AA-Dedupe: An Application-Aware Source Deduplication Approach for Cloud Backup Services in the Personal Computing Environment

Yinjin Fu[#, §], Hong Jiang[§], Nong Xiao[#✉], Lei Tian[§], Fang Liu[#]

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

yinjinfu@gmail.com, {nongxiao, liufang}@nudt.edu.cn

[§]*Department of Computer Science & Engineering, University of Nebraska-Lincoln, Lincoln, NE, 68588, USA*

{jiang, tian, yfu}@cse.unl.edu

***Abstract*—The market for cloud backup services in the personal computing environment is growing due to large volumes of valuable personal and corporate data being stored on desktops, laptops and smartphones. Source deduplication has become a mainstay of cloud backup that saves network bandwidth and reduces storage space. However, there are two challenges facing deduplication for cloud backup service clients: (1) low deduplication efficiency due to a combination of the resource-intensive nature of deduplication and the limited system resources on the PC-based client site; and (2) low data transfer efficiency since post-deduplication data transfers from source to backup servers are typically very small but must often cross a WAN. In this paper, we present *AA-Dedupe*, an application-aware source deduplication scheme, to significantly reduce the computational overhead, increase the deduplication throughput and improve the data transfer efficiency. The AA-Dedupe approach is motivated by our key observations of the substantial differences among applications in data redundancy and deduplication characteristics, and thus is based on an application-aware index structure that effectively exploits this application awareness. Our experimental evaluations, based on an AA-Dedupe prototype implementation, show that our scheme can improve deduplication efficiency over the state-of-art source-deduplication methods by a factor of 2-7, resulting in shortened backup window, increased power-efficiency and reduced cost for cloud backup services.**

I. 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. According to IDC research [2], digital data is doubling every 18 months, and more than 30% of the information created requires high standards of protection. 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. Recent studies indicate that, while 31% of personal computer (PC) users have lost all of their PCs' files to events beyond their control [23], the cost of a lost, unprotected laptop is roughly $49,000 [3]. To avoid data loss due to hardware failures, accidental deletion of data, or device theft/loss, enterprises and individuals have increased their use of data protection and recovery tools in the personal computing environment.

Cloud backup service has become a cost-effective choice for data protection of personal computing devices. In recent years, similar to cloud computing, cloud storage has received increasing attention from industry and academia as it offers virtually infinite storage resources that are available on demand and charged according to usage [4]. Since traditional backup services require data to be backed up to dedicated external drives, which can be inconvenient or costly to the users, data backup for personal computing has emerged to be a particularly attractive application for outsourcing to the 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 inflection point, and offers simple offsite storage, always a critical concern for data backup.

Data deduplication, an effective data compression approach that exploits data redundancy, partitions large data objects into smaller parts, called chunks, and represents and replaces these chunks by their fingerprints (i.e., generally a cryptographic hash of the chunk data) for the purpose of communication or storage efficiency. Depending on the location where redundant data is eliminated, deduplication can be categorized into source deduplication that applies data deduplication at the client site and target deduplication that eliminates redundant data at the backup server site. Since data backup for personal computing 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), source deduplication is obviously preferred to target deduplication due to the former's ability to significantly reduce the amount of data transferred over WAN with low communication bandwidth. However, data deduplication is a resource-intensive process, which entails the CPU-intensive hash calculations for 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 an

978-0-7695-4516-5/11 $26.00 © 2011 IEEE
DOI 10.1109/CLUSTER.2011.20

optimal tradeoff between deduplication effectiveness and deduplication overhead for personal computing devices with limited system resources.

In this paper, we propose AA-Dedupe, an application-aware source deduplication scheme, to achieve high deduplication efficiency based on a number of key observations drawn from our preliminary and experimental study of data deduplication in the personal computing environment. This study (detailed in Section II) motivates our AA-Dedupe design by the following four observations of deduplication for cloud backup services in the personal computing environment:

- O1: The majority of storage space is occupied by a small number of compressed files with low sub-file redundancy.
- O2: Static chunking (SC) [8] method can outperform content defined chunking (CDC) [15] in deduplication effectiveness for static application data and virtual machine images.
- O3: The computational overhead for deduplication is dominated by data capacity.
- O4: The amount of data shared among different types of applications is negligible.

These observations reveal a significant difference among different types of applications in the personal computing environment in terms of data redundancy, sensitivity to different chunking methods, independence in the deduplication process. Thus, the basic idea of AA-Dedupe is to effectively exploit this application difference and awareness by treating different types of applications differently and adaptively during the deduplication process to significantly improve the deduplication efficiency and reduce the overhead.

The main contributions of our paper include:

- A new metric, "bytes saved per second", is proposed to measure the efficiency of different deduplication schemes on the same platform.
- According to our observations on the application-oriented deduplication effectiveness, an intelligent deduplication scheme with application-aware index structure is presented to improve the deduplication efficiency for personal computing devices.
- To improve data transfer efficiency, a container management strategy is proposed to aggregate small data packet transfers into a single larger one for cloud storage.
- Our prototype implementation of AA-Dedupe and real dataset driven evaluations show that it outperforms the existing state-of-the-art source deduplication schemes in terms of backup window, power efficiency, and cost saving for the high deduplication efficiency in cloud backup services.

