A Long-Term User-Centric Analysis of Deduplication Patterns

Zhen Sun,[*][†] Geoff Kuenning,[‡] Sonam Mandal,[†] Philip Shilane,[§] Vasily Tarasov,[¶] Nong Xiao,[*][‖]and Erez Zadok[†]
[*]State Key Laboratory of High Performance Computing, College of Computer, National University of Defense Technology, China.
[†]Stony Brook University. [‡]Harvey Mudd College.
[§]EMC Corporation. [¶] IBM Research–Almaden.
[‖]School of Data and Computer Science, Sun Yat-sen University, China.
Email: sunzhen@nudt.edu.cn, geoff@cs.hmc.edu, somandal@cs.stonybrook.edu, philip.shilane@emc.com,
vtarasov@us.ibm.com, xiaonong@nudt.edu.cn, ezk@cs.stonybrook.edu

Abstract— **Deduplication has become essential in disk-based backup systems, but there have been few long-term studies of backup workloads. Most past studies either were of a small static snapshot or covered only a short period that was not representative of how a backup system evolves over time. For this paper, we collected 21 months of data from a shared user file system; 33 users and over 4,000 snapshots are covered. We analyzed the data set for a variety of essential characteristics. However, our primary focus was individual user data. Despite apparently similar roles and behavior in all of our users, we found significant differences in their deduplication ratios. Moreover, the data that some users share with others had a much higher deduplication ratio than average. We analyze this behavior and make recommendations for future deduplication systems design.**

I. Introduction

The explosive growth of data in recent years [28] has made deduplication a hot topic. Deduplication systems split data into chunks and then use hashes to identify and eliminate redundant chunks. This approach has proven highly effective in saving space, especially in backup storage [35].

Many researchers have analyzed data sets from various environments, such as disk [32] and tape [11] backup, primary [5], [15], [22], [26] and archival storage [14], and HPC centers [25]. By understanding such data sets' characteristics, we can design more efficient storage systems [8], [9], [29]. However, data sets may vary significantly across different environments (e.g., whole-file chunking efficiencies range between 20% and 87% compared to sub-file chunking [5], [25], [32]). As a result, conclusions drawn from only few data sets cannot be used to guide the design of an efficient deduplication system. Thus, new, large-scale studies using different types of data sets and investigating new metrics are desirable.

High redundancies in backups [12] make deduplication an integral part of backup systems. The space savings of deduplication made the transition from tape-based to disk-based backup systems possible [35]. However, our understanding of these systems' real-life performance is still poor as there are few long-term studies of large backup data sets: most prior studies draw conclusions based on the entire data set, whereas we show that studying per-user trends is valuable and produces surprising results.

In this paper, we first introduce the data set we have been collecting and releasing publicly (21 months of daily snapshots taken over a 2.5-year period), and the tools we developed. Our data set has longer duration than previous studies, which is important for investigating realistic, long-term trends.

We then present an analysis of this data set, with sometimes unexpected results. For example, we found that because of the size of the chunk index itself, smaller chunk sizes are not always better at saving space. However, we found that whole-file chunking is much worse than sub-file chunking, because larger files tend to dominate space usage and have a small deduplication ratio (defined as the logical storage space divided by the physical storage space after deduplication).

Next, we studied the data set from the users' point of view. Given that our users were largely similar in their background, behavior, and job function, we found and investigated three surprising results: **(1)** The deduplication ratios of each user's own data set varied significantly, and their sensitivity to chunking size was also different. This suggests that even similar users behave quite differently, which should be accounted for in future deduplication systems. **(2)** Deduplication ratios across users ranged widely, but in combination with other information, can help us group users together to improve the effectiveness of clustered deduplication systems. **(3)** The data that users share with each other had a higher deduplication ratio than average, and the duplicate data tended to be hot. This knowledge can benefit the caching and prefetching components of deduplication systems.

The rest of this paper is organized as follows: Section II provides background and related work. In Section III, we give a brief introduction of our data set and the tools that collected these snapshots. We discuss our results in Section IV, and Section V concludes.

II. Background and Related Work

Now that deduplication has become essential in both backup and primary storage, researchers have tried to improve its performance, e.g. by enhancing storage efficiency [13], [33], enabling scaling [1], [3], [4], [6], [10], [19], [30], and resolving bottlenecks [2], [8], [16], [18], [23], [35]. Among these studies, those that analyzed real-world data played important roles because they helped improve design decisions based on data characteristics [5], [15], [25], [27], [32].

