Application-Aware Local-Global Source Deduplication for Cloud Backup Services of Personal Storage using ALG-Dedup Algorithm

Sujata V. Randad[1] and V. R.Chirchi[2]

[1,2]*Computer Science and Information Technology Department, MBES COE, Ambajogai, Maharashtra, India.*

ABSTRACT

Abstract— Insufficient personal storage and limited system resources is the really challenging issue in today's personal computing devices. As we know these computing devices generating pervasive data continuously so the personal storage space is running out within short periods. This may be overcome by improving data deduplication efficiency in local as well global source backup services. In this paper, we have implemented ALG-Dedupe algorithm for Application-aware Local-Global source deduplication that improves data deduplication efficiency by exploiting application awareness, and further combines local and global duplicate detection to improve cloud storage capacity and to reduce deduplication. We have implemented virtual user space prototype to achieve the results.

Keywords *– Cloud backup, personal storage, source deduplication, Data deduplication for backup services, local global source deduplication.*

I. INTRODUCTION

Personal storage is found to be insufficient for today's computing devices due to their high definition data generating components. As the data is generated it very soon runs out total personal storage space of the computing device. One of the cost-effective solutions to overcome this problem is to use cloud backup services for file storage and download when necessary. But the problem occurs when data get duplicated in personal and cloud storage. This data can be deduplicated by using ALG-Dedup algorithm to avoid extra loads on cloud backup storage services. Data deduplication is an effective approach to reduce data redundancy, partitions large data objects into smaller parts also called chunks and only transfers or stores the unique chunks to improve storage space. This similar technique can also be found in some android app like Files Google app where user can easily track duplicate files and on deletion one copy is preserved in storage.

In this paper, we have implemented ALG-Dedupe algorithm for an Application-aware Local-Global source deduplication to exploit application awareness, and to combines local and global duplication detection, to achieve high deduplication efficiency while saving as much cloud storage cost as the application-aware global deduplication. Our applica-tion-aware deduplication design is motivated by the systematic deduplication analysis on personal storage. We observe that there is a significant difference among different types of applications in the personal computing environment in terms of data redundancy, sensitivity to different chunking methods, and independence in the de-duplication process. Thus, the basic idea of ALG-Dedupe is to effectively exploit this application difference and awareness by treating different types of applications independently and adaptively during the local and global duplicate check processes to significantly improve the deduplication efficiency and reduce the system overhead.[1]

This paper gives all experimental results carried out in our proposed system using deduplication algorithm like section II consists of Secure Hash algorithm, section III consists of Two Threshold Two Divisor Switch (TTTDS), section IV consists of Secure Deduplication With Efficient And Reliable Convergent Key Management, section V describes Hybrid Cloud Approach. Section VI describes Block level data Deduplication algorithm and section VII discusses on ALG-Dedupe algorithm in detail.

II. Design And Implementation

2.1 Architectural Overview:

The ALG-Dedupe algorithm is basically used to identify low-overhead local resources and high –overhead cloud resources to reduce computational overhead on cloud backup services. The system proposed in two modules first one is user space i.e. for local source deduplication and admin space i.e. for global source deduplication. The user space is allotted to a number of users who can save their data on personal storage. At the time of uploading a file to the local personal storage the file is checked automatically for data deduplication. For data duplication we follow ALG-Dedupe architecture as illustrated in Fig. 4, where tiny files are first filtered out by file size filter for efficiency reasons, and backup data streams are broken into chunks by an intelligent chunker using an application-aware chunking strategy [1]. These data chunks are then checked for deduplication with cloud or personal storage to reduce redundancy.

