Fu YJ, Xiao N, Liao XK *et al.* Application-aware client-side data reduction and encryption of personal data in cloud backup services. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 28(6): 1012–1024 Nov. 2013. DOI 10.1007/s11390-013-1394-5

# Application-Aware Client-Side Data Reduction and Encryption of Personal Data in Cloud Backup Services

Yin-Jin Fu[1] (付印金), Nong Xiao[1,*] (肖　侬), *Member*, *IEEE*, Xiang-Ke Liao[2] (廖湘科), *Member*, *IEEE* and Fang Liu[1] (刘　芳), *Member*, *CCF*

[1] *State Key Laboratory of High Performance Computing, National University of Defense Technology Changsha 410073, China*

[2] *School of Computer, National University of Defense Technology, Changsha 410073, China*

E-mail: yinjinfu@gmail.com; {nongxiao, xkliao, liufang}@nudt.edu.cn

Received December 10, 2012; revised May 6, 2013.

**Abstract** Cloud backup has been an important issue ever since large quantities of valuable data have been stored on the personal computing devices. Data reduction techniques, such as deduplication, delta encoding, and Lempel-Ziv (LZ) compression, performed at the client side before data transfer can help ease cloud backup by saving network bandwidth and reducing cloud storage space. However, client-side data reduction in cloud backup services faces efficiency and privacy challenges. In this paper, we present Pangolin, a secure and efficient cloud backup service for personal data storage by exploiting application awareness. It can speedup backup operations by application-aware client-side data reduction technique, and mitigate data security risks by integrating selective encryption into data reduction for sensitive applications. Our experimental evaluation, based on a prototype implementation, shows that our scheme can improve data reduction efficiency over the state-of-the-art methods by shortening the backup window size to 33%∼75%, and its security mechanism for sensitive applications has negligible impact on backup window size.

**Keywords** cloud backup, data reduction, application awareness, selective encryption

## 1 Introduction

Personal computing systems, such as desktops, laptops, tablets, smartphones and personal digital assistants (PDAs), have become primary platforms for many users, increasing the importance of data on these devices. To avoid data loss due to hardware failure, accidental deletion of data, or device theft/loss, individuals have increased their use of data protection and recovery tools in the personal computing devices. Due to the virtually infinite storage resources that are available on demand and charged according to usage, the cloud storage services (e.g., Amazon S3, Microsoft's Azure Storage and Google Storage) bring considerable economic advantages to both cloud providers and cloud users[1]. As shown in Fig.1, data backup for personal storage has emerged to be a particularly attractive application for outsourcing to cloud storage providers because users can manage data much more easily without having to worry about maintaining the backup infrastructure. This is possible because the centralized cloud management has created an efficiency and cost infle-

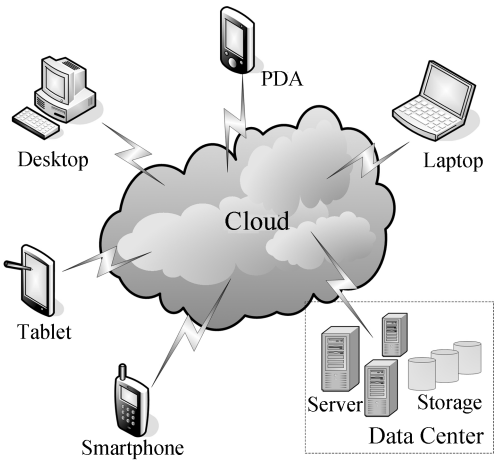


Fig.1. Cloud backup platform for personal computing devices.

Regular Paper

This work was supported in part by the National High Technology Research and Development 863 Program of China under Grant No. 2013AA013201, the National Natural Science Foundation of China under Grant Nos. 61025009, 61232003, 61120106005, 61170288, and 61379146.

*Corresponding Author

ction point, and the cloud offers simple offsite storage for disaster recovery, which is always a critical concern for data backup.

Data reduction techniques, like deduplication, delta encoding, and Lempel-Ziv (LZ) compression, can significantly improve communication and storage efficiency by exploiting data redundancy and similarity. ESG (Enterprise Strategy Group) data shows that over 90% data is duplicated in backup datasets[2]. Since data backup for personal storage in the cloud storage environment implies a geographic separation between the client and the service provider that is usually bridged by wide area networks (WANs), client-side data reduction is obviously preferred to cloud-side data reduction due to the former's ability to significantly reduce the amount of data transferred over WAN with low communication bandwidth. It has been adopted by many cloud backup services including Dropbox①, SpiderOak②, MozyHome③ and Wuala④. However, data reduction techniques are resource intensive processes, which entail the CPU-intensive calculations for hashing and compression 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 high efficiency in data reduction by taking a good balance between the data reduction ratio (i.e., the ratio of the amounts of data before and after data reduction) and backup window size.

Furthermore, data privacy protection is a necessary consideration for sensitive applications to mitigate data security risks in cloud storage. A recent study conducted by Ponemon Institute indicates that, the average cost of a data breach in lost, unprotected laptops is nearly US$40 000[3]. Traditional data encryption is incompatible with cross-user data reduction since two identical data blocks, encrypted with different keys, will yield different encrypted data blocks which can no longer be shared. Convergent encryption can produce identical ciphertexts from identical plaintext files, even if the files are encrypted by different users[4]. It was once used in popular cloud backup services, like Dropbox and Wuala. However, it leads to the leakage of private users' files to outside attackers in client-side deduplication by identifying files or learning the contents of files[5-6]. Hence, it is expected to take a good tradeoff between data privacy protection and cloud storage capacity saving.

In this paper, we propose *Pangolin*, a fast and secure cloud backup service for personal storage by exploiting application awareness. Firstly, we provide an application-aware data reduction scheme based on a significant difference of various data reduction techniques in terms of data reduction ratio and processing speed among different types of applications, to achieve high data reduction efficiency with high data reduction ratio and shorten the backup window by leveraging both limited local resources on personal computing clients and abundant remote resources of cloud computing. Secondly, due to our independent data reduction design for each application, we adopt selective data encryption for sensitive applications in cloud storage to balance the data privacy protection and cloud storage cost saving.

The main contributions of our paper include:

• According to our deep analysis of data reduction techniques on various application datasets, we first discover that the data overlapping among different applications is negligible small.

• We design an application-aware data reduction scheme to reduce cloud storage capacity and shorten backup window based on our observations on the application-oriented data reduction process.

• To protect the data privacy of sensitive applications in cloud storage, we adaptively select convergent encryption or private-key encryption to integrate into client-side data reduction for secure cloud backup services.

• Our prototype implementation of Pangolin and its real-data driven evaluation show that it outperforms the existing state-of-the-art client-side data reduction schemes in terms of data reduction efficiency and data privacy in cloud backup services.