A study of Microsoft primary storage [26] collected data from 857 users across 4 weeks. They found that whole-file chunking works well in their specific environment, and fragmentation is not a serious problem when a defragmenter runs weekly. Follow-on work by Microsoft [5] implemented a scalable deduplication system in Windows Server 2012, based on the findings from primary deduplication analysis. Wallace *et al.* [32] investigated EMC's Data Domain backup systems, showing that the backup characteristics vary significantly with primary workloads. Backup workloads tend to have a high churn rate, a lot of stream locality, a high demand for writing, and high redundancy.

Other studies focused on file analysis [22]; conclusions about file type and size distributions, and about deduplication and compression ratios, have helped improve the design of deduplication systems. Data sets from different storage environments, such as HPC [25] and VDI [15], have also been collected and analyzed, and many of their conclusions are also applicable to other storage systems.

Most of the aforementioned data sets are static snapshots of storage systems or only cover a short time period. Our work extends these studies by using a long-term workload that extends from March 2012 to November 2014. Because backup systems are designed for long-term storage, our results—based on a long history—offer new valuable insights.

III. Methodology

We now describe the tools we developed, the data set itself, and the limitations of this study. Both the tools and the data-set have been publicly available online [7]; the data-set will be updated periodically as we continue to collect snapshots. Our data-set has already been used in a number of research papers [17], [20], [21], [31], [34].

A. Tools

To conduct our analysis, we developed tools that collect and parse file-system snapshots. *Fs-hasher* scans a file system and collects snapshots that contain both file system data and rich meta data. It does not collect actual file content; instead, it chunks each file and collects the hashes, similar to real-world deduplication systems. *Fs-hasher* supports both fixed and variable chunking. Users can specify various parameters (e.g., expected average chunk size for variable chunking) and hash functions (e.g., MURMUR, MD5, and SHA1) when collecting snapshots. Hash files collected by *Fs-hasher* can be parsed by a purpose-written program or by *Hf-stat*, which prints hash files in a human-readable and post-processable format. *Hf-stat* provides options to control and filter its output. *Fs-hasher* does not anonymize the snapshots; the released data set, however, is anonymized during post-processing.

B. Data set

The *Homes* data set contains almost-daily snapshots of our users' home directories on a shared file system; we collected one daily snapshot per user for a total of over 4,000 snapshots. The users are all Linux systems software developers who work on several joint projects. The snapshots have been collected since March 2012, with collection planned to continue into the foreseeable future, and with periodic releases of updated data sets. We used variable chunking with 7 average chunk sizes (2–128KB) and whole-file chunking (WFC) to allow a detailed comparison of performance and overhead among common chunking methods. To speed the snapshot collection procedure and reduce the data-set size, we chose a 48-bit MD5-based hash function. We used a short hash, but our collision rate is still acceptable for research purposes [32], because even with 2KB chunking the number of unique hashes is about 10^8, so the expected number of collisions in our data set is only about 4,000 (or 0.004%). Although this number would be unacceptable in commercial deduplication systems, it has a negligible impact on our findings. For larger chunk sizes, our collision rate was even lower.

Previous data sets have used static snapshots or have covered a short time period. Our data set contains 21 months of daily snapshots from a 2.5-year period. It includes data on 33 users, allowing us to analyze deduplication on a per-user basis. Although we did not collect the full file content, the rich meta data and extensive hashes make it possible to conduct a wide range of studies. For example, by examining modification and access times we can simulate various backup methods, including full and incremental backups at different periods. Table I shows the details of the *Homes* data set.

Data set	Homes
Total size	456TB
Start and end time	03/09/2012–11/23/2014
Number of users	33
Number of snapshots	4,181 dailies (about 21 months)
Chunking method	Content-defined chunking (CDC), Whole-file chunking (WFC)
Average chunking size	2, 4, 8, 16, 32, 64, and 128KB
Hash function	48-bit MD5
Number of files	1.3×10^8
Number of logical chunks	1.9×10^{10} (2KB chunk size) 4.0×10^8 (128KB chunking)
Number of unique chunks	9.8×10^7 (2KB chunk size) 3.3×10^6 (128KB chunking)
Meta-data included	File pathname, size, atime, ctime, mtime, UID, GID, permission bits, device ID, inode number

TABLE I. Features of the Homes data set.