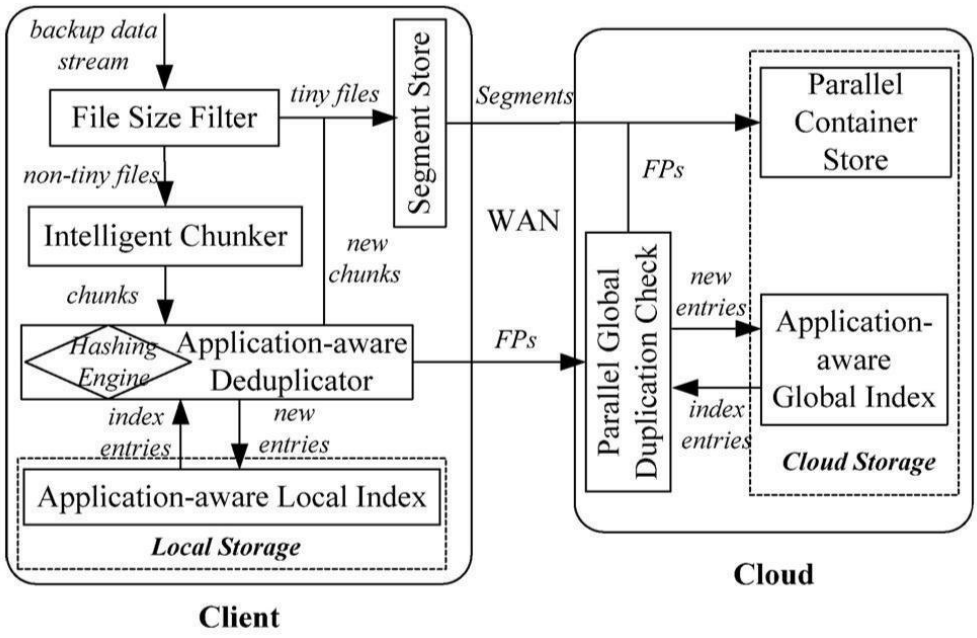

Fig. 1 Architectural overview of the ALG-Dedupe design

Then these chunks are stored in local storage space, if the chunk has similar fingerprints then file will be rejected from uploading by prompting message of deduplication. These fingerprints are derived by converting file into binary code format. We will now describe the deduplication process in more detail in the rest of this section.

2.2 File Size Filter:

In file size filter small files are taken in account for classification. It is found that these files taking more than 60% of storage space so these file are grouped together in larger chunks to increase the transfer efficiency over a network. This will reduce the metadata overhead which can cause due to the tiny nature of the files.

2.3 Intelligent Data Chunking:

Depending on whether the file type is compressed or whether Static Chunking (SC) (of 4 KB chunk size) [2] can outperform Content Defined Chunking (CDC) [3] (of 4 KB average chunk size) in deduplication efficiency, we divide files into three main categories: compressed files, static uncompressed files, and dynamic uncompressed files. The dynamic files are always editable, while the static files are uneditable in common. To achieve the better efficiency in duplicate elimination ratio and deduplication overhead, we deduplicate compressed files with Whole File Chunking (WFC) [2], separate static uncompressed files into fix-sized chunks by SC with ideal chunk size, and break dynamic uncompressed files into variable-sized chunks with optimal average chunk size using CDC based on the Rabin fingerprinting to identify chunk boundaries.[1] [9]

It is important to decide the chunk size for chunking module where both very small and very large chunks are undesirable. The very small sized chunks will lead to a larger number of chunks to be indexed, which increases the load of the indexing module. Moreover, for small chunks, the ratio between the chunk metadata size to chunk size is high, leading to performance degradation and smaller dedup savings. The very large sized chunks may exceed the allowed unit cache/memory size, which leads to implementation difficulties in other parts of the dedup systems [8]. A standard way to avoid very small and very large chunk sizes is to use thresholds for minimum and maximum chunk sizes respectively.

2.4 Application-Aware Deduplicator:

Data chunks will be deduplicated in the application-aware deduplicator by checking the chunk fingerprints. For checking this deduplication we have used binary conversion of file as shown in figure 2 (a). These chunks are checked among both the local client and remote cloud.

