International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 4 | ISSN : 2456-3307

A Deduplication Aware similarity in Cloud with Secure Deduplication of Encrypted Data

S. Palani[1], P. Bharath Kumar Reddy [2], K. Roja[2]
1Asst. Professor Department of Computer Applications Sri Venkateswara College of Engineering and Technology(Autonomous) Chittoor, Andhra Pradesh, India
2PG scholar Department of Computer Applications Sri Venkateswara College of Engineering and Technology(Autonomous) Chittoor, Andhra Pradesh, India

ABSTRACT

Data reduction has become progressively necessary in storage systems owing to the explosive growth of digital data within the world that has ushered within the big data era. One in every of the most challenges facing large-scale data reduction is the way to maximally observe and eliminate redundancy at terribly low overheads. In this paper, we present a our scheme, a low-overhead Deduplication-Aware resemblance detection and Elimination theme that effectively exploits existing duplicate-adjacency data for extremely economical resemblance detection in data deduplication primarily based backup/archiving storage systems. the most plan behind our scheme is to use a theme, call Duplicate-Adjacency primarily based resemblance Detection (DupAdj), by considering any two data chunks to be similar (i.e., candidates for delta compression) if their various adjacent data chunks are duplicate in a very deduplication system, and then further enhance the resemblance detection efficiency by an improved super-feature approach.

Keywords: Data deduplication, delta compression, resemblance Detection.

I. INTRODUCTION

The redundancy of the information on the cloud garage is growing. Thus exploiting the replica information can help in saving the gap. It allows in lowering the time too required for moving statistics in low bandwidth network. Data discount is the method of minimizing the quantity of information that wishes to be stored in information garage surroundings. Data Deduplication has turn out to be an important and monetary manner to dispose of the redundant information segments, accordingly assuaging the stress incurred by using large amounts of statistics want to store, Fingerprints are used to symbolize and identify equal records blocks while acting statistics deduplication.

To address this task, statistics deduplication technique is desired. Data deduplication strategies are broadly utilized by storage servers to do away with the opportunities of storing multiple copies of the information. Deduplication identifies replica data portions going to be stored in storage systems also removes duplication in existing saved data in storage systems. Hence yield a big fee saving. There are strategies available for duplication checking which includes: 1) File level duplication check. 2) Chunk level duplication test. In first method, best the document with identical call are removed from the storage whereas in second, the duplicate chunks of identical documents are eliminated and shops best one reproduction of them. In this paper, we introduce our scheme, non-replica aware similarity identification and elimination our scheme. Our

scheme integrates two our schemes i.e. information non-replica and delta compression to acquire excessive statistics discount performance at less expenses. A "DupAdj" technique is proposed to take advantage of present reproduction adjacency records behind non-replica to locate almost identical information blocks for delta compression. Precisely, due to locality of comparable information in support datasets, the non-replica blocks which might be neighboring to the duplicate ones are examined as correct delta compression applicants for similarly statistics discount.

A conceptual and actual learning of the conventional terrific function method is performed, which indicates that stepped forward similarity identification for additionally delta compression is viable whilst the previously mentioned present duplicate-adjacency statistics is missing or constrained. An research into the rehabilitation of no replicated support facts indicates that delta compression has the capability to refine the statistics-repair overall execution of non-replicate most effective networks with the aid of similarly removing redundancy after deduplication and for that reason enlarging the logical space of the restoration cache.

II. PROPOSED SYSTEM

Architecture Overview:

Proposed our scheme is designed to improve resemblance detection for additional data reduction in deduplication-based backup/archiving storage systems. As shown in Figure 3, our scheme architecture consists of three functional modules, namely, the Deduplication module, the DupAdj Detection module, and the improved Super-Feature module. In addition, there are five key data structures in our scheme, namely, Dedupe Hash Table, SFeature Hash Table, Locality Cache, Container, Segment, and Chunk, which are defined below:

• **A chunk** is the atomic unit for data reduction. The non-duplicate chunks, identified by their SHA1 fingerprints, will be prepared for resemblance detection in our scheme.

• **A container** is the fixed-size storage unit that stores sequential and NOT reduced data, such as nonduplicate & non-similar or delta chunks, for better storage performance by using large I/Os .

