Application-Aware Local-Global Source Deduplication for Cloud Backup Services of Personal Storage

Yinjin Fu, Hong Jiang, *Senior Member, IEEE*, Nong Xiao, *Member, IEEE*, Lei Tian, Fang Liu, and Lei Xu

Abstract—In personal computing devices that rely on a cloud storage environment for data backup, an imminent challenge facing source deduplication for cloud backup services is the low deduplication efficiency due to a combination of the resource-intensive nature of deduplication and the limited system resources. In this paper, we present ALG-Dedupe, an Application-aware Local-Global source deduplication scheme that improves data deduplication efficiency by exploiting application awareness, and further combines local and global duplicate detection to strike a good balance between cloud storage capacity saving and deduplication time reduction. We perform experiments via prototype implementation to demonstrate that our scheme can significantly improve deduplication efficiency over the state-of-the-art methods with low system overhead, resulting in shortened backup window, increased power efficiency and reduced cost for cloud backup services of personal storage.

Index Terms—Cloud backup, personal storage, source deduplication, deduplication efficiency, application awareness

1 Introduction

Nowadays, the ever-growing volume and value of digital information have raised a critical and increasing requirement for data protection in the personal computing environment. Cloud backup service has become a cost-effective choice for data protection of personal computing devices [1], since the centralized cloud management has created an efficiency and cost inflection point, and offers simple offsite storage for disaster recovery, which is always a critical concern for data backup. And the efficiency of IT resources in the cloud can be further improved due to the high data redundancy in backup dataset [2].

Data deduplication, an effective data compression approach that exploits data redundancy, partitions large data objects into smaller parts, called chunks, represents these chunks by their fingerprints (i.e., generally a cryptographic hash of the chunk data), replaces the duplicate chunks with their fingerprints after chunk fingerprint index lookup, and only transfers or stores the unique chunks for the purpose of communication or storage efficiency. Source deduplication that eliminates redundant data at the client site is obviously preferred to target deduplication due to the former's ability to significantly reduce the amount of data transferred over wide area network (WAN) with low communication bandwidth [19]. For dataset with logical size L and physical size P, source deduplication can reduce the data transfer time to P/L that of traditional cloud backup. However, data deduplication is a resource-intensive process, which entails the CPU-intensive hash calculations for chunking and fingerprinting and the I/O-intensive operations for identifying and eliminating duplicate data. Unfortunately, such resources are limited in a typical personal computing device. Therefore, it is desirable to achieve a tradeoff (i.e., deduplication efficiency) between deduplication effectiveness (i.e., duplicate elimination ratio) and system overhead for personal computing devices with limited system resources.

In the traditional storage stack comprising applications, file systems, and storage hardware, each of the layers contains different kinds of information about the data they manage and such information in one layer is typically not available to any other layers. Codesign for storage and application is possible to optimize deduplication based storage system when the lower-level storage layer has extensive knowledge about the data structures and their access characteristics in the higher-level application layer. ADMAD [3] improves redundancy detection by application-specific chunking methods that exploit the knowledge about concrete file formats. ViDeDup [4] is a frame-work for video deduplication based on an application-level view of redundancy at the content level rather than at the byte level. But all these prior work only focus on the

- *Y. Fu is with the State Key Lab. of High Performance Computing and School of Computer, National University of Defense Technology, Changsha, Hunan 410073, China, and also with the Department of Computer Science and Engineering, University of Nebraska, Lincoln, NE 68588 USA. E-mail: yinjinfu@gmail.com; yfu@cse.unl.edu.*
- *H. Jiang, L. Tian, and L. Xu are with the Department of Computer Science and Engineering, University of Nebraska, Lincoln, NE 68588 USA. E-mail: {jiang, tian, lxu}@cse.unl.edu.*
- *N. Xiao and F. Liu are with the State Key Lab. of High Performance Computing and School of Computer, National University of Defense Technology, Changsha, Hunan 410073, China. E-mail: {nongxiao, liufang}@nudt.edu.cn.*

Manuscript received 3 Nov. 2012; revised 21 Apr. 2013; accepted 25 June 2013. Date of publication 1 July 2013; date of current version 21 Mar. 2014. Recommended for acceptance by J. Wang.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2013.167

effectiveness of deduplication to remove more redundancy without consider the system overheads for high efficiency in deduplication process.

The existing source deduplication strategies can be divided into two categories: local source deduplication [7], [6], [21] that only detects redundancy in backup dataset from the same device at the client side and only sends the unique data chunks to the cloud storage, and global source deduplication [8], [16] that performs duplicate check in backup datasets from all clients in the cloud side before data transfer over WAN. The former only eliminates intra-client redundancy with low duplicate elimination ratio by low-latency client-side duplicate data check, while the latter can suppress both intra-client and inter-client redundancy with high deduplication effectiveness by performing high-latency duplication detection on the cloud side. Inspired by Cloud4Home [9] that enhances data services by combining limited local resources with low latency and powerful Internet resources with high latency, local-global source deduplication scheme that eliminates intra-client redundancy at client before suppression inter-client redundancy in the cloud, can potentially improve deduplication efficiency in cloud backup services to save as much cloud storage space as the global method but at as low latency as the local mechanism.

In this paper, we propose ALG-Dedupe, an *A*pplication-aware *L*ocal-*G*lobal source deduplication scheme that not only exploits application awareness, but also combines local and global duplication detection, to achieve high deduplication efficiency by reducing the deduplication latency to as low as the application-aware local deduplication while saving as much cloud storage cost as the application-aware global deduplication. Our application-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 deduplication 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.

We have made several contributions in the paper. We propose a new metric, ''bytes saved per second,'' to measure the efficiency of different deduplication schemes on the same platform. We design an application-aware deduplication scheme that employs an intelligent data chunking method and an adaptive use of hash functions to minimize computational overhead and maximize deduplication effectiveness by exploiting application awareness. We combine local deduplication and global deduplication to balance the effectiveness and latency of deduplication. To relieve the disk index lookup bottleneck, we provide application-aware index structure to suppress redundancy independently and in parallel by dividing a central index into many independent small indices to optimize lookup performance. We also propose a data aggregation strategy at the client side to improve data transfer efficiency by grouping many small data packets into a single larger one for cloud storage. Our prototype implementation and real dataset driven evaluations show that our ALG-Dedupe outperforms the existing state-of-the-art source deduplication schemes in terms of backup window, energy efficiency, and cost saving for its high deduplication efficiency and low system overhead.

The remainder of this paper is organized as follows: We formulate the research problem in Section 2 and conduct deduplication analysis on personal data in Section 3. We describe the detailed design of ALG-Dedupe in Section 4. We evaluate ALG-Dedupe by comparing it with the existing state-of-the-art schemes in Section 5. We conclude with remarks on future work in Section 6.