The remainder of this paper is organized as follows. Section 2 presents the necessary background in data reduction techniques and their data leakage risks, and then conducts a preliminary quantitative study on data reduction among application datasets to motivate our research. Section 3 describes the system architecture of Pangolin, and details the design of how to integrate adaptive encryption into the application-aware data reduction design. Section 4 shows our evaluation results for Pangolin based on its prototype implementation with real datasets, by comparing it with the existing state-of-the-art schemes in terms of data reduction efficiency and encryption overheads. Section 5 presents some conclusions.

①https://www.dropbox.com/, Mar. 2010.

②https://spideroak.com/, Nov. 2011.

③http://mozy.com/home/, Feb. 2011.

④http://www.wuala.com/, Dec. 2010.

## 2 Background and Motivation

In this section, we provide the necessary background on technologies related to our research and present key observations drawn from our preliminary study to motivate our study in the application-aware client-side data reduction and encryption for cloud backup services in personal storage.

### 2.1 Data Reduction Techniques

Data reduction techniques are the most effective ways to promote data storage efficiency by deleting data redundancy. It includes data compression, delta encoding and deduplication.

Data compression eliminates redundancies within data objects to represent original information using fewer bits[7]. It can be either lossy or lossless. Lossless compression reduces bits by identifying and eliminating statistical redundancy. The LZ compression methods are among the most popular algorithms for lossless storage, and it is widely applied in compression applications, like WinRAR, GZIP and GIF. Lossy compression reduces bits by identifying marginally important information and removing it. It gives a corresponding tradeoff between information loss and the size reduction. In some popular applications, including images, audio and video, some loss of information is acceptable. Data compression only achieves a limited data reduction ratio due to its intra-object data reduction nature.

Delta encoding can compress cross data objects by exploiting data similarity[8]. It locates contents common to both the new version and the reference version of data objects, and encodes the new version by indicating contents that can be located in the reference version and contents that are added explicitly. A delta object is the encoding of the output of a differencing algorithm on the new version and the reference version of the same data object. The new version can be compressed into small patches after the reference version be transferred due to small or constant variation between the two versions. Hence, it only stores and transmits data efficiently in the form of differences between sequential data objects rather than complete data objects. To identify similar data objects, a sketch is calculated for each object by selecting several weak hash values on its shingles[8-9]. Here a shingle is a fixed length of bytes in the object. The similarity between two data objects can be defined by their sketch comparison. If two objects have the same sketch, they are likely near-duplicates. Delta encoding adds extra compression by inter-object fine-grained resemblance detection at the cost of extra computation to create sketches and lookup sketches in a cache.

Data deduplication is a specialized data reduction technique that eliminates coarse-grained redundant data by partitioning large data objects into smaller parts, called chunks, and representing or replacing these chunks by their fingerprints (i.e., generally a cryptographic hash of the chunk data) for the purpose of communication or storage efficiency[10]. It includes data chunking, hash fingerprinting, duplicate detection and unique data store. Data deduplication is also a resource-intensive process, which entails the CPU-intensive operations for chunking and fingerprinting, and I/O intensive operations for identifying and eliminating duplicate data. According to the chunking granularity, we can divide it into file-level and chunk-level deduplication, while the latter can be further classified into static chunking deduplication and content-defined chunking deduplication based on whether its chunk size is fixed or variable. Deduplication can achieve high data reduction ratio by inter-object duplicate detection in backup and archive storage systems[11].

In the above description for the three data reduction techniques, we know an optimal combination is needed due to these techniques are orthogonal to each other in achieving high storage efficiency. Recent studies[9,12] show that efficiently adding delta encoding to similar data objects in deduplicated data storage can find extra several times of capacity saving by compression. We perform data reduction in an optimal combination of the three techniques with our considerations on application awareness; it is motivated by our observations in Subsection 2.3. In our Pangolin design, we prefer to eliminate data redundancy by client-side data reduction, rather than cloud-side data reduction, since it can dramatically improve IT economy by minimizing storage requirements and network bandwidth consumption as the redundant data is eliminated prior to it traveling across the network to the cloud server. Though client-side local data reduction can achieve several to dozens of times duplicate elimination ratio with low latency, from an empirical estimation in NEC[13], server-side global deduplication can outperform local deduplication at 20% to 50% greater in deduplication effectiveness. While the latency is always the Achilles Heel of cloud computing, and the average global duplicate detection latency per chunk is dozens or hundreds of the times the latency of local duplicate detection. To balance the cloud storage cost saving and backup window shrinking in these two schemes, we choose both local and global duplicate detection, which reduces the backup window size by exploiting limited local personal computing resource to reduce latency and save cloud storage cost by leveraging abundant remote cloud computing resource to improve data reduction effectiveness.

### 2.2 Data Leakage Risk in Cloud Storage

The cloud is a multi-tenant environment where resources are shared, and it is always managed by a third party that has the potential to access a user's data. Whether accidental, or due to a malicious attack, data leakage would be a major security violation in cloud storage. Thus, all data and metadata should be encrypted at the edge before leaving the cloud client. Data compression and delta encoding before encryption on client side can add entropy for sensitive information, which is good for encryption. To save more cloud storage capacity, our client-side deduplication employs cross-user global duplicate detection. Unfortunately, deduplication and encryption are, to a great extent, diametrically opposed to each other, because the ciphertexts of the same data object after encrypted with two different keys are not identical any more. Convergent encryption is an effective way to make encryption and deduplication compatible using the hash value of a data object as the encryption key[4,14], but it has serious privacy implications[5], such as identifying files, learning the contents of files, and converting channel attack. The existing solutions[5-6] try to mitigate these data leakage risks with high system overhead or low capacity saving.

In our Pangolin design, client-side data reduction protects the data of sensitive applications from data leakage by adaptive data encryption strategy. After data reduction by deduplication, delta encoding or compression, sensitive application data is encrypted before it transfers from clients to the cloud datacenter via insecure WAN and is stored in incompletely trusted cloud backup servers. Users first classify their personal data into three categories by application awareness: common applications, low sensitive applications, and high sensitive applications. Motivated by our observation on application-aware data reduction in Subsection 2.3, each type of application data can be processed independently. We can selectively encrypt data in sensitive applications: convergent encryption for low sensitive applications to balance the cloud storage efficiency and data leakage risk, and private-key encryption for high sensitive applications to protect data privacy.

### 2.3 Motivational Observations for Application-Aware Data Reduction

In this subsection, we will investigate how data reduction techniques affect each other in terms of space efficiency and computation overhead in diverse applications, through our preliminary experimental study. In what follows, we draw the following observations and analysis from this study to motivate our research.

*Observation* 1. The best way to reduce data capacity is LZ compression for the delta-encoded data after deduplication process. Compressed files have very low data reduction ratios on delta encoding and deduplication.