C. Limitations

Since storing full file contents would consume too much space, we record only hashes and meta data. Thus, we are unable to analyze content-based properties, such as compression performance. For other properties like backup throughput, we can calculate simulated results from other metrics, such as the cache miss ratio and average disk I/O latency.

Although we attempted to collect daily snapshots for 2.5 years, some periods were missed, mainly due to major power outages (severe weather), hardware failures, and long breaks when most data remain unchanged due to user absence. Still, we believe that our data set is sufficiently large and long-term to serve as the basis of a valuable study.

IV. Results of Analysis

We begin by describing results from analyzing the entire data set: deduplication ratio analysis (Section IV-A) and file-based analysis (Section IV-B). Then, in Section IV-C, we present the results of the user-centric analysis.

A. *Deduplication Ratios*

One of the key measures of a deduplication system is the *deduplication ratio*, defined as the size of original data set divided by the size of what is physically stored on the media. In *Homes*, the variety of meta-data that we collected makes it possible to simulate a number of realistic backup situations. For this paper we chose three typical backup strategies: *Full*, *Incremental*, and *Weekly-Full* (a full backup each Saturday and incrementals for the rest of the week). Since full snapshots were collected daily, they inherently represent full backups. For incremental and weekly-full backups we needed to detect newly added and modified files. By comparing two consecutive snapshots we identified whether a file was newly added. By checking the *mtime* we determined the files that were modified since the previous snapshot. Table II shows the deduplication ratios when using different backup strategies and chunk sizes.

Chunk size	Full backup	Incremental backup	Weekly-full backup
2KB	**218.5**	**13.6**	**42.8**
4KB	197.0	12.6	39.4
8KB	181.9	11.7	36.5
16KB	167.4	10.7	33.6
32KB	153.3	9.8	30.8
64KB	139.1	8.9	27.9
128KB	128.0	8.2	25.7
WFC	16.4	1.1	2.3

TABLE II. RAW DEDUPLICATION RATIOS FOR VARIOUS CHUNKING METHODS AND BACKUP STRATEGIES. WFC STANDS FOR WHOLE-FILE CHUNKING.

The deduplication ratios in our data set are relatively high compared with other studies, primarily because for the purpose of the long-term study, snapshots are accumulated daily and never deleted. To cap storage costs, many deployed backup systems periodically delete snapshots, yet there is a clear benefit in preserving as many historical snapshots as possible.

The numbers shown in Table II are raw deduplication ratios. However, a smaller chunk size implies higher meta-data overhead, since more hashes must be stored. To calculate the efficiency of various chunking sizes, we adopted the approach proposed by Wallace et al. [32]. Suppose L is the size before deduplication, P is the raw data size afterwards, and f is the size of each chunk's meta-data divided by the chunk size. Then the raw deduplication ratio is $D = L/P$. The meta-data size is $f \times (L + P)$: $f \times L$ is the size of a file's *recipe* (needed to reconstruct its original contents) and $f \times P$ is the size of the hash index. Thus the overall stored size is $P + f \times (L + P)$. Based on this formula, the practical deduplication ratio including all costs, D', is:

$$D' = \frac{L}{P + f \times (L + P)} = \frac{D}{1 + f \times (D + 1)} \qquad (1)$$

Although our snapshots used a 6-byte (48-bit) MD5-based hash function, in this analysis we assume 30 bytes per chunk to show what would happen in a real deduplication system that stores longer hashes and other information such as chunk pointers and file recipes. This value is chosen from the middle of a range of common meta-data sizes [24], [32].

Chunk size	Full backup	Incremental backup	Weekly-full backup
2KB	50.9	11.1	25.8
4KB	79.3	**11.4**	30.2
8KB	107.9	11.1	**32.0**
16KB	127.2	10.5	31.6
32KB	**133.9**	9.7	29.9
64KB	130.5	8.9	27.6
128KB	124.3	8.2	25.7

TABLE III. EFFECTIVE DEDUPLICATION RATIOS AFTER ACCOUNTING FOR META-DATA OVERHEADS.