The remainder of this paper is organized as follows. In the next section, we provide the necessary background on source deduplication and conduct a preliminary quantitative study on data backup for personal computing in cloud storage to motivate our research. In Section III, we describe the system architecture of AA-Dedupe, and detail the design of an intelligent deduplication scheme with application-aware index structure. We evaluate AA-Dedupe on its prototype implementation with real datasets, by comparing it with the existing state-of-the-art schemes in terms of deduplication efficiency, backup window size, cloud storage cost and energy efficiency in Section IV. We discuss related work in Section V and conclude with remarks on future work in Section VI.

II. 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 research in the application-aware source deduplication for cloud backup services.

A. Cloud Backup

Cloud is a new business model wrapped around new technologies to reduce the cost of using IT resources [1]. It refers to both the applications delivered as services over the Internet and the hardware and systems software in data centers that provide such services. Recent interest in cloud storage has been driven by new offerings of storage resources that are attractive due to per-use pricing and elastic scalability, providing a significant advantage over the typical acquisition and deployment of equipment or infrastructure that was previously required [4]. In the personal computing environment, cloud backup, from an end user's perspective, is nothing but an unlimited amount of online storage space that is secure, inexpensive and highly available to backup data from personal computing devices. Different from traditional backup that requires dedicated and high bandwidth network connectivity between the client machines and the servers, cloud backup significantly relaxes this requirement and thus is suitable for environments with reasonable network connectivity but limited bandwidth and low data change rate in small dataset, such as the typical personal computing environment.

B. Source Deduplication

In source deduplication, elimination of duplicate data occurs close to where data is created, rather than where data is stored as in the case of target deduplication. Performing deduplication at the source can dramatically improve IT economics by minimizing storage requirements and network bandwidth consumption since the redundant data is eliminated prior to its traverse across the network to the target backup server [6]. Based on different deduplication granularities, source deduplication can be further divided into source file-level deduplication [26] and source chunk-level deduplication [12][24], where the former removes duplicate data at the file granularity with low duplicate elimination effectiveness and low computational overhead, while the latter removes the duplicate data at the sub-file (that is, chunk) level with high duplicate elimination effectiveness and high computational overhead. To achieve high effectiveness of deduplication, source chunk-level deduplication has become popular and represents state of the art. However, such fine-grained data deduplication is very expensive in terms of memory and processing especially on resource-constrained clients in the

personal computing environment. Therefore it is desirable to achieve an optimal tradeoff (i.e., efficiency) between data deduplication effectiveness and deduplication overhead for source deduplication in the personal computing environment. Moreover, source deduplication faces another challenge in the form of low data transfer efficiency, since post-deduplication data transfers from source to target are typically very small but must often traverse across a WAN. As the overhead of lower layer protocols can be high for small data transfers [19], source-side data aggregation strategies have become a necessary consideration in existing techniques[12][20].

C. *Motivational Observations*

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, through our preliminary experimental study. In what follows, we draw the following observations and analysis from this study to motivate our research.

Observation 1: *A disproportionally large percentage of storage space is occupied by a very small number of large files with very low sub-file redundancy.*

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

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 3 desktops and 7 laptops of 10 users and show the results in Fig. 1 and Fig. 2. We observe that about 61% of all files are smaller than 10KB, accounting for only 1.2% of the total storage capacity, and only 1.4% files are larger than 1MB but occupy 75% of the storage capacity. These results are consistent with previously published studies [8][9]. 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 the deduplication efficiency. The achieved data deduplication ratio (DR, i.e., the ratio of the amounts of data before and after deduplication) actually depends on which data is deduplicated and on the effectiveness and efficiency of the specific technologies used to perform capacity optimization. To verify the data redundancy observed from our intuitive scenarios, we carried out chunk-level deduplication using Static Chunking (SC) of 8KB chunk size and Content Defined Chunking (CDC) of 8KB average chunk size (Min: 2 KB, Max: 16 KB) after file-level deduplication in about 41 GB data of typical PC applications, respectively. From the statistics shown in Table 1, we observe that different applications or data types have different levels of data redundancy, where almost all data types with low chunk-level data redundancy are related to applications using compression, while files in these applications are of large average file sizes. It is similar to the results in [7]. In the datasets of 10 PCs mentioned above, compressed files larger than 1 MB occupy 53% storage space. This is consistent with the results in the published research [8] indicating that the deduplication ratio begins to decrease when the file size is larger than 8MB. For low sub-file data redundancy, 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. In the file-level deduplication process of compressed files, a weak hash function is sufficient to avoid hash collisions due to the very small number of large files in the personal computing environment.