Though the three popular data reduction techniques: deduplication, delta encoding and LZ compression, are widely applied in various storage systems, none of previous researches focuses on the discussion of the optimal combination of the suppression of redundancy. We test the data reduction ratios with various combinations of the data reduction techniques on 120 GB Linux source code, which is downloaded from the website⑤, and 283 GB virtual machine disk images, collected from the desktops of our laboratory. In our experiment, delta encoding always follows deduplication to make it easy to implement. We employ content-defined-chunking based deduplication with 4 KB average chunk size, similar to [15], and delta encoding approach of selecting the four smallest Rabin fingerprints of 32 B shingles as similarity features of data chunk for resemblance detection, like [8]. As shown in Fig.2, deduplication (Dedupe) performs better than LZ compression (LZ) in the Linux kernel source code dataset, and LZ compression for delta-encoded data chunks after deduplication (Dedupe-Delta-LZ) can achieve the highest data reduction ratio. However, deduplication and delta encoding (Delta) on compressed data cannot improve the data reduction ratio due to the extra entropy added by compression. Furthermore, the additional metadata overhead may reduce the data reduction ratio of compressed data.

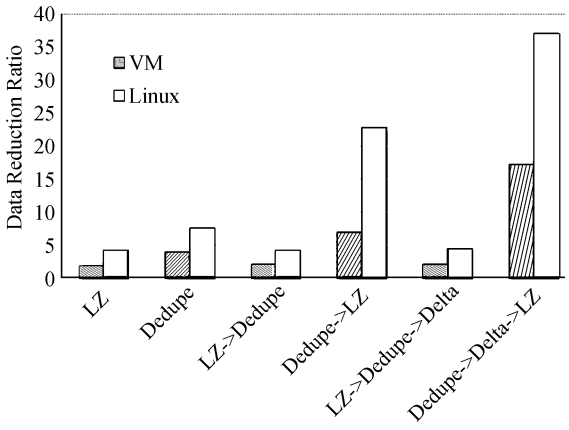


Fig.2. Data reduction techniques comparison on Linux source code and virtual machine disk images.

*Observation* 2. A disproportionally large percentage of storage space is occupied by a very small number of

⑤http://www.kernel.org/, Mar. 2012.

large compressed files with very low sub-file level redundancy in personal storage. File-level deduplication is sufficient to eliminate data redundancy for compressed files.

To reveal the relationship between file size and storage capacity, we collect statistics on the distribution of files and storage space occupied by files of different sizes in five desktops and seven laptops of eight users and show the results in Fig.3. We observe that about 71% of all files are smaller than 32 KB, accounting for only 1.9% of the total storage capacity, and only 1.4% files are larger than 2 MB but occupy 75.2% of the storage capacity. These results are consistent with previously published studies[16-17]. This suggests that small files can be ignored during the data reduction process to improve the efficiency, since it is the large files in the tiny minority that dominate the data reduction efficiency. In the datasets of 12 PCs mentioned above, there is a large number of application datasets are compressed files, such as audio, video, photo and some archive files, and those compressed files larger than 2 MB occupy 55.8% storage space. To verify the sub-file redundancy in the compressed application data, we carry out delta encoding after chunk-level deduplication using static-chunking based deduplication (SCdedupe) of 4 KB chunk size or content-defined chunking based deduplication (CDCdedupe) of 4 KB average chunk size (minimal: 2 KB, maximal: 32 KB) after file-level deduplication in about 430 GB data of typical PC applications, respectively. From statistics in Table 1, we observe that the data reduction ratios (DR) in all application types using compression almost equal 1 due to their low sub-file data redundancy while files in these applications are of large average file sizes. For low sub-file data redundancy, file-level deduplication can achieve almost the same effectiveness of data reduction as chunk-level deduplication plus delta encoding, and it can enhance the search speed for redundant data by the reduced metadata overhead. In the file-level deduplication process of compressed files, a weak hash function

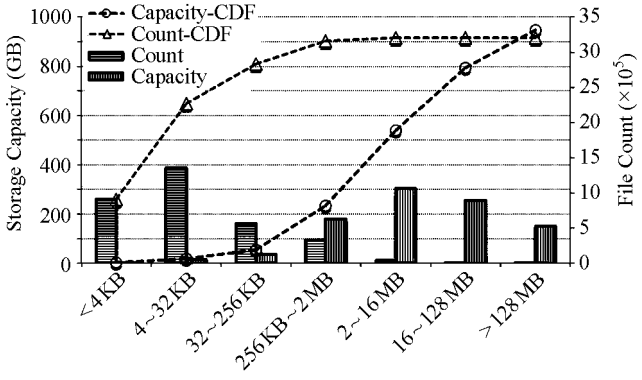


Fig.3. Distribution of file capacity and count.

**Table 1.** Subfile-Level Data Redundancy After File-Level Deduplication in Typical Compressed Applications

| File Type | Dataset Size (GB) | Mean File Size (MB) | SCdedup + Delta DR | CDCdedupe + Delta DR |
|---|---|---|---|---|
| AVI | 85 | 165 | 1.008 | 1.009 |
| MP3 | 83 | 5 | 1.012 | 1.013 |
| DMG | 41 | 105 | 1.014 | 1.014 |
| JPG | 48 | 2 | 1.010 | 1.010 |
| GZ | 70 | 43 | 1.005 | 1.007 |
| RAR | 103 | 21 | 1.011 | 1.012 |

is sufficient to avoid hash collision due to the coarse-grain nature of compressed files and the very small number of large files in personal computing environments.

*Observation* 3. The amount of data shared among different types of applications is negligible. Application-aware data reduction has the potential to improve the efficiency of data reduction by eliminating redundancy in each application independently and in parallel.

By classifying application data with file type, we compare the intra-application data reduction with global data reduction, which includes both intra-application and inter-application data reduction, using chunk-level deduplication with 4 KB chunk size and delta encoding with four similarity features for each chunk. In our preliminary study, we analyze the difference in a dataset about 750 GB with seven applications, including Linux kernel code, VM disk images, web, mail, photo, audio and video files. We find that total capacity of data overlapping among applications is 372.8 MB, which is only about 0.5% of the whole dataset size. As illustrated in Fig.4, the intra-application data redundancy for all application types is above 99.7% of total data redundancy in the dataset, so the inter-application data overlap is negligible small due to the difference in data content and format among these applications. As the data sharing among different applications is negligible, application-aware data reduction is poised to eliminate redundancy in each application independently and in parallel without reducing the

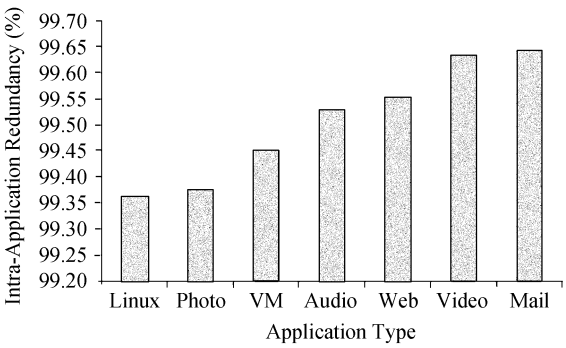


Fig.4. Intra-application redundancy in various applications.

effectiveness of data reduction. As a result, both the full fingerprint index of deduplication process and the similarity feature index in delta encoding 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[10,18] by leveraging application locality while exposing higher index access parallelism with low lock contention on independent indices.