Table III shows the effective deduplication ratio for each strategy. We can see that 32KB chunking performed best for full backups; 4KB for incrementals; and 8KB for weekly-full ones. This suggests that the best chunk size may depend on the backup strategy and frequency. Table III also shows that as chunk sizes increase, a decrease in deduplication ratio is not obvious or guaranteed. A larger chunk size can reduce meta-data size significantly enough to compensate for missing some duplicates. Moreover, reducing the amount of meta-data also reduces the number of I/Os to the chunk index.

B. *File-Based Analysis*

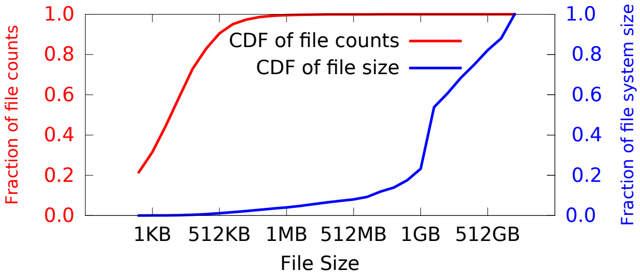

Fig. 1. Distribution of file sizes (blue) and counts (red).

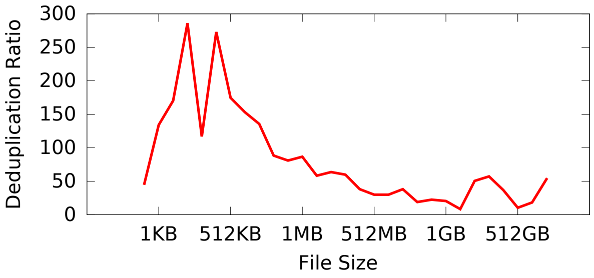

Fig. 2. Whole-file chunking deduplication ratio of files of different sizes.

The performances of whole-file chunking varies widely based on the characteristics of different data sets [26], [32], but it is clearly not a good choice in *Homes*, as shown in Table II. The reason can be found in Figures 1 and 2. Figure 1 shows the distribution of files in size and count. The upper line of Figure 1 shows that more than 99% of the files are smaller than 1MB, but the lower line demonstrates that in aggregate, these small files consume less than 4% of total space. When we grouped files by size and calculated a separate whole-file deduplication ratio for each group (Figure 2), we observed

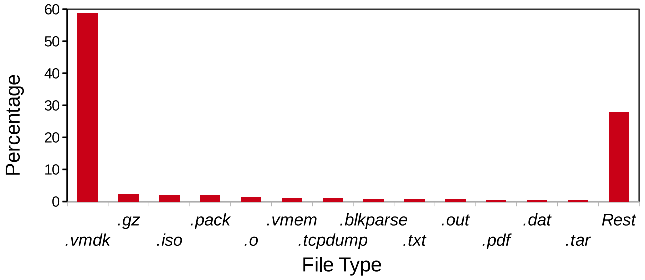

Fig. 3. Distribution of total storage by file type before deduplication. The *Rest* bar comprises the file types that independently contributed less than 0.4% of total storage each.

a large variance. Files between 2KB and 256KB deduplicate well—all higher than 100× and the highest at about 290×. In contrast, the average deduplication ratio is smaller than 50× for files larger than 1MB. Thus we can draw a conclusion that the total size of our data set is mainly occupied by large files that have a low deduplication ratio. As a result, whole-file chunking is ineffective at saving space in our data-set.

Figure 3 shows the percentage of total space occupied by various common file types; we can see that virtual machine images (vmdk) consume nearly 60% of the entire data size.

Figure 4 shows the raw deduplication ratio of different file types at various chunking sizes; here we selected the file types that occupied the most disk space. We can see that the deduplication ratio of different file types varies significantly: using 8KB chunking, the ratio ranges from 50 (`.vdi`) to 877 (`.h`). In addition, file types also have different sensitivities to the chunking size. For some file types, increasing the chunking size from 8KB to 128KB leads to a large drop in deduplication ratio, e.g., drop by 43% (`.disk`), 34% (`.vmdk`), and 26% (`.vmem`). However, for most file types this decrease is not significant, with some types showing no drop at all (e.g., `.gz`, `.mp3`, `.zip`). There are two reasons for this phenomenon. First, some file types such as compressed files, MP3, and video might be rarely changed in the software development environment, so the chunking size will not have a great effect on the deduplication ratio. Second, when collecting snapshots we did not combine multiple small files into a single larger chunk, which means that when a file is smaller than the chunking size, the file itself is a chunk. As a result, for file types that tend to be small, the chunks will not change once the chunking size has exceeded the file size.