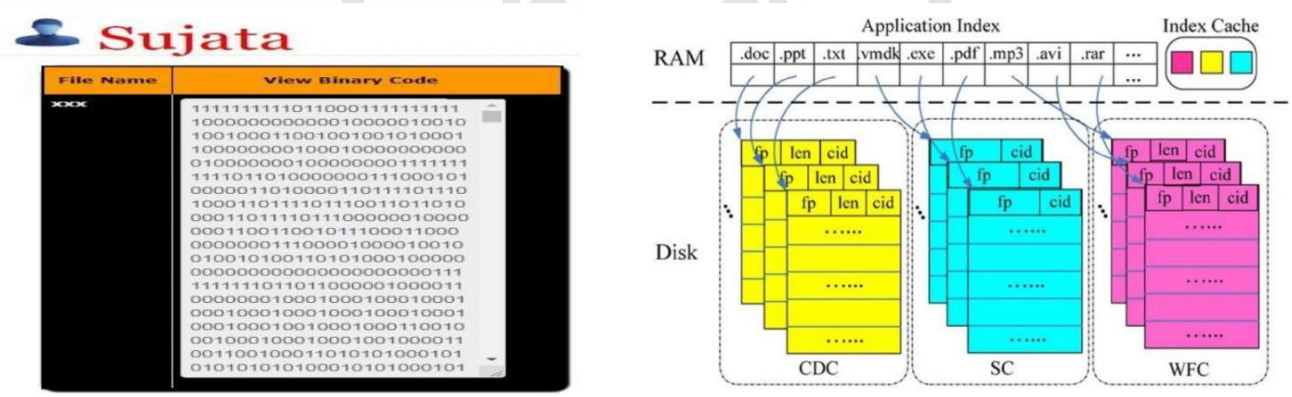

Fig. 2 (a) Binary code format of file (b) Application-aware index structure

To achieve high deduplication efficiency, the application-aware deduplicator first detects duplicate data in the application-aware local index corresponding to the local dataset with low deduplication latency in the PC client, and then compares local deduplicated data chunks with all data stored in the cloud by looking up fingerprints in the application-aware global index on the cloud side for high data reduction ratio. Only the unique data chunks after global duplicate detection are stored in the cloud storage with parallel container management. [1]

2.5 Application-Aware Index Structure:

An application-aware index structure for ALG-Dedupe is shown in Fig. 2 (b). It consists of an in-RAM application index and small hash-table. Each file chunk entry is inserted into this hash table according to the same file type. Each entry of the index stores a mapping from the fingerprint (fp) of a chunk or with its length (len) to its container ID (cid). When chunk index is found in backup data [4], a small index cache is allocated in RAM to speedup index lookup by reducing disk I/O operations. ALG-Dedupe use two application-aware chunk indices: a local index on the client side and a global index on the cloud side. When a new entry of file is to be uploaded to the cloud it will first check into local index at client side if it is not found then it will be checked with global index. Comparing with traditional deduplication mechanisms, ALG-Dedupe can achieve high deduplication throughput by looking up chunk fingerprints concurrently in small indices classified by applications rather than a single full, unclassified index for both local and global scenarios. Furthermore, a periodical data synchronization scheme is also proposed in ALG-Dedupe to backup the application-aware local index and file metadata in the cloud storage to protect the data integrity of the PC backup datasets.[1]

Table 1: Comparison with and without ALG-Dedupe Algo.

File Type	ALG-Dedupe	Without Dedupe
Total	391	566
Cdc file	271	325
Sc File	81	174
Wfc file	39	119