## 3 Designs and Implementation of Pangolin

To achieve fast and secure cloud backup services, cloud clients require significant processing to reduce data before it is transferred over WAN, resulting in performance and privacy challenges. Traditional approaches to meeting these challenges typically result in a reduction in data reduction ratio or increased data leakage risk. Pangolin, motivated in part by our observations made in Section 2, is designed to meet the requirement of data reduction efficiency and data privacy. The main idea of Pangolin is to reduce the unnecessary computational overhead by employing an intelligent data chunking scheme and the adaptive use of hash functions based on application awareness, to mitigate the chunk on-disk index lookup bottleneck by dividing the full index into small independent and application specific indices in an application-aware index structure and to protect data privacy by selectively encrypting sensitive data with convergent encryption or private-key encryption.

### 3.1 Architectural Overview

An architectural overview of Pangolin is illustrated in Fig.5, where tiny files are first filtered out from backup data stream in the file size filter for efficiency reasons, and then non-tiny files are processed in application-aware data reductor to eliminate data redundancy. The application-aware data reductor performs both local redundancy detection in clients for all non-tiny files and remote global redundancy detection in cloud for data chunks of low sensitive applications by looking up application-aware fingerprint indices for deduplication and application-aware sketch indices for delta encoding. In client-side local redundancy detection, they are deduplicated with coarse granularity in an intelligent deduplicator using an application-aware deduplication strategy; compressed unique files are directly sent to selective encryption module after deduplication, while uncompressed files need further delta encoding and LZ compression before data encryption. After application-aware local data reduction, the data chunks of the sensitive applications are encrypted in the selective encryption module. Since sensitive data is typically stored in tiny files[5], tiny files and other data in high sensitive (HS) applications are processed in private-key encryption without further global redundancy suppression. While data chunks in low sensitive (LS) applications are encrypted with convergent encryption after local data reduction, global data reduction is performed on them by sending fingerprints and sketches of encrypted chunks for global redundancy detection in cloud. We will now describe the design of Pangolin in more details in the rest of this section.

### 3.2 File Size Filter

As shown in our statistical evidences in Section 2, most of the files in the PC dataset are tiny files that only occupy a negligibly small percentage of the storage capacity. Similar to SAM[19], Pangolin filters out these tiny files (i.e., less than 32 KB in file size) by the file

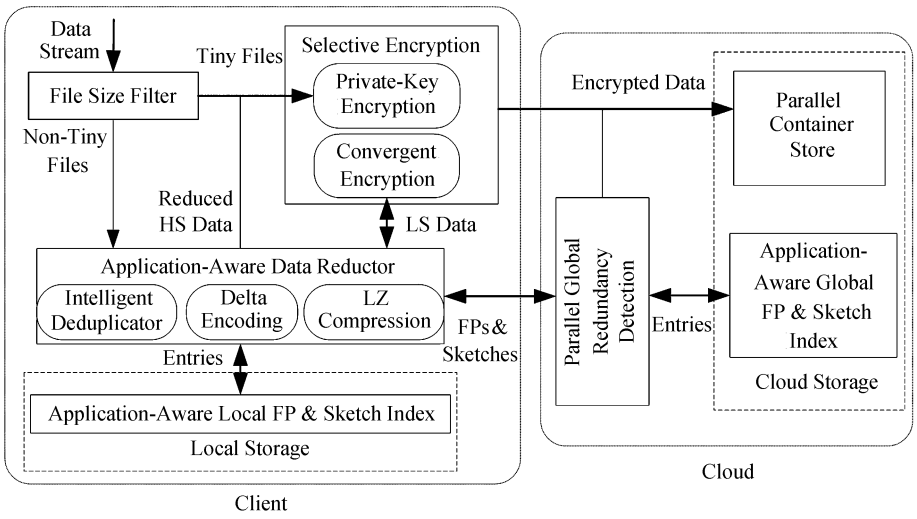


Fig.5. Architectural overview of Pangolin.

size filter before the data reduction process to reduce the metadata overhead in deduplication and delta encoding, and groups data from many smaller files together into larger units of about 1 MB each in the container store, like Cumulus[20], to increase data transfer efficiency over WAN.

### 3.3 Application-Aware Data Reductor

As we discovered in Section 2, the amount of data shared among different types of application is negligible small, and application-aware data reduction can eliminate data redundancy in each application independently and in parallel without reducing the effectiveness of data reduction. In the application-aware data reductor, we adopt whole file chunking (WFC) and MD5 hashing for compressed files to reduce the system overhead since file-level deduplication is enough to eliminate the redundancy for compressed files and their large file size as we discussed in Section 2, and employ Rabin fingerprinting based content-defined chunking (CDC) with 4 KB average chunk size and SHA1 hashing for uncompressed files to mitigate redundancy in chunk level for high deduplication effectiveness. In the intelligent deduplicator module, data chunks are deduplicated by looking up their fingerprints (FPs) in application-aware local FP index that is stored in the PC client and application-aware global FP index in the cloud. If a match is found in local FP index, the metadata for the file containing that chunk is updated to point to the location of the existing chunk; if there is no match in local storage, then the chunk is a unique data chunk for high sensitive applications, or the request is sent to the cloud for further looking up fingerprint in global FP index for data chunks in low sensitive applications. If a match is found in global FP index, the corresponding metadata is updated in application-aware global FP index in the cloud side; otherwise the chunk is a unique data chunk, and the compressed chunk is sent to the selective encryption module while the uncompressed chunk is further processed by delta encoding and LZ compression in the application-aware data reductor. In the delta encoding module, the redundancy in uncompressed unique chunks is further reduced to only transfer deltas between sequential data by exploiting chunk similarity using Broder's approach[8]; we select the four smallest Rabin fingerprints of 32 B shingles as a sketch to represent the similarity feature of data chunk for resemblance detection in delta encoding. Like FP indices, we build both application-aware local sketch index and global sketch index to find similar chunk for delta encoding in low sensitive applications. If no similar chunk is found after local sketch index search, the request is sent to the cloud for further global sketch index lookup. After resemblance detection, a candidate chunk has been selected as the base used for delta encoding, and we use a technique based on Xdelta[21] to optimize the compression of highly similar data regions. Then, the unique uncompressed chunks and their deltas are further compressed in the LZ compression module before data transfer. After the chunk or sketch is stored based on the container management in the cloud, the metadata for the associated chunk or sketch is updated to point to it and a new entry is added into both the application-aware local FP & sketch index in the client and the global FP & sketch index in cloud to index the new chunk or sketch.