• **A segment** consists of the metadata of a number of sequential chunks (e.g., 1MB size), such as the chunk fingerprints, size, etc., which serves as the atomic unit in preserving the backup-stream logical locality for data reduction. Here our scheme uses a data structure of doubly-linked list to record the chunk adjacency information for the DupAdj detection. Note that the SFeature in the segment may be unnecessary if the DupAdj module has already confirmed this chunk as being similar for delta compression.

• **Dedupe Hash Table** serves to index fingerprints for duplicate detection for the deduplication module.

• **SFeature Hash Table** serves to index the super features after the DupAdj resemblance detection. It manages the super-features of non-duplicate and non-similar chunks.

• **Locality Cache** contains the recently accessed data segments and thus preserves the backup-stream locality in memory, to reduce accesses to the on disk index from either duplicate detection or resemblance detection.

Here we describe a general workflow of our scheme. For the input data stream, our scheme will first detect duplicate chunks by the Deduplication module. Any of the many existing deduplication approaches can be implemented here and the preservation of the backup-stream logical locality in the segments is required for further resemblance detection. For each non-duplicate chunk, our scheme will first use its DupAdj Detection module to quickly determine whether it is a delta compression candidate. If it is not a candidate, our scheme will then compute its features and super-features, using its improved Super-Feature Detection module to further detect resemblance for data reduction.

DupAdj: Duplicate-Adjacency based Resemblance Detection:-

As a salient feature of our scheme, the DupAdj approach detects resemblance by exploiting existing duplicate adjacency information of a deduplication system. The main idea behind this approach is to consider chunk pairs closely adjacent to any confirmed duplicate-chunk pair between two data streams as resembling pairs and thus candidates for delta compression.

According to the description of the our scheme data structures in figure 3, our scheme records the backup-stream logical locality of chunk sequence by a doubly-linked list, which allows an efficient search of the duplicate adjacent chunks for resemblance detection by traversing to prior or next chunks on the list, as shown in Figure 1. When the DupAdj Detection module of our scheme processes an input segment, it will traverse all the chunks by the aforementioned doubly-linked list to find the already duplicate-detected chunks. If chunk Am of the input segment A was detected to be a duplicate of chunk Bn of segment B, our scheme will traverse the doubly-linked list of Bn in both directions (e.g., Am+1 & Bn+1 and Am–1 & Bn–1) in search of potentially similar chunk pairs between segments A and B, until a dissimilar chunk or an already detected duplicate or similar chunk is found. Note that the detected chunks here are considered dissimilar (i.e., NOT similar) to others if their similarity degree is smaller than a predefined threshold, such as 0.25, a false positive for resemblance detection. Actually, the similarity degree of the DupAdj-detected chunks tends to be very high, larger than 0.88,

• **Memory overhead**: Each chunk will be associated with two pointers (about 8 or 16 Bytes) for building the doubly-linked list when our scheme loads the segment into the locality cache. But when the segment is evicted from the cache, the doubly-linked list will be immediately freed. Therefore, this RAM memory overhead is arguably negligible given the total capacity of the locality cache.

• **Computation overhead**: Confirming the similarity degree of the DupAdj-detected chunks may introduce additional but omitted computation overhead. First, the delta encoding results for the confirmed resembling (i.e., similar) chunks will be directly used as the final delta chunk for storage. Second, the actual extra computation overhead occurs when the DupAdj-detected chunks are NOT similar, which is a very rare event as discussed in the previous paragraph.

In all, the DupAdj detection approach only adds a doubly-linked list to an existing deduplication system; our scheme avoids the computation and indexing overheads of the conventional super-feature approach. in case where the duplicate-adjacency information is lacking, limited, or interrupted due to operations such as file content insertions/deletions or new file appending, our scheme will use an improved super-feature approach to further detect and eliminate resemblance.

Improved Super-Feature Approach:-

Traditional super-feature approaches generate features by Rabin fingerprints and group these features into super-features to detect resemblance for data reduction. For example, Featurei of a chunk (length = N), is uniquely generated with a randomly pre-defined value pair mi & ai and N Rabin fingerprints (as used in Content-Defined Chunking) as follows

$$Feature_i = Max_{j=1}^{N}\{(m_i * Rabin_j + a_i) \bmod 2^{32}\} \quad (1)$$

A super-feature of this chunk, SFeaturex, can then be calculated by several such features as follows:

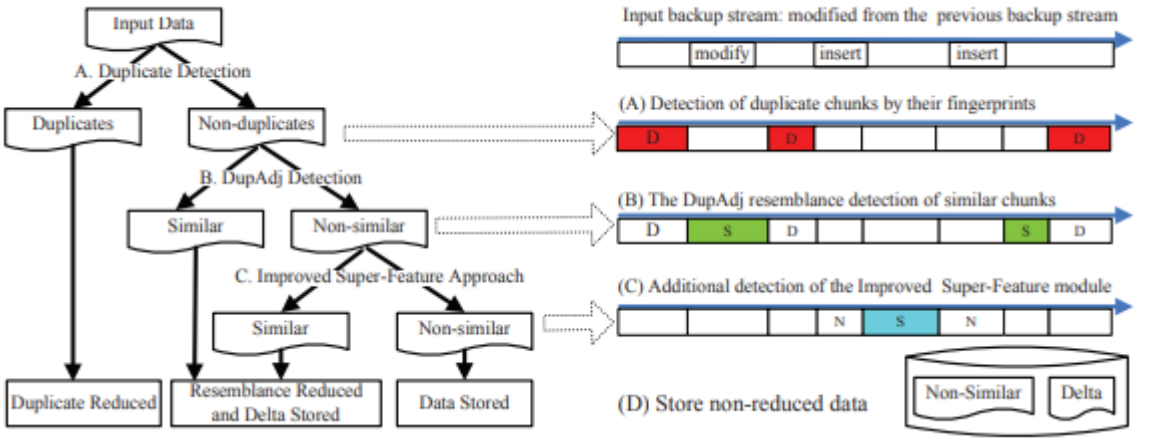
$$SFeature_x = Rabin(Feature_{x*k}, \ldots, Feature_{x*k+k-1}) \quad (2)$$

for example, to generate two super-features with k=4 features each, we must first generate 8 features, namely, features 0...3 for SF eature1 and features 4...7 for SF eature2. For similar chunks that differ only in a tiny fraction of bytes, most of their features will be

identical due to the random distribution of the chunk's maximal-feature positions. Thus two data chunks can be considered very similar if any one of their super features matches. The state-of-the-art studies on delta compression and resemblance detection recommend the use of 4 or more features to generate a super-feature to minimize false positives of resemblance detection.

Delta Compression:-

to reduce data redundancy among similar chunks, xdelta, an optimized delta compression algorithm, is adopted in our scheme after a delta compression candidate is detected by our scheme's resemblance detection. Our scheme also only carries out the one-level delta compression for similar data as employed in derd and sidc. This is because we aim to minimize the data fragmentation problem that would cause a single read request to issue multiple read operations to multiple data chunks, a likely scenario if multi-level delta compression is employed. In other words, in our scheme, delta compression will not be applied to a chunk that has already been delta compressed to avoid recursive backward referencing. And our scheme records the similarity degree as the ratio of compressed size to original size after delta compression (note that "compressed size" here refers to the size of redundant data reduced by delta compression). For example, if delta compression removes 4/5 of data volume in the input chunks detected by our scheme, then the similarity degree of the input chunks is 80%, meaning that the volume of the input chunks can be reduced to 1/5 of its original volume by the resemblance detection and delta compression techniques.



Figuer 1. The data reduction workflow

Putting It All Together:-

For an incoming backup stream, our scheme goes through the following four key steps:

1) Duplicate Detection. The data stream is first chunked, fingerprinted, duplicate-detected, and then grouped into segments of sequential chunks to preserve the backup-stream logical locality. Note that the locality information will be exploited by the following DupAdj resemblance detection.

2) Resemblance Detection. The DupAdj resemblance detection module in our scheme first detects duplicate adjacent chunks in the segments formed in step (1). After that, our scheme's improved super-feature module further detects similar chunks in the remaining non-duplicate and non-similar chunks that may have been missed by the DupAdj detection module when the duplicate-adjacency information is lacking or weak.

3) Delta Compression. For each of the resembling chunks detected in step (2), our scheme reads its base chunk, then delta encodes their differences. In order to reduce disk reads, an LRU and locality preserved

cache is implemented here to prefetch the base-chunks in the form of data segments.

4) Storage Management. The data NOT reduced, i.e., non-similar and delta chunks, will be stored as containers on the disk. The file mapping relationships among the duplicate chunks, resembling chunks, and non-similar chunks will also be recorded as the file recipe to facilitate future data restore operations in our scheme.