We define *chunk popularity* as the number of duplicate occurrences of a chunk. In Figure 5 we present the cumulative chunk popularity distribution for incremental backup and a 4KB chunk size. (We also evaluated all other chunk sizes and did not find any significant difference from what we present here.) The upper curve shows that about 60% of all chunk hashes appear only once in the data set, and according to the lower curve these chunks consume less than 10% of the entire non-deduplicated data size. In contrast, chunks that appear at least 20 times take more than 70% of the total non-deduplicated space (lower curve, center to right-hand side) but account for fewer than 9% of all chunks (upper curve, right-hand portion). Chunks that appear at least 100 times are fairly popular: they take less about 0.6% of the unique chunks but occupy nearly 30% of the whole space.

This skew in chunk popularity has also been found in primary storage [5], [22] and HPC systems [25]; those researchers found that chunks that appear 2–32 times contribute the most to the deduplication ratio in primary storage, while in our data set, the popularity is much higher. Identifying such popular chunks would be useful in optimizing performance. For example, keeping hot chunks in memory could accelerate chunk indexing and improve cache hit ratios.

We also found that most of the shared data among different users belongs to such popular chunks, a phenomenon that we discuss in Section IV-C2.

To study the distribution of file sizes, we chose several representative users. Figure 6 shows that the sizes of most files are between 1 and 64KB. However, the average file size in our data set is 366KB due to large vmdk files. Our average file size is smaller than has been reported in backup workloads [32] because backup software often combines many smaller files into larger `tar` files.

C. User-Based Analysis

Past studies have often focused on whole data sets and did not study the data from the users' perspective. Although each user's data forms a basic unit that is part of the data set, per-user data can have its own special characteristics. In this section, we show some interesting results from our 33 users. We have studied our data carefully, both individually and in various groupings. Due to space limitations, we present a representative sample of these individuals and groups, carefully chosen to highlight key findings seen repeatedly. For example, when studying each user's deduplication ratio, we selected accounts that covered different characteristics, such as the total size or lifetime of the user's data. To show how deduplication ratios change over time, we selected users who have matching start times.

1) Per-User Analysis: Due to users joining and leaving our system, their snapshots have varying start and end times. When users leave, we keep all their previous snapshots but stop collecting new ones. The duration of each user's observations varied from 1–20 months. The data-set sizes are also different; the largest user's data is about three orders of magnitude larger than the smallest.

The deduplication ratio for each user at different chunk sizes is shown in Figure 7. Here we chose 7 representative users based on characteristics such as size and duration; user 15 has both the longest duration (20 months) and the largest data-set size (8.3TB). Figure 7 shows large differences among users' effective deduplication ratios considering meta-data overhead. Using 128KB chunking, the highest deduplication ratio is over 2,400 (uncommon for most users) while the lowest is less than 40. Moreover, each user's sensitivity to chunk size also varies significantly. For Users 18 and 21, the raw deduplication ratio is so large that meta-data takes a large fraction of the space after deduplication; as a result, their effective deduplication ratio at 2KB chunking is only about 33% of that achieved by 128KB chunks. But for User 1, the 2KB-chunking deduplication ratio is about 76% of the 128KB one. The best chunking sizes for each user are also different:

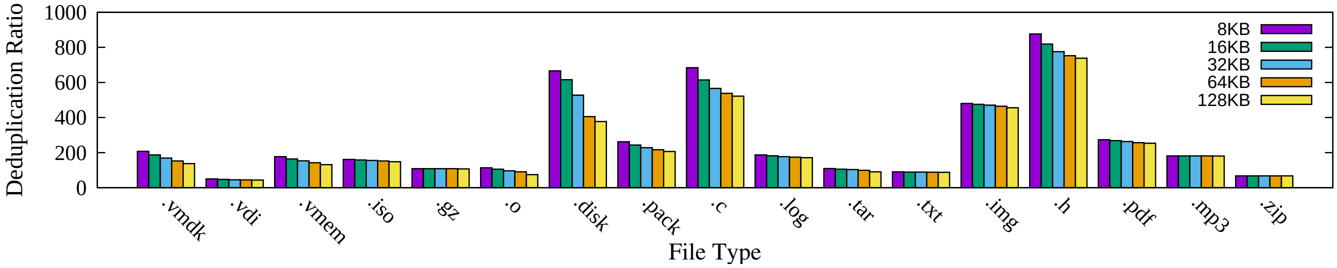

Fig. 4. Deduplication ratio of different file types at different chunking sizes

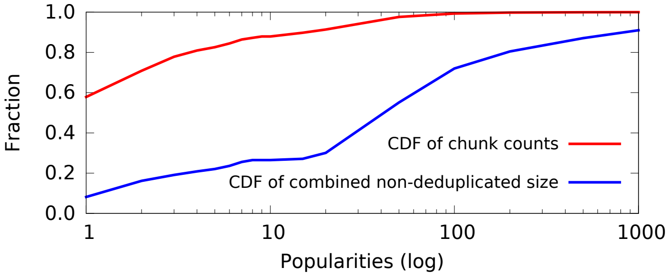

Fig. 5. Chunk popularities (red) and their sizes (blue).

User 1 does well at 16KB, User 18 at 128KB, and several others are in between. User 18 is special, as we can see a steady increase in deduplication ratio as the chunking size increases. This is because several large and highly redundant benchmarking data files take over 98% of the total size. This user's raw deduplication ratio is over 5,000; thus the meta-data takes a large fraction of the total post-deduplication space.

One reasonable explanation for the difference in deduplication ratios is the varying user lifetimes in the system. For full backups, a user's deduplication ratio will probably increase over time as more snapshots are added. However, we found that the large differences in deduplication ratios among users in our dataset were independent of their lifetimes. Figure 8 shows how ratios changed as snapshots accumulated. The snapshots of these five users had the same start times. But as we can see, User 17's deduplication ratio increased rapidly from the start. Although User 17 had fewer snapshots, the final deduplication ratio was over 1,000, far greater than the others, mainly due to high internal redundancy. Conversely, User 5's deduplication ratio *dropped* over time, mainly because a high file churn rate led to the addition of many new chunks. From Figure 8, we can see that the number of snapshots is not the sole factor affecting a user's overall deduplication ratio. We also found that the file type is not the main reason for differences in deduplication ratios. Even users with similar file type distributions (e.g., heavy users of VMDK files) also varied significantly in deduplication ratio, while some users with similar deduplication ratios had different file types. The important factors seems to be the characteristics of the users' own data, such as internal redundancy, and the user's activity level. Overall, we conclude that "not all users are created equal," even when they are performing similar jobs. Therefore, future deduplication systems should account for such behavior to improve efficiency.

2) Analysis of Groups of Users: We now turn to cross-user redundancies. We used a representative sample of users, as shown in Figure 9. Each number in the heat map is the percentage of a user's data that is also found in another's. For example, for User 1, more than 40% of that user's data can be found in User 15's data, and the same is true for User 28's (though precisely what is shared might differ between the two). This figure is not symmetric because each user's own data size is different. The figure shows that for each user, redundancies with others varied significantly. We confirmed these results for all users and found no obvious pattern.

While studying inter-user duplication, we found that users can be divided into groups in which the shared data of any two members occupies a large fraction of their total space. To analyze the characteristics of the shared data, we selected four representative users (13, 15, 19, and 28) as a group. Here we define a user's unique data set as S; $|S|$ means the number of chunks in S; and the data shared between users X and Y is $S_X \wedge S_Y$. Our results showed that 98% of the chunks shared between Users 13 and 19 could also be found in the shared data between Users 13 and 15. Stated mathematically:

$$\frac{|(S_{13} \wedge S_{19}) \wedge (S_{13} \wedge S_{15})|}{min(|S_{13} \wedge S_{19}|, |S_{13} \wedge S_{15}|)} = 0.98 \tag{2}$$

We checked this number for other combinations in this 4-user virtual group, and the results were between 91–98%. Thus, the data shared by the users in this group is fairly similar, so in a cluster deduplication system, grouping them into one storage node would improve the overall deduplication ratio.