### 3.4 Selective Encryption

After application-aware data reduction, compressed data chunks of sensitive applications are encrypted in the selective encryption module for secure data outsourcing in the cloud. In traditional data encryption schemes, encrypting data invalidates the deduplication for no data sharing after identical data are encrypted by two users with different keys. To protect data privacy in client deduplication based cloud backup services, we utilize convergent encryption to enable encryption while still allowing deduplication on common chunks for low sensitive applications. We define $e$ as the encryption function, $P_{\text{chunk}}$ as the plaintext of a data chunk and $C_{\text{chunk}}$ as the ciphertext of the data chunk. Convergent encryption can be defined as:

$$C_{\text{chunk}} = e(hash(P_{\text{chunk}}), P_{\text{chunk}}). \tag{1}$$

It uses the hash value $hash(P_{\text{chunk}})$ as the encryption key for $P_{\text{chunk}}$ rather than a user specified key, so that any client encrypting a given chunk will use the same key to do so. The fingerprint of data chunk *ChunkID* can be expressed as:

$$ChunkID = hash(C_{\text{chunk}}). \tag{2}$$

This scheme makes it possible to protect data privacy without compromising the data reduction efficiency, while an adversary with no knowledge of the plaintext cannot deduce the key from the encrypted chunk. However, there are several kinds of data leakage risks in convergent encryption as pointed out by Harnik *et al.*[5] For high sensitive applications, we adopt the traditional private-key encryption that encrypts data chunk with a user specified key to avoid data leakage. To provide a better security against the brute force attack than other common encryption algorithms, we adopt 256 bit AES[22] to encrypt data in our encryption implementation. All the metadata in the client file system, including the map that associates chunks with a given

file and the application-aware FP & sketch indices, are encrypted using a single private key as in [14] and [4] to simplify the data management. As data reduction and encryption occurs on the client and plaintext data is never transmitted, our scheme strengthens the system against both internal and external adversaries.

### 3.5 Application-Aware FP & Sketch Index Structure

In our application-aware data reduction scheme, data chunk fingerprint index and sketch index are key data structures in our Pangolin design. According to our discussion in Section 2, we build an application-aware FP & sketch index structure to support independent and parallel data reduction among multiple applications. As shown in Fig.6, the application-aware FP & sketch index consists of an in-RAM application index and a group of independent on-disk CDC/WFC based FP indices and sketch indices. Application index maps each filename extension to the independent FP index or sketch index of the corresponding application type. Since our cloud storage management is based on container[10], those on-disk FP/sketch indices map a chunk fingerprint/sketch to the identifier of the container (CID) that stores the chunk, and they are all hash-table based index structures. According to the accompanied file type information, the incoming chunk can be directed to the chunk index with the same file type. As the chunk locality exists in application data streams, we also build an index cache in RAM to buffer the small independent FP/sketch indices of the recent accessed applications, and the cache replacement policy is least-recently-used (LRU) on cached chunk fingerprints and sketches. Compared with the single full, unclassified index in traditional data reduction mechanisms, Pangolin can achieve high data reduction throughput by looking up chunk fingerprints in concurrently small indices classified by applications with high hit ratio in the index cache. Furthermore, a periodical data synchronization scheme is also proposed in Pangolin to backup the application-aware local FP & sketch index structure in cloud storage to protect the data integrity of the PC backup datasets.

### 3.6 Parallel Container Management

Pangolin will group data from smaller files and chunks into larger units called containers before these data are transferred over WAN. A container is a self-describing data structure in which a metadata section includes fingerprints, sketches, offsets and sizes for the stored chunks. It is also adopted in the state-of-the-art schemes such as DDFS[10] and Sparse Indexing[15] to improve management and performance. In cloud storage, to support parallel container management, an open chunk container is maintained for each incoming backup data stream, appending each new chunk or small file to the open container corresponding to the stream it is part of. When a container fills up with a predefined fixed size (e.g., 4 MB), a new one is opened up. If a container is not full but needs to be written to the disk, it is padded out to its full size. This process uses chunk locality[15] 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 to merge almost empty containers, collect garbage, and possibly

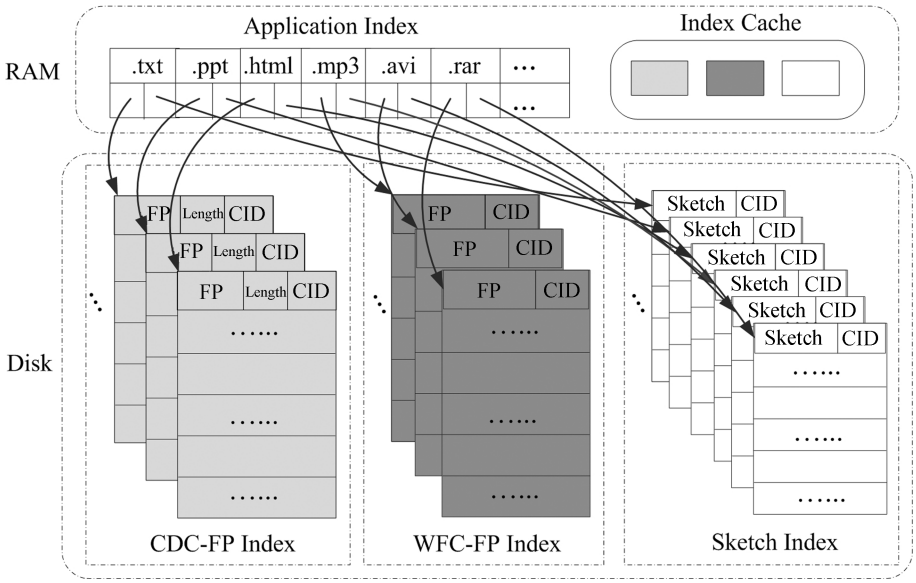


Fig.6. Application-aware FP & sketch index structure.

defragment. Aggregation of data produces larger files for cloud storage, which can be beneficial in avoiding high overhead of lower layer protocols due to small transfer sizes, and in reducing the cost of cloud storage. Amazon S3, for example, has both per-request and per-byte cost when storing a file, which encourages using files greater than 100 KB in size.

## 4 Evaluations

We have built a prototype of Pangolin in approximately 8000 lines of C++ code and fed the real-world dataset to evaluate the use of cloud backup services in personal computing environment. In the implementation of Pangolin, we choose an expected chunk size of 4 KB with 1 KB minimum and 32 KB maximum for Rabin fingerprint based CDC deduplication method using 48 bytes fixed sliding window size and 1 byte step size. Based on these results in [9], the compression differences between multi-level and one-level delta encoding is small. So we only implements one-level delta based on Xdelta to balance compression and through-put, and apply LZ compression for unique data chunks and their deltas. Our goal is to answer the following high-level questions:

• How effective is Pangolin in data reduction and shrinking backup window with real world datasets?