2 Problem Formulation

For a backup dataset with logical dataset size L, its physical dataset size will be reduced to P_L after local source deduplication in personal computing devices and further decreased to P_G by global source deduplication in the cloud, $P_L > P_G$. We divide the backup process into three parts: local duplicate detection, global duplicate detection and unique data cloud store. Here, the latencies for chunking and fingerprinting are included in duplicate detection latency. Meanwhile, we assume that there are average local duplicate detection latency T_L, average global duplicate detection latency T_G and average cloud storage I/O bandwidth B for average chunk size C, $T_G > T_L$. We can build models to calculate BWS_L and BWS_G for the average backup window size per chunk of local source deduplication based cloud backup and global source deduplication based cloud backup as in (1) and (2), respectively

$$BWS_L = T_L + \frac{C}{B} \times \frac{P_L}{L} \tag{1}$$

$$BWS_G = T_G + \frac{C}{B} \times \frac{P_G}{L}. \tag{2}$$

Though local deduplication can achieve several to tens times of duplicate elimination ratio $R = L/P_L$ with low latency, from an empirical estimation in NEC [10], global deduplication can outperform local deduplication at 20 percent to 50 percent greater in deduplication effectiveness, and the research results in EMC [20] show that inter-client data overlapping can reach up to 75 percent, though around 10 percent is more common. While the latency is always the Achilles Heel of cloud computing, and the average global duplicate detection latency per chunk T_G is dozens or hundreds of times the latency of local duplicate detection T_L [9]. To balance cloud storage cost saving and backup window shrinking in these two schemes, we choose local-global source deduplication, which reduce the backup window size by exploiting local resources to reduce deduplication latency and save cloud storage cost by leveraging cloud resources to improve deduplication effectiveness. A novel model to estimate BWS_{LG}, the average backup window size per chunk for local-global source deduplication, is expressed in (3). It can outperform

local source deduplication if (4) is satisfied when network bandwidth is very low, and also outperform global source deduplication since (5) is always satisfied due to high cloud latency

$$BWS_{LG} = T_L + T_G \times \frac{P_L}{L} + \frac{C}{B} \times \frac{P_G}{L} \tag{3}$$

$$\frac{P_G}{P_L} + \frac{B \times T_G}{C} < 1 \tag{4}$$

$$\frac{P_L}{L} + \frac{T_L}{T_G} < 1. \tag{5}$$

We can define a metric for deduplication efficiency to balance the cloud storage cost saving and backup window shrinking in source deduplication based cloud backup services. It is well understood that the deduplication efficiency is proportional to deduplication effectiveness that is always defined by duplicate elimination ratio $R = L/P$, and inversely proportional to the average backup window size per chunk BWS with average chunk size C. Based on this understanding and to better quantify and compare deduplication efficiency of a wide variety of deduplication techniques, we propose a new metric, called ''bytes saved per second,'' which is expressed in (6), to measure the deduplication efficiency DE of different deduplication schemes on the same platform. Our local-global source deduplication design can achieve high efficiency for its global deduplication effectiveness and reduced backup window

$$DE = \frac{C}{BWS} \times \left(1 - \frac{1}{R}\right). \tag{6}$$

Different from the traditional deduplication based cloud backup services that are oblivious to the file level semantic knowledge, we optimize the efficiency for the source deduplication based cloud backup services by exploiting application awareness. We can divide backup dataset into n application data subsets according to file semantics, and improve the deduplication effectiveness and decrease deduplication latency by dedicated deduplication process for each kind of application data. For application i, we define L_i, P_{Li}, and P_{Gi} as its logical data subset size, its physical data subset sizes after traditional local and global source deduplication, respectively; P_{ALi} and P_{AGi} as its physical data subset sizes after local and global application aware source deduplication, respectively. We assume its average latency per chunk for local and global application aware duplication detection are T_{ALi} and T_{AGi}, respectively, then $T_{ALi} < T_L$ and $T_{AGi} < T_G$ due to the index lookup optimization by exploiting application awareness. According to our observation in Section 3: the amount of data shared among different types of applications is negligibly small, we have $P_{Li} \approx P_{ALi}$ and $P_{Gi} \approx P_{AGi}$ for any i is established. We make formulas (7) and (8) to estimate the upper bound of physical dataset size P_{AG} and the average backup window size per chunk BWS_{ALG} for application-aware local-global source deduplication based cloud backup services, and then we know formula (9) is established, and it indicates that the efficiency of application-aware local-global source deduplication DE_{ALG} is higher than that of local-global source deduplication DE_{LG}

TABLE 1
Datasets Used for Deduplication Analysis

Workload	#Device	#App	Size
Research Desktops	5	31	907 GB
Personal Laptops	7	34	1.1 TB
Personal Workstations	2	29	875 GB
Shared Home Server	1	31	1.3 TB
Total	15	46	4.1 TB

$$\begin{aligned} BWS_{ALG} &= \sum_{i=1}^{n} \frac{L_i}{L} \times \left(T_{Li} + T_{Gi} \times \frac{P_{ALi}}{L_i} + \frac{C}{B} \times \frac{P_{AGi}}{L_i}\right) \\ &< \sum_{i=1}^{n} \frac{L_i}{L} \times \left(T_L + T_G \times \frac{P_{ALi}}{L_i} + \frac{C}{B} \times \frac{P_{AGi}}{L_i}\right) \\ &\approx \sum_{i=1}^{n} \frac{L_i}{L} \times T_L + T_G \times \sum_{i=1}^{n} \frac{P_{Li}}{L} + \frac{C}{B} \times \sum_{i=1}^{n} \frac{P_{Gi}}{L} \\ &= T_L + T_G \times \frac{P_L}{L} + \frac{C}{B} \times \frac{P_G}{L} \\ &= BWS_{LG} \end{aligned} \tag{7}$$

$$P_{AG} = \sum_{i=1}^{n} P_{AGi} \approx \sum_{i=1}^{n} P_{Gi} = P_G \tag{8}$$

$$\begin{aligned} DE_{ALG} &= \frac{C}{BWS_{ALG}} \times \left(1 - \frac{P_{AG}}{L}\right) > \frac{C}{BWS_{LG}} \times \left(1 - \frac{P_G}{L}\right) \\ &= DE_{LG}. \end{aligned} \tag{9}$$

3 Deduplication Analysis on Personal Data

In this section, we will investigate how data redundancy, space utilization efficiency of popular data chunking methods and computational overhead of typical hash functions change in different applications of personal computing to motivate our research. We perform preliminary experimental study on datasets collected from desktops in our research group, volunteers' personal laptops, personal workstations for image processing and financial analysis, and a shared home server. Table 1 outlines the key dataset characteristics: the number of devices, applications and dataset size for each studied workload. To the best of our knowledge, this is the first systematic deduplication analysis on personal storage.

Observation 1

A disproportionally large percentage of storage space is occupied by a very small number of large files with very low chunk-level redundancy after file-level dedupe.

Implication

File-level deduplication using weak hash fun-ctions for these large files is sufficient to avoid hash collisions for small datasets in the personal computing environment.