Lastly, we found that chunks shared among users tended to have much higher popularity than average, as shown in Figure 10. For User 13, for example, the average chunk popularity was 22 (shown as the blue bar), while the popularities of chunks shared with Users 15, 19, and 28 were 189, 262, and 192, respectively. This means there was a 8.6–11.9× increase in these shared chunks' popularity compared with the average popularity of all chunks in User 13's data. Our results for other users and groups supported this conclusion. This means that shared user data has a *higher* deduplication ratio than one user's own data.

V. Conclusion and Future work

We studied a locally collected data set that spans a period of 2.5 years (over 4,000 daily user snapshots accounting for over 21 months). Our data is publicly available, with continuing release updates. The long time span and rich meta-data make the data valuable for studies of realistic long-term trends. Our findings can help inform future deduplication storage designs.

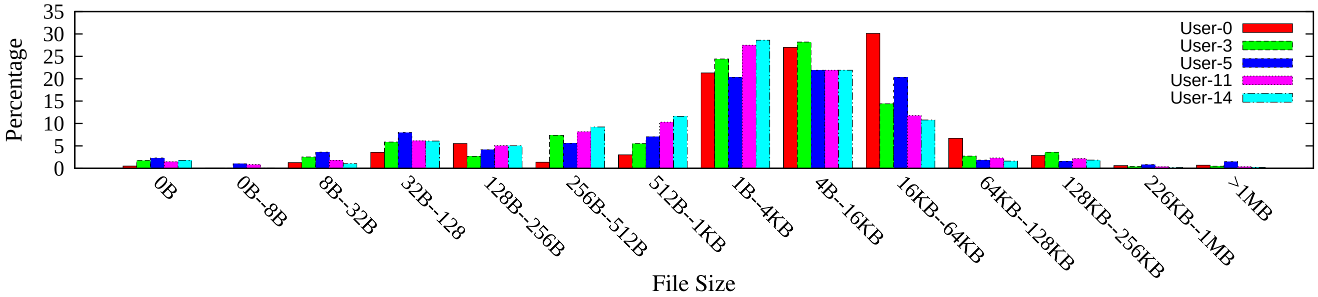

Fig. 6. Distribution of counts of different file types for a few users.

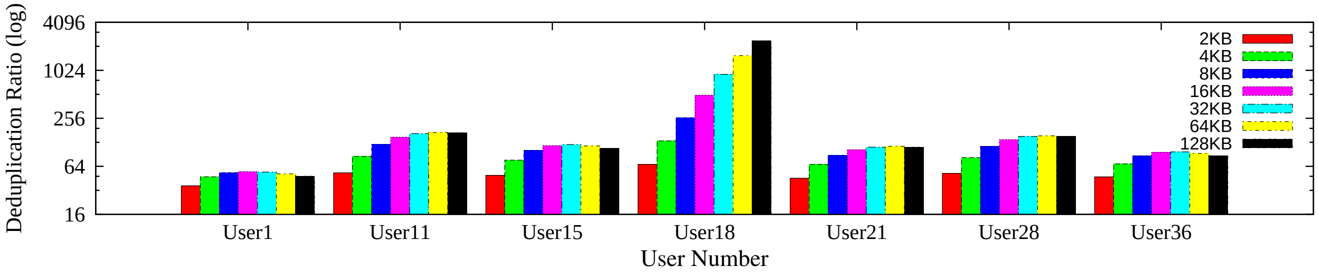

Fig. 7. Effective deduplication ratios of seven users at different chunking sizes, considering meta-data overheads.

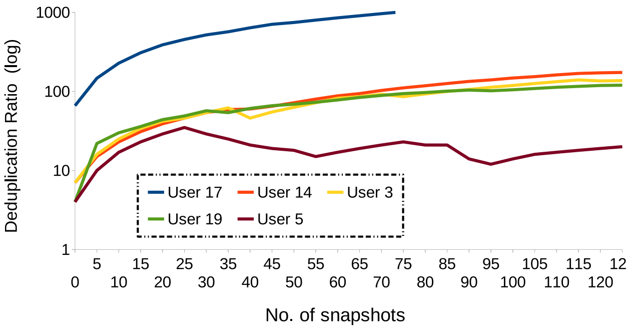

Fig. 8. Users' deduplication ratio (log) vs. number of snapshots.