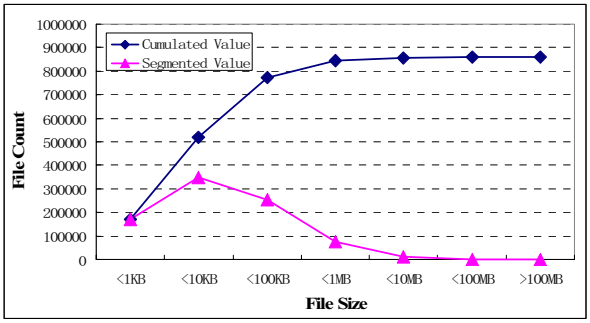

Fig. 1. File count in the PC datasets

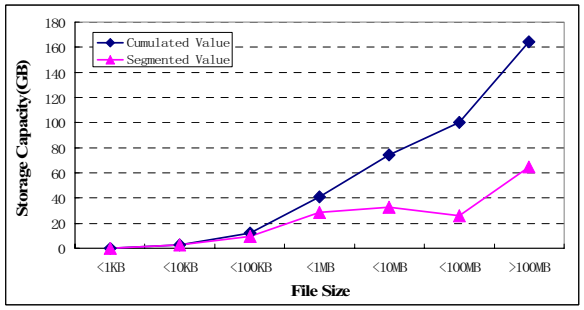

Fig. 2. Storage capacity in the PC datasets

TABLE I

CHUNK LEVEL DATA REDUNDANCY IN TYPICAL PC APPLICATIONS

File Type	Dataset Size(MB)	Mean File Size (B)	SC DR	CDC DR
AVI	2243	198 M	1.0002	1.0002
MP3	1410	5 M	1.001	1.002
ISO	1291	646 M	1.002	1.002
DMG	1032	86 M	1.004	1.004
RAR	1452	12 M	1.008	1.008
JPG	1797	2 M	1.009	1.009
PDF	910	403 K	1.015	1.014
EXE	400	298 K	1.063	1.062
VMDK	28473	312 M	1.286	1.168
DOC	550	180 K	1.231	1.234
TXT	906	615 K	1.232	1.259
PPT	320	977 K	1.275	1.3

Observation 2: *The amount of data shared among different types of applications is negligible.*

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

We compare the chunk fingerprints of datasets used in Observation 1 after intra-application deduplication using chunk-level deduplication with 8 KB chunk size in different applications, including TXT, DOC, PPT, PDF, JPG, DMG, MP3, ISO, EXE, AVI and RAR files, and find that only one 16 KB chunk is a duplicate. This comes from the difference in data content and format among these applications. As the data sharing among different applications is negligible, application-aware deduplication is poised to eliminate redundancy in each application independently and in parallel without reducing the effectiveness of data deduplication. 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 [13][15] while exposing higher index access parallelism.

Observation 3: *SC can outperform CDC on deduplication effectiveness for static application data or virtual machine disk images.*

Implication: *Despite of the fact that CDC-based deduplication is traditionally considered the best strategy in finding redundancy in data deduplication, SC-based methods may be preferred in deduplicating datasets of static applications or VM disk images.*

To find the sub-file redundancy, we can 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 [14] caused by data updates. In the datasets of static applications where data updating operations are infrequent, or some specific data format, like virtual machine disk images [22], we observe that SC-based deduplication can approach or even outperform CDC-based deduplication in deduplication effectiveness. Table 1 also shows that SC-based deduplication performs similar or even better than CDC-based deduplication on the effectiveness of data reduction for PDF, EXE and VMDK files because a large number of chunks are cut forcibly when reaching the maximum length in the CDC processing. This suggests that we can choose to deduplicate these kinds of files using SC-based methods in application-aware deduplication to improve the efficiency.

Observation 4: *The computational overhead of deduplication is determined by data capacity, not the granularity of deduplication.*

Implication: *The use of weaker hash functions for more coarse-grained chunks is the only way to reduce the computational overhead.*

To evaluate the computational overhead of typical hash functions, we measured the execution times of the Rabin hash, MD5 and SHA-1 hash algorithms on a laptop for Whole File Chunking (WFC)-based deduplication, which is a variant of CDC-based methods that uses an entire file as the basis for duplicate detection [8], and SC-based deduplication with 8KB fixed chunk size in a 60MB dataset. Rabin hash is a rolling hash function with lower computational overhead than cryptographic hash functions (e.g., SHA-1 and MD5), and we choose a 96-bit extended hash value to keep a low probability of hash collisions. The results are shown in Fig. 3. The total execution time of file-level deduplication (WFC) is almost the same as that of chunk-level deduplication (SC), since the bulk of the processing time is spent on the hash calculation itself and not on storing hash values. The insight obtained here is consistent to the findings reported in [10]. In other words, the use of weaker hash functions for more coarse-grained chunks is an effective way to reduce the computational overhead. The experimental results, shown in Fig. 4, compare the deduplication throughputs of three typical data chunking methods, WFC, SC and CDC, when applied with the three hash functions, Rabin hash, MDS and SHA-1, for chunk fingerprinting, respectively. The deduplication strategy based on simpler chunking schemes (e.g., WFC or SC) can achieve a higher throughput because of their lower metadata overhead, while deduplication strategies with weaker hash functions (e.g., Rabin hash) obtain a higher throughput because of their lower computational overhead. This suggests that we may employ WFC-based deduplication with the Rabin hash function for compressed files to enhance deduplication throughput, and SC-based deduplication using the MD5 hash function for static application data or virtual machine disk images. Nevertheless, we should still use the SHA-1 hash function for CDC-based deduplication because most of its computational overhead is on identifying the chunk boundaries instead of chunk fingerprinting.