For the restore operation, our scheme will first read the referenced file recipes and then read the duplicate and non-similar chunks one by one from the referenced segments on disk according to mapping relationships in the file recipes. for the resembling chunks, our scheme needs to read both delta data and base-chunks and then delta decode them to the original ones.

III. CONCLUSION

In this paper, we present our scheme, a deduplication-aware, low-overhead similitude detection and elimination our scheme for data reduction in backup/archiving storage systems. our scheme uses a unique approach, DupAdj, which exploits the duplicate-adjacency information for economical resemblance detection in existing deduplication systems, and employs an improved super-feature approach to additional detecting resemblance once the duplicate adjacency information is lacking or restricted. Results from experiments driven by real-world and synthetic backup datasets counsel that our scheme will be a powerful and economical tool for increasing data reduction by additional detecting resembling data with low overheads.

IV. REFERENCES

[1]. A. Adya, W. J. Bolosky, M. Castro, R. Chaiken, G. Cermak, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. Wattenhofer. FARSITE: Federated, available, and reliable storage for an incompletely trusted environment. In Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI), Boston, MA, Dec. 2002. USENIX.

[2]. N. Agrawal, W. J. Bolosky, J. R. Douceur, and J. R. Lorch. A five-year study of file-system metadata. In Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST), pages 31-45, Feb. 2007.

[3]. R. Anderson, R. Needham, and A. Shamir. The steganographic file system. In Proceedings of the International Workshop on Information Hiding (IWIH 1998), pages 73-82, Portland, OR, Apr. 1998.

[4]. S. Annapureddy, M. J. Freedman, and D. Mazières. Shark: Scaling file servers via cooperative caching. In Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI), pages 129-142, 2005.

[5]. D. Bhagwat, K. Pollack, D. D. E. Long, E. L. Miller, J.-F. Paris, and T. Schwarz, S. J. Providing high reliability in a minimum redundancy archival storage system. In Proceedings of the 14th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '06), Monterey, CA, Sept. 2006.

[6]. W. J. Bolosky, S. Corbin, D. Goebel, and J. R. Douceur. Single instance storage in Windows 2000. In Proceedings of the 4th USENIX Windows Systems Symposium, pages 13-24. USENIX, Aug. 2000.

[7]. P. J. Braam. The Lustre storage architecture. http://www.lustre.org/documentation.html, Cluster File Systems, Inc., Aug. 2004.

[8]. A. Brinkmann, S. Effert, F. Meyer auf der Heide, and C. Scheideler. Dynamic and redundant data placement. In Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS '07), 2007.

[9]. I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. Lecture Notes in Computer Science, 2009:46-66, 2001.

[10]. J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer. Reclaiming space from duplicate files in a serverless distributed file system. In Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS '02), pages 617-624, Vienna, Austria, July 2002.

[11]. F. Douglis and A. Iyengar. Application-specific delta-encoding via resemblance detection. In Proceedings of the 2003 USENIX Annual Technical Conference, pages 113-126. USENIX, June 2003.

[12]. D. Goldschlag, M. Reed, and P. Syverson. Onion routing. Communications of the ACM, 1999.

[13]. G. R. Goodson, J. J. Wylie, G. R. G anger, and M. K. Reiter. Efficient Byzantine-tolerant erasure-coded storage. In Proceedings of the 2004 Int'l Conference on Dependable Systems and Networking (DSN 2004), June 2004.

[14]. H. S. Gunawi, N. Agrawal, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, and J. Schindler. Deconstructing commodity storage clusters. In Proceedings of the 32nd Int'l Symposium on Computer Architecture, pages 60-71, June 2005.

[15]. S. Hand and T. Roscoe. Mnemosyne: Peer-to-peer steganographic storage. Lecture Notes in Computer Science, 2429:130-140, Mar. 2002.

[16]. Health Information Portability and Accountability Act, Oct. 1996.

[17]. D. Hitz, J. Lau, and M. Malcom. File system design for an NFS file server appliance. In Proceedings of the winter 1994USENIX Technical Conference, pages 235-246, San Francisco, CA, Jan. 1994.

Author's Profile:

S. Palani working as an Assit.professor in Sri Venkateswara college of engineering &technology, Chittoor, Andhra Pradesh

P.Bharath Kumar Reddy received the PG degree from Sri Venkateswara college of engineering& technology ,Chittoor, Andhra Pradesh

K. Roja received the PG degree from Sri Venkateswara college of engineering& technology ,Chittoor, Andhra Pradesh