• What are the monetary costs of Pangolin scheme in our datasets using an online service like Amazon S3 for backup?

• What is about the performance overhead of the selective encryption mechanism in Pangolin?

The following evaluation subsections answer these questions, beginning with descriptions of experiment platform and PC backup datasets we use as inputs of the experiments.

### 4.1 Experiment Platform and Datasets

Our experiments are performed on a MacBook Pro with 2.53 GHz Intel® Core 2 Duo processor, 4 GB RAM, and 250 GB SATA disk, and it can reach about 8 Mbps average upload speed and 17 Mbps average download speed with AirPort Extreme 802.11g wireless card. We use backup datasets in one of the authors' PCs as workloads to drive our evaluations, and model the use of remote backup. The backup process is performed as depth-first search in the directory tree of file systems, and the file load overhead can be mitigated with prefetching by exploiting application locality in our application-aware design. There are 10 consecutive weekly full backups in the workloads with total 2 TB data that consist of more than 16 million files in 12 applications as listed in Table 2. We backup the reduced data into Amazon S3, and use a campus LAN server with 4-core 8-thread Intel® X3440 CPU running at 2.53 GHz and 16 GB RAM and a 2 TB RAID0 to perform the parallel global redundancy detection with reasonable remote latency added into the execution process, and more than 1 TB backup dataset from other two users in our research group is backed up before our test.

**Table 2.** Dataset Characteristics of Our Experiment

| Applications | Size (GB) | File Types |
|---|---|---|
| Document | 138 | .doc, .ppt, .pdf |
| Source code | 210 | .h, .c |
| Video | 556 | .flv, .mp4 |
| Image | 129 | .jpg, .gif |
| Archive file | 675 | .rar, .zip |
| VM disk image | 313 | .vmdk |

We compare a variety of existing cloud backup techniques with Pangolin in data reduction efficiency, backup window size and cloud storage cost. Such techniques include file incremental cloud backup Jungle Disk⑥, local file-level client-side deduplication based cloud backup like BackupPC⑦, global chunk-level client-side deduplication based cloud backup similar to Avamar⑧, hybrid cloud backup scheme with local file-level and global chunk-level client-side deduplication described in SAM[19], and local client-side application-aware deduplication based cloud backup AA-Dedupe[23]. As some of these techniques do not consider the security mechanism, we turn off the security module in others to compare the data reduction efficiency. Finally, we make a comparison for through-put of Pangolin with and without security mechanism to estimate the security cost.

### 4.2 Data Reduction Effectiveness

Data reduction effectiveness is very important for both cloud backup providers and users. Providers expect less data stored in their datacenters to reduce data storage and management cost, whereas users prefer transferring less data for shorter backup time. Our experimental results first include the data reduction ratio in Table 3 as contributed by deduplication, delta encoding, LZ compression and total data reduction. We only choose the datasets of three major applications: source code (Code), audio files (Audio) and virtual machine disk images (VM), in our personal backup

⑥http://www.jungledisk.com/, Jun. 2012.

⑦http://backuppc.sourceforge.net/, Mar. 2011.

⑧http://www.emc.com/avamar/, Oct. 2012.

datasets. The data reduction ratio is calculated as the number of input bytes divided by the number of output bytes for each stage, and the value greater than 1 indicates a data reduction improvement. Since audio files are compressed, Pangolin only performs whole file chunking based deduplication on them, so its data reduction ratios are all 1 in delta and LZ columns. For uncompressed Code and VM datasets, delta encoding and LZ compression can achieve additional 4.3∼5.5 times data reduction. We also test the cumulative cloud storage capacity for providers with various cloud backup services using data reduction technique. Fig.7 compares the cumulative storage capacity of the five cloud backup techniques. Three client-side deduplication based backup methods can perform better than the incremental backup scheme that is implemented in Jungle Disk. In the client-side deduplication methods, fine-grained Avamar and semantic-based SAM can achieve higher data reduction ratio than coarse-grained client-side deduplication scheme BackupPC by eliminating more data redundancy. AA-Dedupe outperforms the three application-oblivious deduplication methods with its application-aware design. Pangolin can get the highest space efficiency by exploiting both semantic-directed application-aware global deduplication and fine-grained duplicate elimination improved by delta encoding and compression. We observe that the data reduction ratio in Pangolin is 3.4 times that of Jungle Disk, 2.6 times that of BackupPC, about 1.7 times that of Avamar and SAM on average, and 1.4 times that of AA-Dedupe.

**Table 3.** Data Reduction Ratios of Pangolin for Backup Datasets

| Dataset | Capacity (GB) | Dedupe | Delta | LZ | Total |
|---|---|---|---|---|---|
| Code | 159.8 | 7.96 | 2.58 | 2.15 | 44.15 |
| Audio | 304.7 | 3.21 | 1.00 | 1.00 | 3.21 |
| VM | 312.9 | 3.99 | 2.32 | 1.86 | 17.22 |

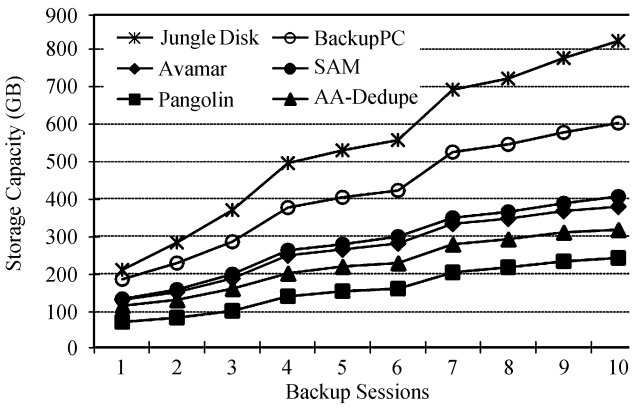


Fig.7. Cloud storage capacity requirement.

### 4.3 Backup Window

Backup window represents the time spent on sending backup dataset to cloud storage, which mainly depends on the volume of the transferred dataset and available network bandwidth. With regard to the four client-side data reduction methods in this study, the backup window consists of two parts: data reduction time and data transfer time. As we have a pipeline design for data reduction processes and data transfer operations and the data reduction throughput (10 s∼100 s MB/s) on our platform is always 1∼2 orders of magnitude faster than WAN throughput (1 s∼10 s Mb/s)[9], the backup window size of each backup session mainly depends on the data transfer process. In our experimental results shown in Fig.8, Jungle Disk can significant reduce the backup time by only transfering the incremental backup files, but it still spends too much time and increases the data management overhead. In data reduction based schemes, the backup window size is less than incremental backup for their small data volume is transferred over low bandwidth WAN after data reduction. Though Avamar can save more cloud storage capacity than SAM, it has longer backup window for its higher global deduplication overhead. AA-Dedupe can significantly shrink the backup window by its high effective application-aware deduplication and low overhead local design. Pangolin remains to perform the best in the six cloud backup schemes owing to its high data reduction effectiveness. Its backup window size can be shorten to 75% of AA-Dedupe, 33% of incremental backup Jungle Disk, 42%∼56% of the other three conventional client-side deduplication based cloud backup services.