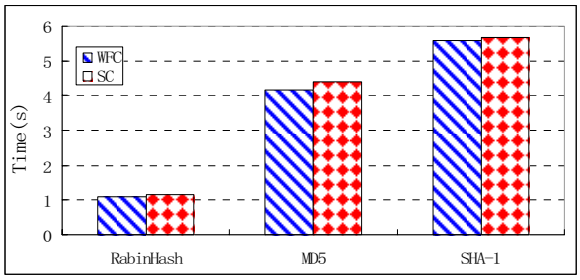

Fig. 3. Computational overhead of typical hash functions

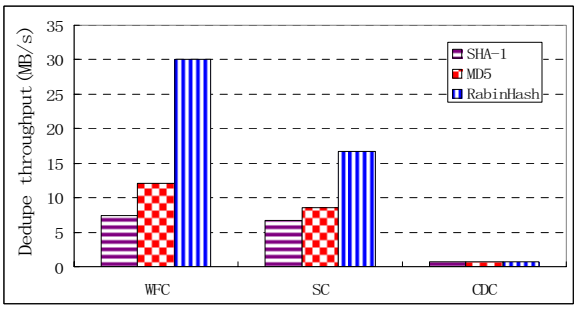

Fig. 4. Deduplication throughputs of different implementations

III. Design and Implementation of AA-Dedupe

To achieve high data transfer rates, cloud clients require significant processing to deduplicate data in cloud backup services, resulting in index size and performance challenges. Traditional approaches to meeting these challenges are to use methods that incur lower computational load but weaker data redundancy detection. Those approaches typically result in a reduction in data deduplication ratio. AA-Dedupe, motivated in part by our observations made in Section II, is designed to meet the requirement of deduplication efficiency. The main idea of AA-Dedupe is to reduce the computational overhead by employing an intelligent data chunking scheme and the adaptive use of hash functions based on application awareness, and 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.

A. An Architectural Overview

An architectural overview of AA-Dedupe is illustrated in Fig. 5, 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 looking up their hash values in an application-aware index that is stored in the local disk. If a match is found, the metadata for the file containing that chunk is updated to point to the location of the existing chunk. If there is no match, the new chunk is stored based on the container management in the cloud, the metadata for the associated file is updated to point to it and a new entry is added into the application-aware index to index the new chunk. We will now describe the deduplication process in more detail in the rest of this section.

B. File size filter

Most of the files in the PC dataset are tiny files that only occupy a negligibly small percentage of the storage capacity. As shown in our statistical evidences (In Section II), about 61% of all files are no larger than 10KB, accounting for only 1.2% of the total storage capacity of the dataset. To reduce the metadata overhead, AA-Dedupe filters out these tiny files (i.e., less than 10KB in file size) in file size filter before the deduplication process, an approach like SAM [11], and groups data from many tiny files together into larger units of about 1MB each in the container store, similar to Cumulus [12], to increase the data transfer efficiency over WAN.

C. Intelligent data chunking

Data chunking has a significant impact on the efficiency of deduplication. In general, the deduplication ratio is inversely proportional to the average chunk size. On the other hand, the average chunk size is also inversely proportional to the space overhead due to file metadata and chunk index in storage systems. This implies that a smaller average chunk size translates to a higher processing cost for data packets during transfer and a lower compression ratio for each chunk. The deduplication efficiency of data among different applications

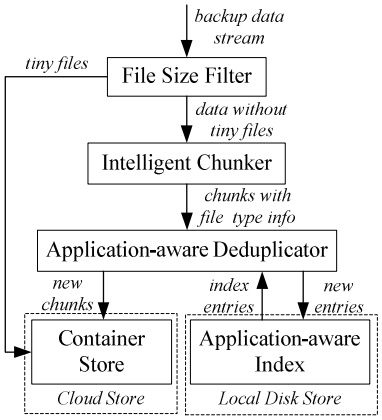

Fig. 5. An architectural overview of the AA-Dedupe design

differs greatly as we discussed in Section II. Depending on whether the file type is compressed or whether it is frequently edited, we divide files into three main categories: compressed files, static uncompressed files, and dynamic uncompressed files. To strike a better tradeoff between deduplication ratio and deduplication overhead, we deduplicate compressed files with WFC for its low sub-file redundancy, separate static uncompressed files into fix-sized chunks with SC, and break dynamic uncompressed files into variable-sized chunks using CDC based on the Rabin fingerprinting function to identify chunk boundaries.

D. Hash function selection in Deduplicator

Most of the computational overhead in deduplication is spent on the calculation of chunk fingerprinting for WFC and SC. Comparing with the overhead of identifying chunk boundaries based on the Rabin fingerprinting, the computational overhead for chunk fingerprinting in CDC is only a small part of deduplication overhead. Since compressed files with WFC have large chunk sizes, an extended 12B Rabin hash value is employed as chunk fingerprint to reduce the computational overhead, and it is justified by the fact that the probability of hash collisions in TB-scale PC datasets is smaller than the probability of hardware error by many orders of magnitude. For the same reason, a 16B MD5 hash value of chunks serves as chunk fingerprint of SC in static uncompressed files. We employ a 20B SHA-1 hash value to provide high data integrity for dynamic uncompressed files with only a slight increase in overhead because chunk fingerprint calculation is only a small part of the total computational overhead in CDC.