To reveal the relationship between file count and storage capacity under various file size, we collect statistics on the distribution of file count and storage space occupied by files of different sizes in the datasets listed in Table 1

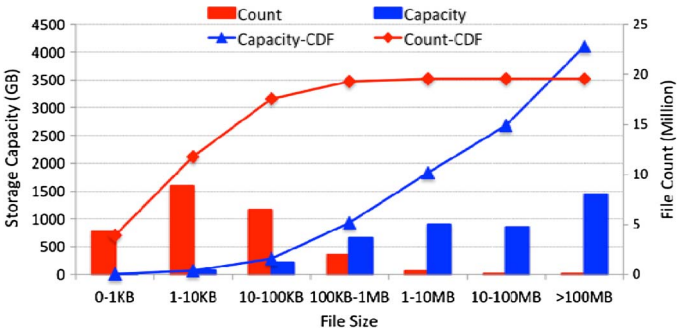


Fig. 1. Distribution of capacity and count as a function of file size. The histograms are the values of discrete density functions on file count and capacity, while the lines are cumulative distribution functions for them.

and shows the results in Fig. 1. We observe that about 60.3 percent of all files are smaller than 10 KB, accounting for only 1.7 percent of the total storage capacity, and only 1.5 percent files are larger than 1 MB but occupy 77.2 percent of the storage capacity. These results are consistent with [11]. This suggests that tiny files can be ignored during the deduplication process as so to improve the deduplication efficiency, since it is the large files in the tiny minority that dominate in determining the deduplication efficiency. In the datasets mentioned above, we also find that compressed files larger than 1 MB occupy 61.2 percent storage space. To verify the data redundancy in the compressed files, we carried out chunk-level deduplication using two popular methods: Static Chunking (SC) [11] of 4 KB chunk size and TTTD based Content Defined Chunking (CDC) [13] of 4 KB average chunk size (Min: 2 KB, Max: 16 KB) after file-level deduplication in about 2.6TB data of typical PC applications using compression, respectively. Table 2 shows the chunk-level data redundancy after file-level deduplication in typical application groups. Here, according to the function of applications, we group file types using compression into application groups: video, audio, image, Linux-AC for compressed archive file types in Linux, Mac-AC for compressed archive file types in Mac OS X, Windows-AC for compressed archive file types in Windows. From the statistics shown in Table 2, we observe

TABLE 2
Chunk-Level Redundancy after File-Level Deduplication in Typical Compressed Application Groups

Application Group	Dataset Size(GB)	Mean File Size(MB)	SC Based *R*	CDC Based *R*
Video	628	122	1.012	1.015
Audio	492	5.1	1.021	1.022
Image	554	2.2	1.018	1.023
Linux-AC	511	454	1.031	1.043
Mac-AC	195	98	1.026	1.027
Windows-AC	266	13	1.009	1.01

SC: Static Chunking, CDC: Content Defined Chunking, R: the ratio of the amounts of data before and after deduplication.

that all these file types using compression have low chunk-level redundancy because $R = 1$ means no redundancy, while files in these applications are large with MB-scale average file sizes. It is similar to the results in [18]. Owing to low chunk-level redundancy of compressed files, file-level deduplication can achieve almost the same effectiveness of data reduction as chunk-level deduplication, and it can also enhance the lookup speed for duplicate data because of reduced metadata overhead. Since the compressed files have coarse granularity and these file count is very small in the personal storage, a weak hash function is sufficient to avoid hash collisions in local file-level deduplication of compressed files.

Observation 2

The optimal combination of chunking and hash fingerprinting methods can reduce system overheads on resource-limited personal computing devices.

Implication

The use of weaker hash functions for coarse-grained chunks and stronger hash functions for fine-grained chunks is an effective way to reduce the computational overhead and RAM usage for local deduplication in PC clients.

To evaluate the computational overhead of typical hash functions on datasets in Table 1, we measured the 4-thread parallel computing throughputs of the Rabin hash, MD5 and SHA-1 hash algorithms respectively in user space on a laptop with 2.53 GHz Intel Core 2 Duo for fingerprinting data chunks obtained, respectively, from the Whole File Chunking (WFC), which uses an entire file as the basis for duplicate detection [11], the SC-based chunking with 4 KB fixed chunk size, and the CDC-based chunking with 4 KB average chunk size. Rabin hash is a rolling hash function with lower computational overhead than cryptographic hash functions SHA-1 and MD5. According to our comparative analysis in the supplementary filewhich is available in the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/167, we find that 12-byte extended Rabin hash for WFC-based deduplication on compressed files, SHA-1 for SC-based deduplication and MD5 for CDC-based dedupli-cation on uncompressed files can keep almost the same level collision resistance for deduplication on personal datasets. As shown in Fig. 2, the average throughput of chunking and fingerprinting in WFC-based deduplication is almost the same as that of SC-based deduplication, since the bulk of the processing

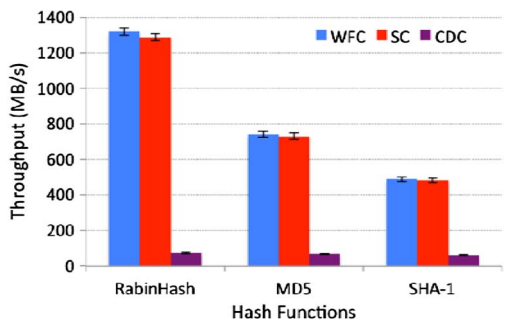


Fig. 2. Throughput of chunking and fingerprinting.

time is spent on the hash calculation itself. CDC-based deduplication has the lowest throughput on chunking and fingerprinting because most of its computational overhead is on identifying the chunk boundaries instead of chunk fingerprinting. The deduplication strategy based on simpler chunking schemes (e.g., WFC or SC) can achieve a higher throughput because of their lower metadata storage and chunking overheads, while deduplication strategies with weaker hash functions (e.g., Rabin hash) obtain a higher throughput because of their lower computational overhead. Furthermore, the combined response time of Rabin hash and MD5 is even less than that of SHA-1. This suggests that we can employ the extended Rabin hash value as chunk fingerprint for local duplicate detection and MD5 for global duplicate detection on compressed files to reduce the computational overhead with low probability of hash collision in both small PC dataset and large-scale cloud dataset. We use SC-based deduplication with the SHA-1 or CDC-based deduplication with MD5 for both local and global deduplication on those uncompressed application datasets.

Observation 3

The amount of data shared among different types of applications is negligibly small due to the difference in data content and format in these applications.

Implication

Application-aware deduplication has a potential to improve the efficiency of deduplication by eliminating redundancy in each application independently and in parallel.