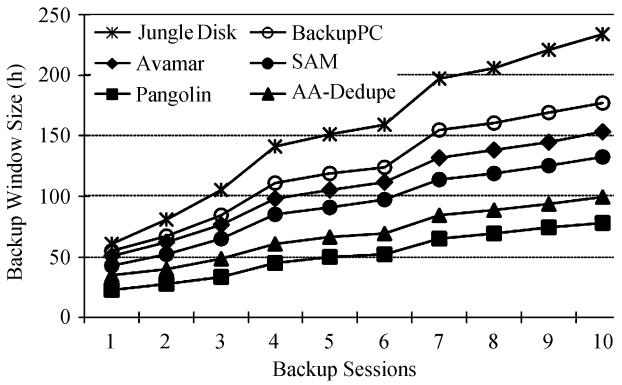


Fig.8. Backup window size of backup sessions.

### 4.4 Cloud Cost

Cloud backup as a service comes at a price. In this subsection, we calculate monetary cost for our workload models. To price cloud-based backup services attractively requires minimizing the capital cost of data

center storage and the operational bandwidth cost of shipping the data there and back. We use the prices for Amazon S3 as an initial point in the pricing space. As of April 2012, these prices are (in US dollars): $0.125 per GB per month for storage, free upload data transfer and $0.01 per 1000 upload requests. Backing up the reduced datasets directly to Amazon S3 can be very slow and costly due to a large number of I/O operations and the way Amazon S3 charges for the uploads. We investigate the cloud cost of our test datasets in the last one month as shown in Fig.9. The cost of cloud storage capacity always dominates the total cloud cost. Compared with more space efficient chunk level client-side deduplication technique Avamar, file/container granularity data transfer in Jungle Disk, BackupPC, AA-Dedupe and Pangolin can benefit more saving in request cost due to a large number of large files in our datasets. Pangolin can reduce the request cost significantly by 25%∼74% of other methods by its high data reduction effectiveness and packing several KB-size small files and chunks into 1 MB containers in PCs before sending them to the cloud.

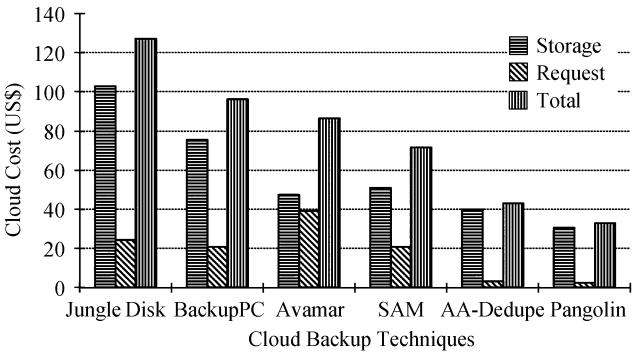


Fig.9. Cost of cloud backup techniques.

### 4.5 Index Cache Utilization

Application-aware index structure is a key data structure in our Pangolin. By leveraging the file type information in file system, it can divide the whole hash-table based chunk index into small hash-table based indices based on application type classified, thus improving the RAM efficiency for the chunk FP & sketch index by only loading the corresponding application's hash index into the memory when receiving the application's data. Our application-aware index structure can significantly improve utilization of index cache by exploiting application locality, and keep high cache hit ratio with low RAM overhead. Since the application-aware sketch index has a structure similar to the application-aware chunk fingerprint index, we only test the cache utilization for the latter to evaluate the effectiveness of our application-aware index structure. We test the cache hit ratio of chunk fingerprint lookup operations of application-aware chunk index (AACI) and traditional hash-table based full chunk index (Naive) as a function of index cache size on a 98 GB dataset with eight different applications. The results are shown in Fig.10. Our application-aware chunk index scheme can still gain high cache hit ratio above 75% even with small index cache size, while the hit ratio on conventional full chunk index drops linearly with the decrement of index cache size.

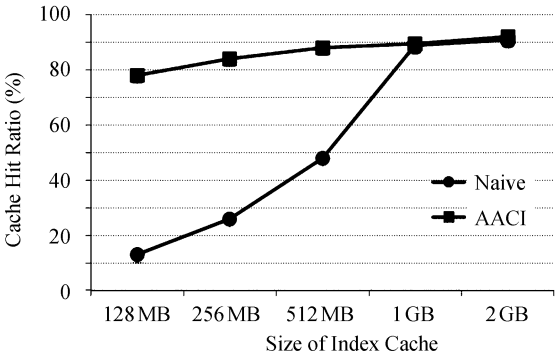


Fig.10. Cache hit ratio of various index cache sizes.

### 4.6 Security Overhead

Our Pangolin scheme can provide not only high data reduction ratio but also data privacy protection by its security mechanism. In the deduplication process of Pangolin, different from traditional deduplication schemes, two hash operations and one encryption operation are needed for convergent encryption of each data chunk, rather than only one hash operation and one encryption operation for common private-key encryption. This will increase the performance cost of data reduction process in personal computing devices. In our experiment, we evaluate the cost of Pangolin security mechanism by comparing the average data reduction throughput between Pangolin with security module (Convergent or PrivateKey) and without security module (Naive) as shown in Fig.11. The average throughput

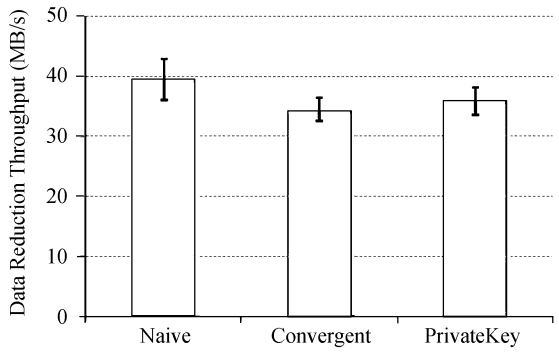


Fig.11. Security overhead of Pangolin.

of Naive is reduced only 9%∼12% during the backup process by encryption operations on reduced data. It is a reasonable cost for Pangolin to provide security cloud backup services in personal computing environment. Due to the network throughput bottleneck, this small security overhead has only negligible impact on backup window size in cloud backup process.