E. Application-aware index structure

As with traditional deduplication schemes, AA-Dedupe requires a chunk index, which maps each chunk hash to where that chunk is stored in the cloud storage, in order to determine which chunks have already been stored. If there is a match in the index, the incoming chunk contains redundant data and can be deduplicated to avoid transmitting it; if not, the chunk needs to be added to the cloud storage and its hash and metadata inserted into the index. The metadata contains the

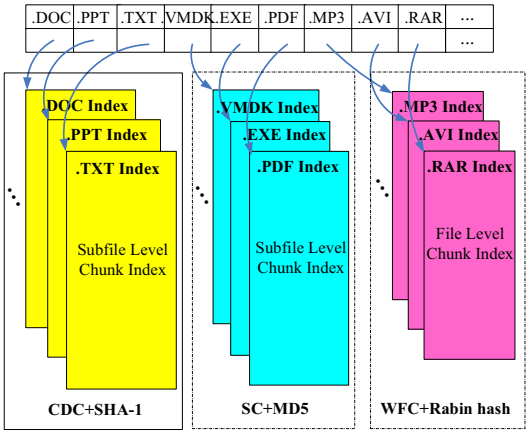

Fig. 6. Application-aware index structure

information such as chunk length and location. In AA-Dedupe, the selection for the proper chunking methods and hash functions in deduplication is entirely based on file type. An application-aware index structure for AA-Dedupe is constructed as shown in Fig. 6. According to the accompanied file type information, the incoming chunk is directed to the chunk index with the same file type. Comparing with traditional deduplication mechanisms, AA-Dedupe can achieve high deduplication throughput by looking up chunk fingerprints concurrently in small indices classified by applications rather than the single full, unclassified index. Furthermore, a periodical data synchronization scheme is also proposed in AA-Dedupe to backup the application-aware index in the cloud storage to protect the data integrity of the PC backup datasets.

F. *Container management*

Because large sequential writes are transformed into small random writes by the deduplication process, AA-Dedupe will often group data from many smaller files and chunks into larger units called container before these data are transferred over WAN. A container is a self-describing data structure in that a metadata section includes the chunk descriptors for the stored chunks. An open chunk container is maintained for each incoming backup data stream, 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 (e.g., 1MB), 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 [20] 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 scheme is also adopted in the state-of-the-art schemes such as DDFS [15] and Sparse Indexing [20] to improve manageability and performance. Aggregation of data produces larger files for the 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 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 in size.

IV. EVALUATIONS

We have built a prototype of AA-Dedupe in approximately 2300 lines of C++ code and fed the real-world datasets to evaluate the use of cloud backup services in a personal computing environment. Our goal in this evaluation is to answer the following questions:

- How effective is AA-Dedupe in striking a reasonable tradeoff between the deduplication effectiveness and deduplication overhead with real world datasets?
- How well does AA-Dedupe work in terms of shortening the backup window?
- What are the monetary costs for cloud backup services based on AA-Dedupe in the given datasets?
- How much energy does AA-Dedupe save compared with traditional source deduplication based cloud backup services?

The following evaluation subsections will answer these questions, beginning with a description of the experiment platform with PC backup datasets we use as inputs to the experiments and the evaluation metric we propose to quantify deduplication efficiency.

A. *Experiment Platform and Datasets*

Our experiments were performed on a MacBook Pro with 2.53 GHz Intel Core 2 Duo processor, 4 GB RAM, and one 250 GB SATA disk, and it can reach about 500KB/s average upload speed and 1MB/s average download speed with the AirPort Extreme 802.11g wireless card. We use backup datasets in the *user* directory of one of the author's PCs as workloads to drive our evaluations, and model the use of remote backup. There are 10 consecutive weekly full backups in the workloads with a total of 351GB data consisting of 68,972 files in 12 applications.

We compare AA-Dedupe against a number of state-of-the-art schemes, including Jungle Disk [25], a file incremental cloud backup scheme, BackupPC [26], a source file-level deduplication based cloud backup, Avamar [24], a source chunk-level deduplication based cloud backup, and SAM [11], a hybrid source deduplication based cloud backup scheme. The comparisons are based on measures of deduplication efficiency, backup window size, cloud storage cost and energy consumption. In all experiments, we choose a fixed chunk size of 8KB for SC-based deduplication strategies, and an expected chunk size of 8 KB (with a 2 KB minimum and 16 KB maximum) for CDC-based method with Rabin fingerprinting that uses a 48-byte fixed sliding window size and 1-byte step size. Table II shows some parameters to model deduplication efficiency, backup window size and cloud cost for cloud backup services.

TABLE II

PARAMETERS FOR CLOUD BACKUP SERVICES

DE	Dedupe Efficiency	SC	Saved Capacity
DT	Dedupe Throughput	DS	Dataset Size
NT	Network Throughput	DR	Dedupe Ratio
BWS	Backup Window Size	SP	Storage Price
OP	Operation Price	TP	Transfer Price
OC	Operation Count	CC	Cloud Cost

B. Metric for Deduplication Efficiency

It is well understood that the deduplication efficiency is proportional to deduplication effectiveness that can be defined by deduplication ratio, and inversely proportional to deduplication overhead that can be measured by deduplication throughput. 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", to measure the efficiency of different deduplication schemes in the same platform, similar to the metric of "bytes saved per cycle" proposed in [16] that estimates the computational overhead. "bytes saved per second" is expressed as:

$$DE = \frac{SC \times DT}{DS} = (1 - \frac{1}{DR}) \times DT .$$

C. Deduplication efficiency

Deduplication effectiveness is very important for both cloud backup providers and users. Providers expect less data stored in their data centers to reduce data storage and management costs, whereas users prefer transferring less data for shorter backup time and lower storage cost. Our experimental results present both the cumulative cloud storage capacity required of the providers and the deduplication efficiency of each backup session for individual users respectively. Fig. 7 compares the cumulative storage capacity of the five cloud backup schemes. The four source deduplication based backup schemes can outperform the incremental backup scheme that is implemented in Jungle Disk, especially for those based on fine-grained methods (e.g., Avamar) or semantic-aware deduplication scheme SAM. AA-Dedupe achieves similar or better space efficiency than Avamar and SAM. The high effectiveness of data deduplication of the fine-grained deduplication schemes comes at a significant overhead that throttles the system throughput. We present a comparison for deduplication efficiency of the five cloud backup schemes in Fig. 8, and employ our proposed new metric of "bytes saved per second" to measure the efficiency of different deduplication approaches in the same platform. AA-Dedupe performs much better than other backup schemes on deduplication efficiency with a low overhead. This significant advantage of AA-Dedupe is primarily attributed to its application-awareness in the deduplication process. We observe that the deduplication efficiency in AA-Dedupe is 2 times that of BackupPC, 5 times that of SAM and 7 times that of Avamar on average, and it can also significantly improve the space saving for some applications in backup sessions 3, 4 and 10 that other approaches fail to achieve.

D. 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. For the four source deduplication schemes considered in this study, the backup window consists of two parts: data deduplication time and data transfer time. Because of our pipelined design for the deduplication processes and the

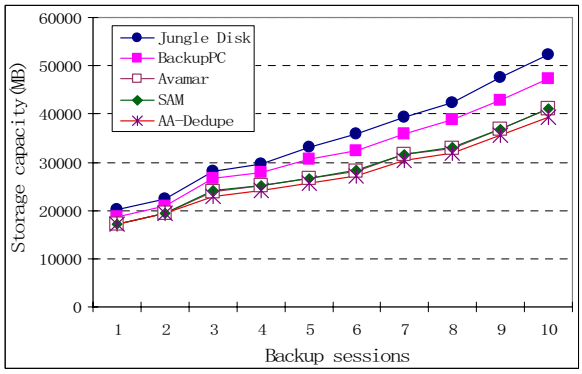

Fig. 7. Cloud storage capacity requirement

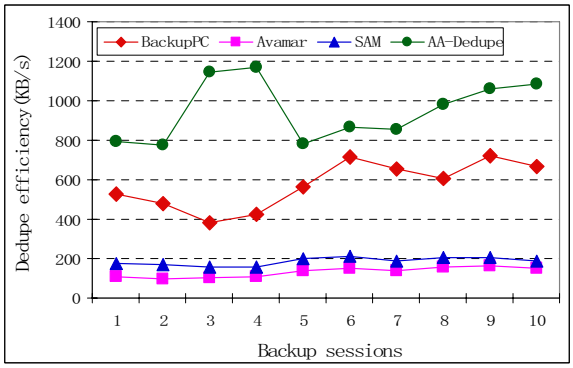

Fig. 8. Data deduplication efficiency of the backup dataset

data transfer operations, the backup window size of each backup session can be calculated based on the expression below:

$$BWS = DS \times Max(\frac{1}{DT}, \frac{1}{DR \times NT}).$$

In our experimental results, shown in Fig. 9, Avamar performs the worst in backup throughput due to the high system overhead in fine-grained deduplication, which is even worse than the full backup method in our test environment. In other schemes, the backup window size is always determined by the data transfer capacity after deduplication due to the low upload bandwidth in WAN. AA-Dedupe consistently performs the best among the five cloud backup schemes owing to its high deduplication efficiency. We observe that the backup window size of AA-Dedupe is shortened from other schemes by about 10%-32% in our test environment.

E. Cloud Cost

Cloud backup as a service, however, comes at a price. In this subsection, we calculate the monetary costs for our workload models. To price cloud-based backup services attractively requires minimizing the capital costs of data center 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 April 2011, these prices are (in US dollars): $0.14 per GB · month for storage, $0.10 per GB for upload data transfer and $0.01 per 1000 upload requests. And the cost of cloud backup services can be modelled as:

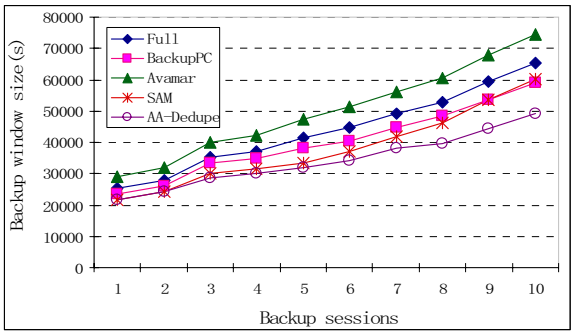

Fig. 9. Backup window size of 5 backup sessions

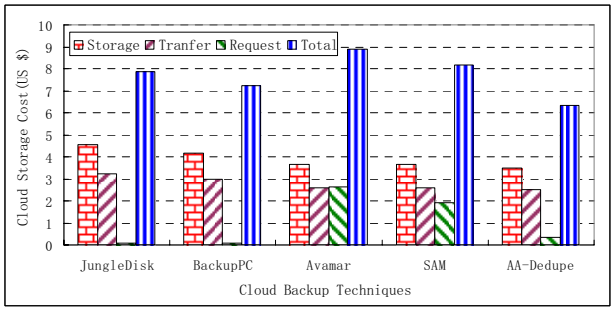

Fig. 10. The cost of cloud backup techniques

$$CC = \frac{DS}{DR} \times (SP + TP) + OC \times OP \,.$$

Backing up the deduplicated datasets directly to Amazon S3 can be very slow and costly due to the large number of I/O operations and the way Amazon S3 charges for the uploads. We estimate the cloud cost of our test datasets in one month, as shown in Fig. 10. Comparing with the more space-efficient schemes such as Avamar and SAM, file-granularity 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. AA-Dedupe can reduce the request cost significantly by packing several KB-sized tiny files and chunks into 1MB containers before sending them to the cloud. We observe that the cloud cost of AA-Dedupe is lower than those of other schemes by about 12%-29% for our backup datasets.