We first made this proposition in our primary study with empirical observations and analysis [21], which was subsequently confirmed by a recent Microsoft Research's paper [18] in their datasets from 15 globally distributed servers. To guide our application-aware deduplication design, we conduct a content overlapping analysis to exploit the independent parallel local-global deduplication among the clients and in the cloud. We first examine the data redundancy of intra-application and inter-application by measuring the space savings of deduplication within applications and across applications. To discover the data redundancy, we chunk files with a fixed chunk size of 4 KB and calculate the corresponding MD5 value as the chunk fingerprint in each application dataset. We first compare fingerprints in each application for intra-application redundancy, then compare fingerprints between any two applications to identify the overlapping data between these applications for inter-application redundancy in all datasets. As seen in Table 3, we find that the loss in deduplication savings is negligibly small for all datasets when partitioning application dataset by file type and only performing intra-application deduplication. So the amount of shared data among the application groups is negligibly small due to the difference in data content and format in application datasets, which makes independent parallel deduplication among different application groups possible. As a result, the full fingerprint index can be divided into small independent indices according to the data type information in different applications, enabling it to greatly benefit from small indices to avoid on-disk index lookup bottlenecks [14], [15] by leveraging data locality in applications to prefetch appropriate application indices into memory, while exposing higher index access parallelism with low lock contention on chunk index structure.

TABLE 3
Space Saving for Inter- or Intra-Application Deduplication by File Type Directed Classification

Workload	Intra-App	Inter-App
Research Desktops	37%	1.1%
Personal Laptops	39%	0.9%
Personal Workstations	14%	0.6%
Shared Home Server	21%	0.7%
Total Redundancy	27.7%	0.8%

Observation 4

To exploit chunk-level redundancy, the best choices of chunking method and chunk size to achieve high deduplication efficiency vary with different application datasets.

Implication

For each application data subset, dedicated deduplication design can significantly improve deduplication efficiency over traditional deduplication schemes with single chunking method and solely chunk size for all application types.

To discover high chunk-level redundancy, we need to choose chunking method and chunk size to strike a good balance between the capability of redundancy discovery and the deduplication overhead. We always use SC-based or CDC-based deduplication schemes. The effectiveness of the former lies in its simplicity in splitting files into small chunks with a fixed chunk size. The latter partitions data into variable size chunks based on the data content rather than the data position to avoid the chunk boundary-shifting problem [13] caused by data updates with high computational overhead. It is also important to select chunk size since poor chunk size selection harms efficiency: too large chunk size reduces the exploitable redundancy in datasets, while too small chunk size can greatly increase the overhead of representing and transferring the datasets. We test the efficiency of local-global source deduplication based cloud backup service on 3 application datasets with high redundancy: 160 GB Linux kernel source code (Linux), 313 GB Virtual Machine disk images (VM) and 87 GB Microsoft Word documents with multiple versions (DOC), as a function of chunking method and chunk size. The results are shown in Fig. 3 with 4.3 MB/s mean upload bandwidth and about 300 ms average cloud latency for duplicate detection. We observe that the optimal chunk size for the highest deduplication efficiency varies among different application types. Hence, our application-aware deduplication design can significantly improve the deduplication efficiency for each application data subset by adaptively selecting chunking method and chunk size according to dataset characteristics.

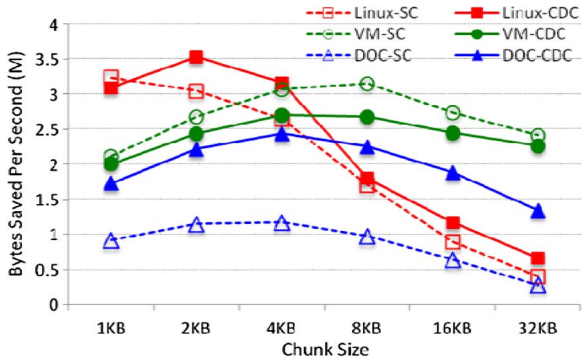


Fig. 3. Difference of deduplication efficiency as a function of chunk size and chunking method for various applications. Linux dataset can achieve highest efficiency by CDC scheme with 2 KB average chunk size, VM dataset reaches best efficiency by SC method with 8 KB chunk size, and DOC dataset can get peak efficiency by CDC scheme with 4 KB average chunk size.

4 Design and Implementation

ALG-Dedupe, motivated in part by our observations made in Section 2, is designed to meet the requirement of deduplication efficiency with high deduplication effectiveness and low system overhead. The main idea of ALG-Dedupe is 1) exploiting both low-overhead local resources and high-overhead cloud resources to reduce the computational overhead by employing an intelligent data chunking scheme and an adaptive use of hash functions based on application awareness, and 2) to mitigate the on-disk index lookup bottleneck by dividing the full index into small independent and application-specific indices in an application-aware index structure. It combines local-global source deduplication with application awareness to improve deduplication effectiveness with low system overhead on the client side.

4.1 Architecture Overview

An architectural overview of ALG-Dedupe is 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. Data chunks from the same type of files are then deduplicated in the application-aware deduplicator by generating chunk fingerprints in hash engine and performing data redundancy check in application-aware indices in both local client and remote cloud. Their fingerprints are first looked up in an application-aware local index that is stored in the local disk for local redundancy check. If a match is found, the metadata for the file containing that chunk is updated to point to the location of the existing chunk. When there is no match, the fingerprint will be sent to the cloud for further parallel global duplication check on an application-aware global index, and then if a match is found in the cloud, the corresponding file metadata is updated for duplicate chunks, or else the chunk is new. On the client side, fingerprints will be transferred in batch and new data chunks will be packed into large units called segments in the segment store module with tiny files before their transfers to reduce cloud computing latency and improve network bandwidth efficiency over WAN. On the cloud datacenter side, segments and its corresponding chunk fingerprints are stored in a self-describing data structure—container—in cloud storage, supported by the parallel container store. We will now describe the deduplication process in more detail in the rest of this section.

4.2 File Size Filter

Most of the files in the PC dataset are tiny files that less than 10 KB in file size, accounting for a negligibly small percentage of the storage capacity. As shown in our statistical evidences in Section 2, about 60.3 percent of all files are tiny files, accounting for only 1.7 percent of the total storage capacity of the dataset. To reduce the metadata overhead, ALG-Dedupe filters out these tiny files in the file size filter before the deduplication process, and groups data from many tiny files together into larger units of about 1 MB each in the segment store to increase the data transfer efficiency over WAN.

4.3 Intelligent Data Chunking

The deduplication efficiency of data chunking scheme among different applications differs greatly as we discussed in Section 2. Depending on whether the file type is compressed or whether SC can outperform CDC 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. Some examples are shown in Fig. 5. To strike a better

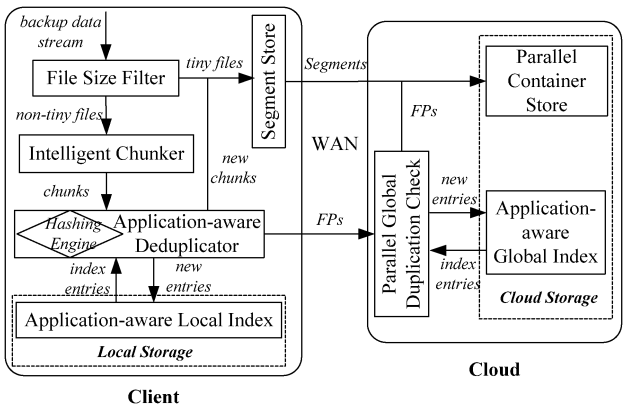


Fig. 4. Architectural overview of the ALG-Dedupe design.