## 5 Conclusions and Future Work

Motivated by the key observations drawn from our preliminary experimental study, we proposed Pangolin, an application-aware client-side data reduction scheme with client-end per-user encryption for cloud backup in the personal computing environment to shorten data backup window and protect data privacy. An intelligent data reduction strategy in Pangolin was designed to exploit file semantics to minimize computational overhead with negligible loss in data reduction effectiveness. The novel data structure in Pangolin, application-aware FP and sketch index structure, can significantly relieve the disk index lookup bottleneck by dividing a central index into many independent small indices to optimize lookup performance with locality-based prefetching. The convergent encryption is adopted in Pangolin to support cross-user deduplication on encrypted data. In our prototype, Pangolin was shown to improve the data reduction efficiency of the state-of-the-art client-side deduplication approaches by shortening the backup window size to 33%∼75%, and save 25%∼74% cloud cost for the cloud backup service with high index cache utilization. And it can support data privacy protection with a reasonable cost of reduced data reduction throughput by 9%∼12% and negligible impact on backup window size.

In this paper, we only focused on how to design and implement a fast and secure data backup process. We will apply application awareness to the restore and deletion operations in cloud backup services with data reduction techniques for high efficiency. What is more, we will extend our data privacy protection strategies with secure access control in the cloud storage environment.

## References


[1] Armbrust M, Fox A, Griffith R, Joseph A D, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M. A view of cloud computing. *Communications of the ACM*, 2010, 53(4): 50-58.

[2] Biggar H. Experiencing data de-duplication: Improving efficiency and reducing capacity requirements. White Paper, the Enterprise Strategy Group, Feb. 2007. www.abtechsystems.com/files/pdfs/WP001 04.pdf, Dec. 2012.

[3] Ponemon L. The cost of a lost laptop. White Paper, Ponemon Institute, Apr. 2009. http://communities.intel.com/docs/DOC-3076, Dec. 2012.

[4] Storer M W, Greenan K, Long D D, Miller E L. Secure data deduplication. In *Proc. the 4th StorageSS*, Oct. 2008, pp.1-10.

[5] Harnik D, Pinkas B, Shulman-Peleg A. Side channels in cloud services: Deduplication in cloud storage. *IEEE Security & Privacy*, 2010, 8(6): 40-47.

[6] Halevi S, Harnik D, Pinkas B, Shulman-Peleg A. Proofs of ownership in remote storage systems. In *Proc. the 18th CCS*, Oct. 2011, pp.491-500.

[7] Blelloch G E. Introduction to data compression. Technical Report, Computer Science Department, Carnegie Mellon University, Oct. 2001. http://www.cs.cmu.edu/afs/cs/project/pscico-guyb/realworld/www/compression.pdf, Oct. 2013.

[8] Douglis F, Iyengar A. Application-specific delta-encoding via resemblance detection. In *Proc. the USENIX ATC*, Jun. 2003, pp.113-126.

[9] Shilane P, Huang M, Wallace G, Hsu W. WAN optimized replication of backup datasets using stream-informed delta compression. *ACM Transactions on Storage*, 2012, 8(4): Article No. 13.

[10] Zhu B, Li K, Patterson H. Avoiding the disk bottleneck in the data domain deduplication file system. In *Proc. the 6th FAST*, Feb. 2008, pp.269-282.

[11] Bois L D, Amatruda R. Backup and recovery: Accelerating efficiency and driving down IT costs using data deduplication. Technical Report, EMC Corporation, Feb. 2010.

[12] Shilane P, Wallace G, Huang M, Hsu W. Delta compressed and deduplicated storage using stream-in- formed locality. In *Proc. the 4th HotStorage*, June 2012, Article No. 10.

[13] Maximizing data efficiency: Benefits of global deduplication. White Paper, NEC, June 2009. http://www.knowledgestorm.com/sol_summary_5136573.asp, Dec. 2013.

[14] Anderson P, Zhang L. Fast and secure laptop backups with encrypted de-duplication. In *Proc. the 24th LISA*, Dec. 2010, Article No. 3.

[15] Lillibridge M, Eshghi K, Bhagwat D, Deolalikar V, Trezise G, Camble P. Sparse indexing: Large scale, inline deduplication using sampling and locality. In *Proc. the 7th FAST*, Feb. 2009, pp.111-123.

[16] Meister D, Brinkmann A. Multi-level comparison of data deduplication in a backup scenario. In *Proc. the SYSTOR*, May 2009, Article No. 8.

[17] Agrawal N, Bolosky W J, Douceur J R, Lorch J R. A five-year study of file-system metadata. In *Proc. the 5th FAST*, Feb. 2007, pp.31-45.

[18] Bhagwat D, Eshghi K, Long D D, Lillibridge M. Extreme binning: Scalable, parallel deduplication for chunk based file backup. In *Proc. the 17th MASCOTS*, Sept. 2009, pp.1-9.

[19] Tan Y, Jiang H, Feng D, Tian L, Yan Z, Zhou G. SAM: A semantic-aware multi-tiered source de-duplication framework for cloud backup. In *Proc. the 39th ICPP*, Sept. 2010, pp.614-623.

[20] Vrable M, Savage S, Voelker G M. Cumulus: Filesystem backup to the cloud. In *Proc. the 7th FAST*, Feb. 2009, pp.225-238.

[21] MacDonald J. File system support for delta compression [Master's Thesis]. Department of Electrical Engineering and Computer Science, University of California at Berkeley, 2000.

[22] Asenjo J C. The advanced encryption standard — Implementation and transition to a new cryptographic benchmark. *Network Security*, 2002, 2002(7): 7-9.

[23] Fu Y, Jiang H, Xiao N, Tian L, Liu F. AA-Dedupe: An application-aware source deduplication approach for cloud backup services in the personal computing environment. In *Proc. the IEEE CLUSTER*, Sept. 2011, pp.112-120.

**Yin-Jin Fu** received his B.S. degree in mathematics from Nanjing University, China, and M.S. degree in computer science from National University of Defense Technology (NUDT), Changsha, in 2006 and 2008, respectively. Now he is a Ph.D. candidate at the State Key Laboratory of High Performance Computing in NUDT. His research areas are data deduplication, cloud storage, and distributed file systems.

**Nong Xiao** received the B.S. and Ph.D. degrees in computer science from NUDT, Changsha, in 1990 and 1996, respectively. Now he is a professor in the State Key Laboratory of High Performance Computing in NUDT. His current research interests include large-scale storage system, network computing, and computer architecture.

**Xiang-Ke Liao** received the B.S. degree in computer science from Tsinghua University, Beijing, and M.S. degree in computer science from NUDT, Changsha, in 1985 and 1988, respectively. Now he is a professor and the dean of the School of Computer in NUDT. His research areas are computer architecture, operating systems, and distributed systems.

**Fang Liu** received the B.S. and Ph.D. degrees in computer science from NUDT, Changsha, in 1999 and 2005, respectively. Now she is an associate professor in the State Key Laboratory of High Performance Computing in NUDT. Her current research interests include distributed file system, network storage and solid-state storage system.