F. Energy Efficiency

Energy efficiency has become a critical issue and its importance seems to be more pronounced in the personal computing environment, especially for mobile devices like laptops, tablets, smartphones and PDAs due to the limited energy in battery. As the deduplication process is a compute-intensive application, energy consumption becomes a challenge for source deduplication based cloud backup in the personal computing environment. In our experiment, we compare the power consumptions of the four source deduplication based cloud backup schemes during the deduplication process, measured by an electricity usage monitor on the whole PC. Fig. 11 shows the energy consumptions of the four cloud backup schemes as a function of backup sessions. Existing highly space-efficient approaches, such as Avamar and SAM, incur high-level power consumptions due to their

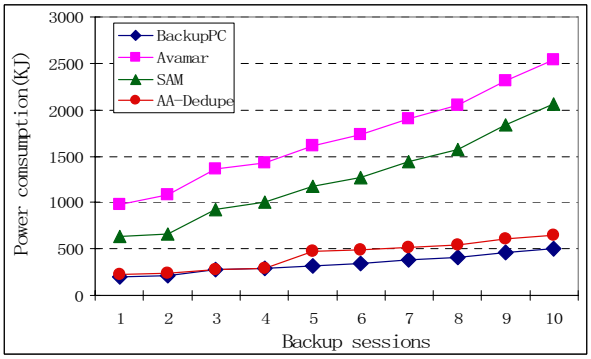

Fig. 11. Power consumption of source deduplication schemes

significant computational overhead during the deduplication process. AA-Dedupe incurs only one fourth of the power consumption of Avamar and one third of that of SAM by adaptively using weaker hash functions in deduplication.

V. Related Work

In cloud backup, performing deduplication at the source can dramatically improve IT economics by minimizing storage requirements, backup windows and network bandwidth utilization because redundant data is eliminated prior to its traverse across the network to the backup server. Comparing with traditional incremental cloud backup schemes, like Jungle Disk [25], source deduplication can achieve higher space efficiency to reduce cloud storage cost, and the deduplication efficiency becomes critical for cloud clients in the personal computing environment due to its limited system resources. BackupPC [26] performs deduplication at the file level to achieve low lookup overhead by reducing metadata, but at the cost of space efficiency. To achieve high space efficiency, EMC Avamar [24] applies CDC-based chunk-level deduplication with high computational overhead and lookup overhead, and ADMAD [17] improves redundancy detection by application-specific chunking methods that exploit the knowledge about concrete file formats. Intuitively, different applications and data types tend to have different levels of data redundancy. SAM [11] designs a hybrid source deduplication scheme, combining file-level and chunk-level deduplication schemes based on file semantics to mitigate lookup overhead by reducing metadata, and has been shown to have very little negative impact on deduplication effectiveness. In contrast, AA-Dedupe improves deduplication efficiency significantly by intelligent data chunking methods with application awareness.

In additional to the granularity of data chunking, the search speed of duplicate data and the computational complexity of hash functions all play important roles in the deduplication efficiency. Unlike Avamar, Cumulus [12] aggregates data from small files for remote storage, and limits the search for unmodified data to the chunks in the previous versions of the file to reduce the lookup overhead in deduplication, but leaves open the challenge of how to identify file versions. AA-Dedupe is inspired by Cumulus, but exploits application-awareness by limiting the search for redundant data to the

chunks within the same kind of applications specified by the file format information. Unlike the traditional source-deduplication schemes, EndRE [5] employs a weak finger-printing scheme rather than strong cryptographic hash such as SHA-1 in redundancy elimination to improve the deduplication efficiency for mobile smartphone. AA-Dedupe differs from EndRE in that it adopts weak hash functions for coarse-grained data to reduce the computational overhead and employs strong hash functions for fine-grained data to avoid hash collisions.

VI. Conclusions and Future Work

Motivated by the key observations drawn from our preliminary experimental study, we propose AA-Dedupe, an application-aware source-deduplication scheme for cloud backup in the personal computing environment to improve deduplication efficiency. An intelligent deduplication strategy in AA-Dedupe is designed to exploit file semantics to minimize computational overhead with negligible loss in deduplication effectiveness. The novel data structure in AA-Dedupe, 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, AA-Dedupe is shown to improve the deduplication efficiency of the state-of-the-art source-deduplication approaches by a factor of 2-7, and shorten the backup window size by 10%-32%, improve power-efficiency by a factor of 3-4, and save 12%-29% cloud cost for the cloud backup service. As a direction of future work, we plan to investigate the secure deduplication issue in cloud backup services of the personal computing environment and further explore and exploit index lookup parallelism availed by the application-aware index structure of AA-Dedupe in a multi-core or many-core system.