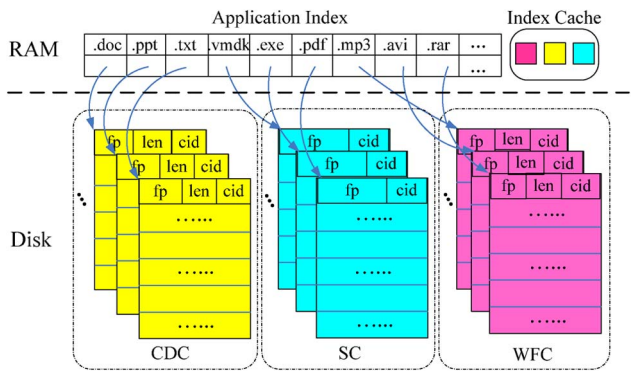


Fig. 5. Application-aware index structure.

tradeoff between duplicate elimination ratio and deduplication overhead, we deduplicate compressed files with WFC, 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.

4.4 Application-Aware Deduplicator

After data chunking in intelligent chunker module, data chunks will be deduplicated in the application-aware deduplicator by generating chunk fingerprints in the hash engine and detecting duplicate chunks in both the local client and remote cloud. ALG-Dedupe strikes a good balance between alleviating computation overhead on the client side and avoiding hash collision to keep data integrity. We employ an extended 12-byte Rabin hash value as chunk fingerprint for local duplicate data detection and a MD5 value for global duplicate detection of compressed files with WFC. In both local and global detection scenarios, a SHA-1 value of chunk serves as chunk fingerprint of SC in static uncompressed files and a MD5 value is used as chunk fingerprint of dynamic uncompressed files since chunk length is another dimension for duplicate detection in CDC-based deduplication. 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.

4.5 Application-Aware Index Structure

An application-aware index structure for ALG-Dedupe is constructed, which is shown in Fig. 5. It consists of an in-RAM application index and small hash-table based on-disk indices classified by application type. According to the accompanied file type information, the incoming chunk is directed to the chunk index with 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). As chunk locality exists in backup data streams [15], a small index cache is allocated in RAM to speedup index lookup by reducing disk I/O operations. The index cache is a key-value structure, and it is constructed by a doubly linked list indexed by a hash table. When the cache is full, fingerprints of those containers that are ineffective in accelerating chunk fingerprint lookup are replaced to make room for future prefetching and caching. The current cache replacement policy in ALG-Dedupe is Least-Recently-Used (LRU) on cached chunk fingerprints.

ALG-Dedupe requires two application-aware chunk indices: a local index on the client side and a global index on the cloud side. 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.

4.6 Segment and Container Management

Aggregation of data produces larger files for the cloud storage, which can be beneficial in avoiding high overhead of lower layer network protocols due to small transfer sizes, and in reducing the cost of the cloud storage. Amazon S3, for example, has both a per-request and a per-byte cost when storing a file, which encourages the use of files greater than 100 KB. ALG-Dedupe will often group deduplicated data from many smaller files and chunks into larger units called segments before these data are transferred over WAN.

After a segment is sent to the cloud, it will be routed to a storage node in the cloud with its corresponding fingerprints, and be packed into container, a data stream based structure, to keep spatial locality for deduplicated data. A container includes a large number of chunks and their metadata, and it has a size of several MB. An open chunk container is maintained for each incoming backup data stream in storage nodes, appending each new chunk or tiny file to the open container corresponding to the stream it is part of. When a container fills up with a predefined fixed size, 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. This process uses chunk locality to group chunks likely to be retrieved together so that the data restoration performance will be reasonably good. Supporting deletion of files requires an additional process in the background. The similar scheme is also adopted in the state-of-the-art schemes such as DDFS [14] and Sparse Indexing [15] to improve manageability and performance.

5 Evaluations

We have built a prototype of ALG-Dedupe in approximately 6000 lines of C++ code. We have evaluated the advantages of our design over the state-of-the-art source-deduplication based cloud backup services in terms of deduplication efficiency, backup window size, energy consumption, monetary costs and system overheads by feeding the real-world datasets in a personal computing device. The following evaluation subsections will show the results, beginning with a description of the experiment platform with PC backup datasets we use as inputs.

5.1 Experiment Platform and Datasets

Our experiments were performed on a MacBook Pro client with 2.53 GHz Intel Core 2 Duo processor, 4 GB RAM, and one 500 GB SATA disk, and the Amazon Web Service for cloud storage, including Amazon SimpleDB for application-aware global chunk index store and Amazon Simple Storage Service (S3) for unique data chunk store. The Macbook is connected to the Internet by campus wireless network connectivity with 10 Mbps $\sim$ 50 Mbps data transfer speed. To support our application-aware

global index structure, we create different domains for each file-type by horizontal partitioning of chunk fingerprints to improve overall throughput with parallel fingerprint lookup. First, all the datasets from more than 15 clients in Table 1 are deduplicated by our ALG-Dedupe scheme, but we only store the global chunk index in SimpleDB without unique data chunk store in S3 to save cloud storage cost and protect data privacy. Then, we use new backup datasets in the *users* directory of one of the author's PCs as workloads to drive our evaluations. There are 10 consecutive weekly full backups in the workloads with a total of 3.81TB logical data capacity consisting of about 54 million files in 17 applications.

We compare ALG-Dedupe against a number of state-of-the-art schemes, including Jungle Disk [17], a file incremental cloud backup scheme, BackupPC [6], a local file-level source deduplication based cloud backup, Cumulus [7], a local chunk-level source deduplication method, SAM [5], a hybrid cloud backup scheme with local chunk-level and global file-level source deduplication, and AA-Dedupe [21], the local-deduplication-only scheme of ALG-Dedupe. To make fair comparisons among these mechanisms, besides ALG-Dedupe, we also implement SAM and AA-Dedupe with the Amazon Web Service for cloud services in our experiments, and choose Amazon S3 to store the unique data for other existing cloud backup services.