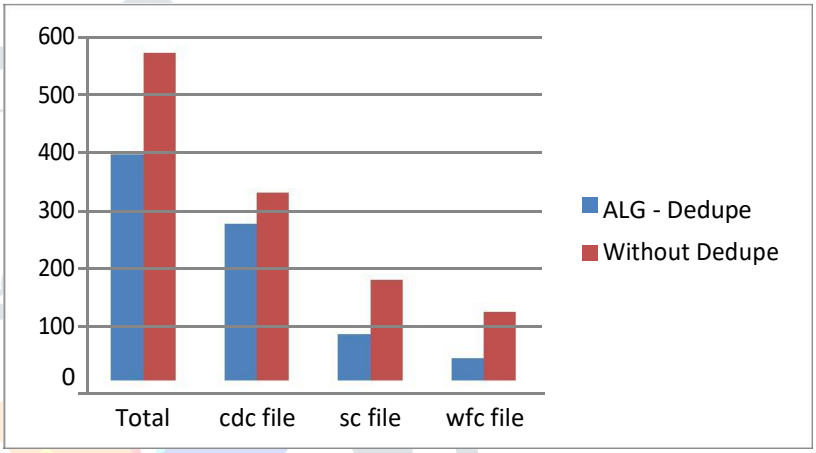

Fig. 3 (a) Parameter table (b) Comparative graph for Cloud Space

As can be seen in figure 3(a), (b) the data backup has drawbacks of over memory storage without deduplication, so it is clear that with ALG-Dedupe we can increase memory efficiency and reduce cost for cloud space. When user stores data on cloud it is checked for deduplication by checking its fingerprints and then stored on cloud storage. By applying this strategy data redundancy can be reduced.

2.6 Segment and Container Management:

By producing larger files for the cloud storage, overheads on data transfer can be reduced and also reducing the cost of the cloud storage. These larger files called as segments. After sending these segments to the cloud an entry of these segments is inserted into the application aware index structure for deduplication check up. These segments are then packed up into the several MB of size containers with their fingerprints to keep spatial locality for deduplicated data. Each new file segment uploaded to the cloud is filled into the chunk container till it reaches to its predefined size after which a new one is opened up. If a container is not full but needs to be written to disk, it is padded out to its full size.[1][5]-[8]

VIII. CONCLUSION

The ALG- dedupe is one of the deduplication algorithm which improves the efficiency of process in personal computing. Also this algorithm designed to minimize computational overhead and maximize deduplication effectiveness using application awareness.

An intelligent deduplication strategy in ALG-Dedupe is designed to exploit file semantics to minimize computational overhead and maximize deduplication effectiveness using application awareness. It combines local deduplication and global deduplication to balance the effectiveness and latency of deduplication. The proposed application-aware index structure can significantly relieve the disk index lookup bottleneck by dividing a central index into many independent small indices to optimize lookup performance.

REFERENCES

[1] Y. Fu, et al., "Application-aware local-global source deduplication for cloud backup services of personal storage", IEEE transactions on parallel and distributed systems, May 2014, vol. 25, pp. 1155-1165.

[2] D. Meister and A. Brinkmann, ''Multi-Level Comparison of Data Deduplication in a Backup Scenario,'' in Proc. 2nd Annu. Int'l SYSTOR, 2009, pp. 1-8.

[3] K. Eshghi, ''A Framework for Analyzing and Improving Content Based Chunking Algorithms,'' HP Laboratories, Palo Alto, CA, USA, Tech. Rep. HPL-2005-30 (R.1), 2005.

[4] M. Lillibridge, K. Eshghi, D. Bhagwat, V. Deolalikar, G. Trezise, and P. Camble, ''Sparse Indexing: Large Scale, Inline Dedupli-cation Using Sampling and Locality,'' in Proc. 7th USENIX Conf. FAST, 2009, pp. 111-123.

[5] A. El-Shimi, R. Kalach, A. Kumar, J. Li, A. Oltean, and S. Sengupta, ''Primary Data DeduplicationVLarge Scale Study and System Design,'' in Proc. USENIX ATC, 2012, pp. 285-296.

[6] P. Shilane, M. Huang, G. Wallace, and W. Hsu, ''WAN Optimized Replication of Backup Datasets Using Stream-Informed Delta Compression,'' in Proc. 10th USENIX Conf. FAST, 2012, pp. 49-64.

[7] F. Douglis, D. Bhardwaj, H. Qian, and P. Shilane, ''Content-Aware Load Balancing for Distributed Backup,'' in Proc. 25th USENIX Conf. LISA, Dec. 2011, pp. 151-168.

[8] Y. Fu, H. Jiang, N. Xiao, L. Tian, and F. Liu, ''AA-Dedupe: An Application-Aware Source Deduplication Approach for Cloud Backup Services in the Personal Computing Environment,'' in Proc. 13th IEEE Int'l Conf. CLUSTER Comput., 2011, pp. 112-120.

[9] Zhu, B., LI, K., and Patterson, H. "Avoiding the Disk Bottleneck in the Data Domain Deduplication File System". In FAST (2008).