Acknowledgments

We thank the anonymous reviewers and other members in ADSL of CSE in UNL for their helpful suggestions. This research was partially supported by the Program for New Century Excellent Talents in University of China under Grant No. NCET-08-0145, the 973 Program of China under Grant No. 2011CB302301,the National Natural Science Foundation of China under Grants No. 60736013, 61025009, 60903040 and 61070198, 863 Program 2009AA01A402, and the US NSF under Grants NSF-IIS-0916859, NSF-CCF-0937993 and NSF-CNS-1016609.

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, "Above the Clouds: A Berkeley View of Cloud Computing," UC Berkeley Reliable Adaptive Distributed Systems Laboratory. Tech. rep. Feb. 2009.

[2] "IDC: Digital Data to Double Every 18 Months," *Information Management Journal*, Sep. 2009, vol. 43/5 Docstoc page 20.

[3] L. Ponemon, "The Cost of a Lost Laptop," Intel Corporation, Apr. 22, 2009.

[4] "Cloud Storage for Cloud Computing," SNIA: Advancing storage & information technology, Sep. 2009. www.snia.org/cloud/CloudStorageForCloudComputing.pdf

[5] B. Agarwal, A. Akella, A. Anand, A. Balachandran, P. Chitnis, C. Muthukrishnan, R. Ramjee and G. Varghese. "EndRE: An End-System Redundancy Elimination Service for Enterprises," in *Proceedings of the Second Symposium on Networked Systems Design and Implementation (NSDI '10),* 2010, pp. 419-432.

[6] L. D. Bois and R. Amatruda, "Backup and Recovery: Accelerating Efficiency and Driving Down IT Costs Using Data Deduplication," EMC Corporation, Feb. 2010.

[7] D. T. Meyer and W. J. Bolosky, "A Study of Practical Deduplication," in *Proceedings of the 9th USENIX Conference on File and Storage Technologies (FAST'11),* 2011, pp. 1-14.

[8] D. Meister and A. Brinkmann, "Multi-level comparison of data deduplication in a backup scenario," in *Proceedings of the 2nd Annual International Systems and Storage Conference (SYSTOR'09),* Haifa, Israel: ACM, 2009.

[9] N. Agrawal, W. J. Bolosky, J. R. Douceur, and J. R. Lorch, "A Five-Year Study of File-System Metadata," in *Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST'07)*, Feb. 2007, pp. 31-45.

[10] C. Dubnicki, K. Lichota, E. Kruus and C. Ungureanu, "Methods and systems for data management using multiple selection criteria," United States Patent 7844581, Nov. 30, 2010.

[11] Y. Tan, H. Jiang, D. Feng, L. Tian, Z. Yan and G. Zhou, "SAM: A Semantic-Aware Multi-tiered Source De-duplication Framework for Cloud Backup," in *Proceedings of the 39th International Conference on Parallel Processing (ICPP'10),* 2010, pp. 614 - 623.

[12] M. Vrable, S. Savage and G. M. Voelker, "Cumulus: Filesystem Backup to the Cloud," in *Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST'09),* Feb. 2009, pp. 225–238.

[13] D. Bhagwat, K. Eshghi, D. D. Long and M. Lillibridge, "Extreme Binning: Scalable, Parallel Deduplication for Chunkbased File Backup," HP Laboratories, Tech. Rep. HPL-2009-10R2, Sep. 2009.

[14] K. Eshghi, "A framework for analyzing and improving content based chunking algorithms," Tech. Rep. HPL-2005-30 (R.1), Hewlett Packard Laboratories, Palo Alto, 2005.

[15] B. Zhu, K. Li, and H. Patterson, "Avoiding the disk bottleneck in the Data Domain deduplication file system," in *Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST'08),* Feb. 2008, pp. 269-282.

[16] P. Kulkarni, F. Douglis, J. Lavoie, and J. M. Tracey, "Redundancy elimination within large collections of files," in *Proceedings of the annual conference on USENIX Annual Technical Conference (ATC'04),* 2004, pp. 59-72.

[17] 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 *Proceedings of the 5th IEEE International Workshop on Storage Network Architecture and Parallel I/Os (SNAPI'08),* 2008, pp.29-35.

[18] N. Xiao, Z. Chen, F. Liu, M. Lai, L. An, "P3Stor: A parallel, durable flash-based SSD for enterprise-scale storage systems," Science China Information Sciences, vol. 54, No. 6, Jun. 2011, pp.1129-1141. DOI: 10.1007/s11432-011-4266-z.

[19] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula and D. Estrin, "A First Look at Traffic on Smartphones," in *Proceedings of the 10th annual conference on Internet measurement (IMC'10),* Nov.2010, pp. 281-287.

[20] M. Lillibridge, K. Eshghi, D. Bhagwat, V. Deolalikar, G. Trezise and P. Camble, "Sparse Indexing: Large Scale, Inline Deduplication Using Sampling and Locality," in *Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST'09),* 2009, pp. 111-123.

[21] P. Anderson and L. Zhang, "Fast and Secure Laptop Backups with Encrypted De-duplication," in *Proceedings of the 24th international conference on Large Installation System Administration (LISA'10),* 2010, pp. 29-40.

[22] VMware Inc. "Virtual disk format." http://www.vmware.com/interfaces/vmdk.html, Nov. 2007.

[23] "Data Loss Statistics," http://www.bostoncomputing.net/consultation/databackup/statistics/

[24] EMC Avamar. http://www.emc.com/avamar.

[25] Jungle disk. http://www.jungledisk.com/.

[26] BackupPC. http://backuppc.sourceforge.net/.