5.2 Deduplication Effectiveness

Our experimental results in Fig. 6 present the cumulative cloud storage capacity required of the providers at each backup session for individual user with the six cloud backup schemes. Different from source deduplication schemes, Jungle Disk fails to achieve high cloud storage saving due to the fact that its incremental backup scheme cannot eliminate file copies written in different places. In the source deduplication schemes, the coarse-grained method BackupPC cannot find more redundancy than other fine-grained mechanisms. The fine-grained Cumulus only performs local duplicate check, and limits the search for unmodified data to the chunks in the previous versions of the file, so it achieves lower space saving than the local-deduplication-only application-aware deduplication AA-Dedupe. ALG-Dedupe improves the deduplication ratio of AA-Dedupe by further leveraging global deduplication with cloud computing. By leveraging application awareness to achieve global chunk-level deduplication effectiveness, it also outperforms SAM that combines local chunk-level and global file-level source deduplication. Due to the application-aware design and global duplicate detection, ALG-Dedupe can outperform Jungle Disk by 64 percent, save 43 percent space for Cumulus, and reduce a third storage usage for SAM. Comparing with the AA-Dedupe, it achieves 23 percent higher space efficiency.

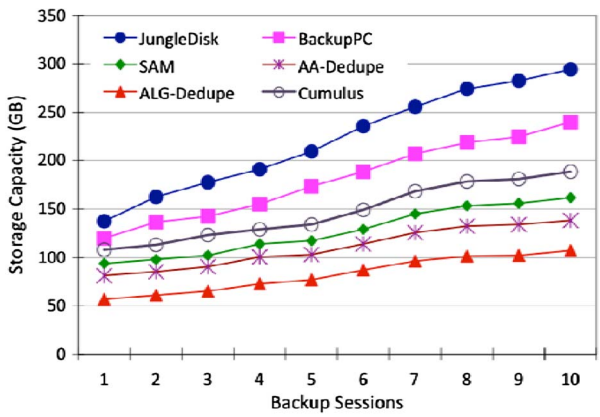


Fig. 6. Cloud storage space requirement.

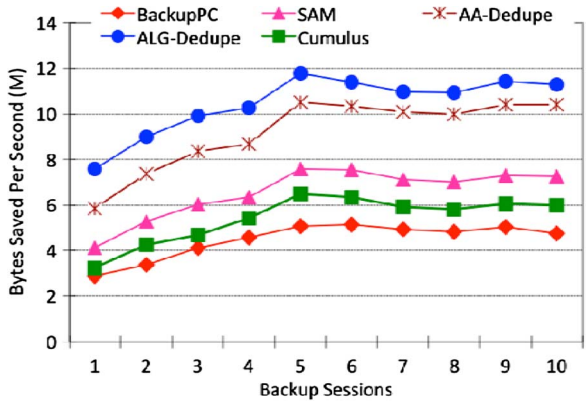


Fig. 7. Data deduplication efficiency of the backup datasets.

5.3 Deduplication Efficiency

The high effectiveness of data deduplication of the fine-grained or global deduplication schemes comes at a significant overhead that throttles the system throughput. In our ALG-Dedupe, we perform parallel local duplication detection on shared hash-table based application-aware index structure that is stored in RAM at client side. For high parallel global duplication check in SimpleDB, we apply horizontal partitioning to divide the whole unclassified index into many small independent domains that are partitioned by file-type directed application grouping. Despite of the high WAN latency, we can significantly improve the global deduplication performance of ALG-Dedupe by batch I/O and parallel query. We present a comparison of the five cloud backup schemes in terms of deduplication efficiency in Fig. 7, and employ our proposed new metric of ''bytes saved per second'', defined in Section 2.3, to measure the efficiency of different deduplication approaches in the same cloud storage platform. ALG-Dedupe performs much better than other backup schemes in the deduplication efficiency measure with a low overhead. This significant advantage of ALG-Dedupe is primarily attributed to its application awareness and global duplicate detection in the deduplication process.

We observe that the deduplication efficiency of ALG-Dedupe is 14 percent higher than our previous local scheme AA-Dedupe, owing to its advantage in global design, about 1.6 times that of the application-oblivious SAM and 1.9 times that of the local-deduplication Cumulus, more than 2.3 times that of the coarse-grained BackupPC on average.

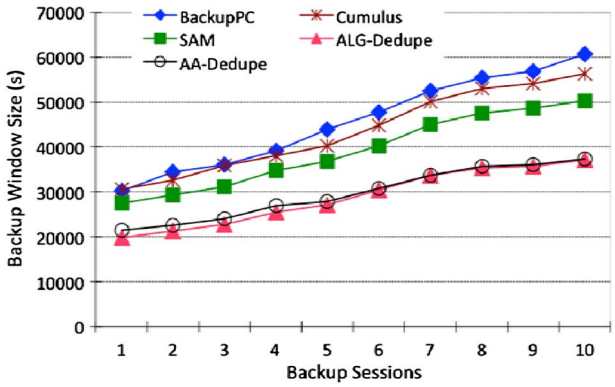


Fig. 8. Backup window size of 10 backup sessions.

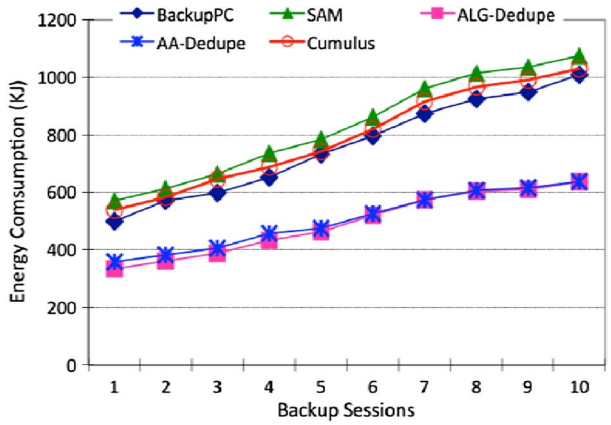


Fig. 9. Energy consumption of source deduplication schemes.

5.4 Backup Window

The backup window represents the time spent on sending a backup dataset to cloud storage, which mainly depends on the volume of the transferred dataset and available network bandwidth. Though the cloud latency affects global deduplication, our local duplicate detection can significantly reduces the number of global fingerprint lookup requests. And data transfer time is sharply decreased by our application-aware source deduplication even though the upload bandwidth is low in WAN. In our experimental results, shown in Fig. 8, BackupPC and Cumulus perform not well for their low duplicate elimination ratio; SAM has shorter backup windows due to their fine-grained and global deduplication schemes. ALG-Dedupe consistently performs the best among the five cloud backup schemes owing to its high deduplication efficiency achieved by our application-aware local-global source deduplication design. The backup window size of ALG-Dedupe is almost the same as AA-Dedupe due to its global deduplication design has high deduplication effectiveness in spite of the deduplication time is increased by global duplicate detection. We observe that the backup window size of ALG-Dedupe is shortened from other schemes by about 26 percent-37 percent in our evaluation.

