang Zhang, Zhigang Huo, Jie Ma, and Dan Meng. Exploiting data deduplication to accelerate live virtual machine migration. In Proceedings of the *2010* IEEE International Conference on Cluster Computing (CLUSTER), pages 88–96. IEEE, September 2010

[8] Avani Wildani, Ethan L. Miller, and Ohad Rodeh. HANDS: A heuristically arranged non-backup in-line deduplication system. Technical Report UCSC-SSRC-12-03, University of California, Santa Cruz, March 2012

[9] Yoshihiro Tsuchiya and Takashi Watanabe. DBLK: Deduplication for primary block storage. In Proceedings of the *27*th IEEE Symposium on Mass Storage Systems and Technologies (MSST), pages 1–5. IEEE, May 2011

[10] Ms. P. Minisha priya, Dr. S.Maheswari, "Performance Analysis of Cloud Storage Using Chunking Algorithm", IEEE-2018

[11] Haonan Su, Dong Zheng, Yinghui Zhang, "An Efficient and Secure Deduplication Scheme Based on Rabin Fingerprinting in Cloud Storage", IEEE-2017

[12] Huijun Wu, Chen Wang, Kai Lu, Yinjin Fu," One Size Does Not Fit All: The Case for Chunking Configuration in Backup Deduplication" Liming Zhu, IEEE-2018.