5.5 Energy Efficiency

Energy efficiency has become a critical issue and its importance seems to become more pronounced in personal computing devices due to the limited energy in battery. In our experiment, we compare the power consumptions of the five source deduplication based cloud backup schemes during the deduplication process, measured by an electricity usage monitor on the whole PC. Fig. 9 shows the energy consumptions of the cloud backup schemes as a function of backup sessions. Existing approaches incur high-level power consumptions due to their significant computational overhead during the deduplication process or high data transfer overhead owing to low space saving. ALG-Dedupe incurs only 59 percent $\sim$ 65 percent that of SAM, BackupPC and Cumulus by adaptively using application-aware design in deduplication. Furthermore, it achieves almost the same energy consumption as its local scheme AA-Dedupe.

5.6 Cloud Cost

To price cloud-based backup services attractively requires minimizing the capital costs of datacenter storage and the operational bandwidth costs of shipping the data back and forth. We use the prices for Amazon S3 as an initial point in the pricing space. As of February 2012, these prices are (in US dollars) less than \$0.125 per GB · month for storage, \$0.01 per 1000 upload requests, \$0.01 per 10000 download requests and \$0.12 per GB · month for data transfer out. We define DS as dataset size, R as duplicate elimination ratio, RC as request count, CC as cloud cost. And we assume SP, TP and RP represent the price of storage, the price of transfer and the price of request, respectively. The cost of cloud backup services can be modelled as follows:

$$CC = \frac{DS}{R} \times (SP + TP) + RC \times RP. \tag{10}$$

We estimate the cloud cost of our test datasets in two months based on (10), as shown in Fig. 10. Comparing with the more space-efficient scheme SAM, file-granula-rity data transfer in Jungle Disk and BackupPC can bring more cost savings in request cost due to the large number of large files in our datasets. ALG-Dedupe can reduce the cloud cost significantly not only by global deduplication effectiveness, but also by packing several KB-sized tiny files and chunks into 1 MB segments before sending them to the cloud as Cumulus and AA-Dedupe. We observe that the cloud cost of AA-Dedupe can be reduced by 23 percent in ALG-Dedupe, which is lower than those of other

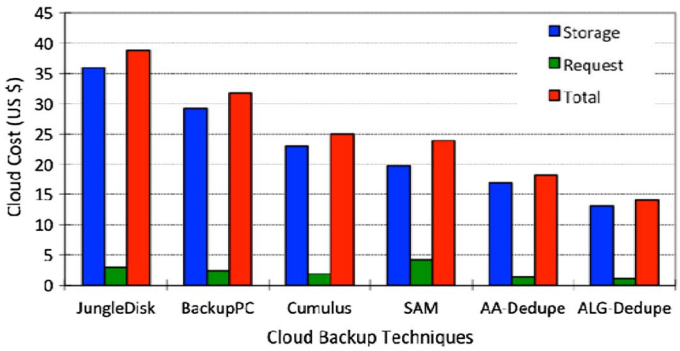


Fig. 10. Monetary cost of cloud storage.

schemes by about 41 percent-64 percent for our backup datasets.

5.7 System Overhead

Considering the limited system resources in PC clients, we estimate the system overhead in terms of CPU processing speed and RAM usage for source deduplication based cloud backup services in personal devices. In our ALG-Dedupe design, we adaptively select chunking method and hash function for different application data subsets to achieve high deduplication efficiency with low system overhead. Since the computational overhead of deduplication is dominated by CDC based chunking and hash fingerprinting, we test the average throughput of performing both chunking and fingerprinting in client for the five source deduplication based cloud backup services in 10 backup sessions. As shown in Fig. 11, our ALG-Dedupe can achieve more than 1.7 $\sim$ 13 times speed of all application-oblivious schemes, but perform slightly lower than our local scheme AA-Dedupe in processing speed due to the involvement of global deduplication. It is well known that chunk fingerprint index cost the main RAM usage for local deduplication in client. AA-Dedupe has the same local deduplication scheme as ALG-Dedupe, so they have almost the same RAM usage at client side. For the given datasets in each backup session, Fig. 12 shows that the on-disk full index size of ALG-Dedupe is slightly larger than that of file-level deduplication method BackupPC, but much smaller than that of chunk-level schemes: SAM and Cumulus. Furthermore, as discussed in Appendix C, our application-aware index structure has much higher utilization of index cache in RAM than traditional index design by exploiting application locality. In short, our ALG-Dedupe scheme can not only significantly improve computing speed for source deduplication, but also greatly save RAM usage with reduced full chunk index size and enhanced index cache efficiency.

6 Conclusion and Future Work

In this paper, we propose ALG-Dedupe, an application-aware local-global source-deduplication scheme for cloud backup in the personal computing environment to improve deduplication efficiency. 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. In our prototype evaluation, ALG-Dedupe is shown to improve the deduplication efficiency of the state-of-the-art application-oblivious source deduplication approaches by a factor of 1.6X $\sim$ 2.3X with very low system overhead, and shorten the backup window size by 26 percent-37 percent, improve power-efficiency by more than a third, and save 41 percent-64 percent cloud cost for the cloud backup service. Comparing with our previous local-deduplication-only design AA-Dedupe, it can reduce cloud cost by 23 percent without increasing backup window size. As a direction of future work, we plan to further optimize our scheme for other resource-constrained mobile devices like smartphone or tablet and investigate the secure deduplication issue in cloud backup services of the personal computing environment.

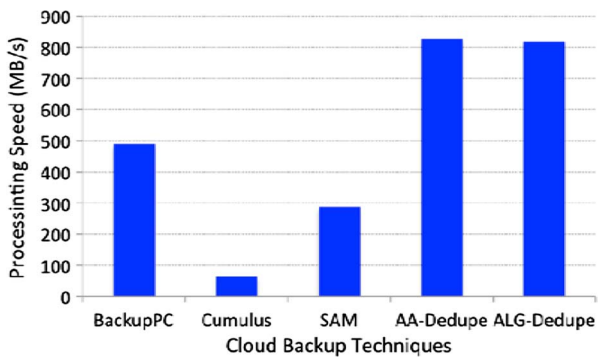


Fig. 11. Speeds of chunking and fingerprinting in PC clients.

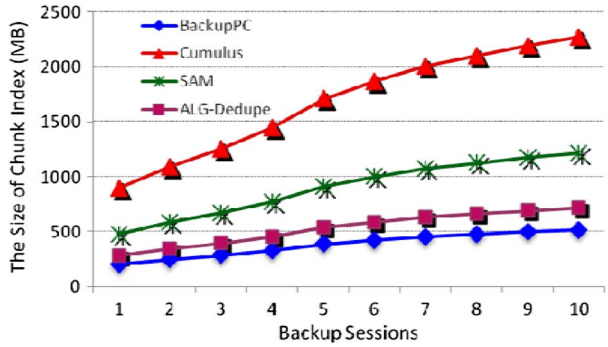


Fig. 12. The size of chunk index for source deduplication.

Acknowledgment

This research was supported in part by the 863 Program of China under Grant 2013AA013201, the National Natural Science Foundation of China under Grant 61025009, 61232003, 61120106005, 60903040, 61070198 and 61170288, China Scholarship Council, and the US NSF under Grants CCF-0937993, IIS-0916859, CNS-1016609 and CNS-1116606. N. Xiao is the corresponding author. A preliminary version of the paper was presented at the 2011 IEEE Cluster Conference.

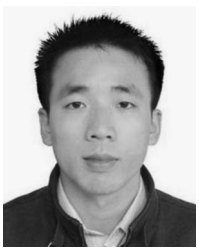
References

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, ''A View of Cloud Computing,'' *Commun. ACM*, vol. 53, no. 4, pp. 49-58, Apr. 2010.

[2] H. Biggar, ''Experiencing Data De-Duplication: Improving Efficiency and Reducing Capacity Requirements,'' Enterprise Strategy Grp., Milford, MA, USA, White Paper, Feb. 2007.

[3] C. Liu, Y. Lu, C. Shi, G. Lu, D. Du, and D.-S. Wang, ''ADMAD: Application-Driven Metadata Aware De-Deduplication Archival Storage Systems,'' in *Proc. 5th IEEE Int'l Workshop SNAPI I/Os*, 2008, pp. 29-35.

[4] A. Katiyar and J. Weissman, ''ViDeDup: An Application-Aware Framework for Video De-Duplication,'' in *Proc. 3rd USENIX Workshop Hot-Storage File Syst.*, 2011, pp. 31-35.
[5] Y. Tan, H. Jiang, D. Feng, L. Tian, Z. Yan, and G. Zhou, ''SAM: A Semantic-Aware Multi-Tiered Source De-Duplication Frame-Work for Cloud Backup,'' in *Proc. 39th ICPP*, 2010, pp. 614-623.
[6] BackupPC, 2011. [Online]. Available: http://backuppc.sourceforge.net/
[7] A. Muthitacharoen, B. Chen, and D. Mazières, ''A Low-Bandwidth Network File System,'' in *Proc. 18th ACM SOSP*, 2001, pp. 174-187.
[8] EMC Avamar, 2011. [Online]. Available: http://www.emc.com/avamar
[9] S. Kannan, A. Gavrilovska, and K. Schwan, ''Cloud4Home—Enhancing Data Services with @Home Clouds,'' in *Proc. 31st ICDCS*, 2011, pp. 539-548.
[10] Maximizing Data Efficiency: Benefits of Global Deduplication-NEC, Irving, TX, USA, NEC White Paper, 2009.
[11] 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.
[12] D. Bhagwat, K. Eshghi, D.D. Long, and M. Lillibridge, ''Extreme Binning: Scalable, Parallel Deduplication for Chunk Based File Backup,'' HP Lab., Palo Alto, CA, USA, Tech. Rep. HPL-2009-10R2, Sept. 2009.
[13] 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.
[14] B. Zhu, K. Li, and H. Patterson, ''Avoiding the Disk Bottleneck in the Data Domain Deduplication File System,'' in *Proc. 6th USENIX Conf. FAST*, Feb. 2008, pp. 269-282.
[15] M. Lillibridge, K. Eshghi, D. Bhagwat, V. Deolalikar, G. Trezise, and P. Camble, ''Sparse Indexing: Large Scale, Inline Deduplication Using Sampling and Locality,'' in *Proc. 7th USENIX Conf. FAST*, 2009, pp. 111-123.
[16] P. Anderson and L. Zhang, ''Fast and Secure Laptop Backups With Encrypted De-Duplication,'' in *Proc. 24th Int'l Conf. LISA*, 2010, pp. 29-40.
[17] Jungle Disk. 2011. [Online]. Available: http://www.jungledisk.com/
[18] A. El-Shimi, R. Kalach, A. Kumar, J. Li, A. Oltean, and S. Sengupta, ''Primary Data Deduplication—Large Scale Study and System Design,'' in *Proc. USENIX ATC*, 2012, pp. 285-296.
[19] 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.
[20] 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.
[21] 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.



Yinjin Fu received the BS degree in mathematics from Nanjing University, Nanjing, China, in 2006, and the MS degree in computer science from National University of Defense Technology (NUDT), Changsha, China, in 2008. He joined the Department of Computer Science and Engineering at University of Nebraska-Lincoln as a visiting scholar from 2010 to 2012. Now he is a PhD candidate in the State Key Laboratory of High Performance Computing at NUDT, China. His current research interests include data deduplication, cloud storage and distributed file systems. He is a student member of the ACM.

Hong Jiang received the BS degree in computer engineering from Huazhong University of Science and Technology, Wuhan, China, in 1982, the MS degree in computer engineering from the University of Toronto, Canada, in 1987, and the PhD degree in computer science from the Texas A&M University, College Station, in 1991. Since August 1991, he has been at the University of Nebraska-Lincoln (UNL), where he served as the vice chair of the Department of Computer Science and Engineering (CSE) from 2001 to 2007, and is a professor of CSE. His present research interests include computer architecture, computer storage systems and parallel I/O, parallel/distributed computing, cluster and grid computing. He has more than 180 publications in major journals and international conferences in these areas, including IEEE TPDS, IEEE TC, JPDC, USENIX-ATC, ISCA, MICRO, FAST, ICDCS, IPDPS, OOPLAS, ECOOP, SC, ICS, MIDDLEWARE, HPDC, ICPP, etc. He is a Senior Member of the IEEE and a member of the ACM and ACM SIGARCH.

Nong Xiao received the BS and PhD degrees in computer science from the College of Computer at National University of Defense Technology (NUDT) in China, in 1990 and 1996, respectively. He is currently a professor in the State Key Laboratory of High Performance Computting at NUDT, China. His current research interests include large-scale storage system, network computing, and computer architecture. He has more than 130 publications to his credit in journals and international conferences including IEEE TSC, IEEE TMM, JPDC, JCST, HPCA, ICCAD, MIDDLEWARE, MSST, IPDPS, CLUSTER, SYSTOR and MASCOTS. He is a member of the IEEE and ACM.

Lei Tian received the BE degree in computer science and technology from Huazhong University of Science and Technology (HUST), China, in 2001, and the ME and PhD degrees in computer architecture from HUST, in 2004 and 2010, respectively. His research interests include RAID-structured storage systems, distributed storage systems, and large-scale metadata management. He has more than 30 publications to his credit in journals and international conferences including IEEE TC, IEEE TPDS, ACM TOS, FAST, ICS, SC, HPDC, ICDCS, MSST, ICPP, IPDPS, and MASCOTS.

Fang Liu received the BS and PhD degrees in computer science from National University of Defense Technology (NUDT), China, in 1999 and 2005 respectively. Now she is an associate professor in the State Key Laboratory of High Performance Computing at NUDT, China. Her current research interests include distributed file system, network storage and solid-state storage system.

Lei Xu received the BS degree in electronic science and technology from Wuhan University, China, in 2005. Now he is a PhD student in the Department of Computer Science at the University of Nebraska Lincoln. His research interests include distributed file system, manycore architecture, and distributed storage system for Big Data.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.