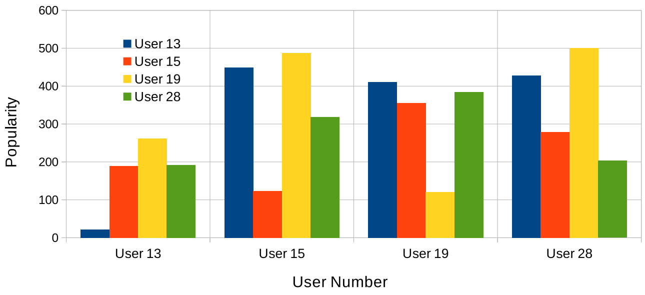

Fig. 10. The popularity of users' shared data. The shared chunks are much more popular than the average.

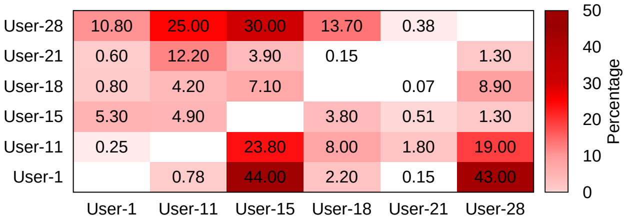

Fig. 9. Data redundancies among users.

In our analysis, we found that a smaller chunk size does not always save space, given the cost of additional meta-data. In our data set, a 32KB or even larger chunk size made the system the most space-efficient. Whole-file chunking produced the worst results in our study, because large files tended to dominate the data and had much lower deduplication ratios. Surprisingly, our user-centric studies showed that data belonging to each user varies significantly in deduplication ratio and sensitivity to chunk size. We also found a large difference in deduplication ratios across users, which can help in grouping users for future cluster storage. A detailed study of users who shared data showed that they have a higher deduplication ratio than average, which suggests that data shared among users tends to be more popular in general.

In the future, we plan to further analyze our data set. Cluster deduplication is a promising direction for future research. Our study indicates that the chunks shared among users in a group were similar, and those chunks also had high popularity. Based on these characteristics as seen in the first several snapshots of each user, we could group users together to maximize deduplication.

Another interesting metric is fragmentation: since our data set has a long duration, we can investigate how fragmentation accumulates and how it affects restore speeds. While collecting the snapshots, we also found that fragmentation decreases backup throughput because a traditional depth-first scan of the file system causes newly added or modified chunks to be stored into new containers, leading to cache misses in future scans. We are currently investigating alternative scan orders to improve cache hit ratios and restore speeds.

Acknowledgments

We thank the anonymous MSST reviewers for their useful feedback. This work was made possible in part thanks to EMC support, NSF awards CNS-1251137 and CNS-1302246, the National Natural Science Foundation of China under Grants No. 61433019 and U1435217, and China 863 program grant 2015AA015305.

References

[1] D. Bhagwat, K. Eshghi, D. Long, and M. Lillibridge. Extreme binning: Scalable, parallel deduplication for chunk-based file backup. In *Proceedings of the MASCOTS Conference*, 2009.

[2] B. Debnath, S. Sengupta, and J. Li. ChunkStash: Speeding up inline storage deduplication using flash memory. In *Proceedings of the USENIX Annual Technical Conference*, 2010.

[3] W. Dong, F. Douglis, K. Li, H. Patterson, S. Reddy, and P. Shilane. Tradeoffs in scalable data routing for deduplication clusters. In *Proceedings of the Ninth USENIX Conference on File and Storage Technologies (FAST '11)*, 2011.

[4] F. Douglis, D. Bhardwaj, H. Qian, and P. Shilane. Content-aware load balancing for distributed backup. In *Proceedings of USENIX Large Installation System Administration Conference*, 2011.

[5] A. El-Shimi, R. Kalach, A. Kumar, A. Oltean, J. Li, and S. Sengupta. Primary data deduplication—large scale study and system design. In *Proceedings of the USENIX Annual Technical Conference*, 2012.

[6] D. Frey, A. Kermarrec, and K. Kloudas. Probabilistic deduplication for cluster-based storage systems. In *Proceedings of the Symposium on Cloud Computing (SOCC)*, 2012.

[7] FSL. Fslhomes data set and tools. tracer.filesystems.org.

[8] Min Fu, Dan Feng, Yu Hua, Xubin He, and Zuoning Chen. Accelerating restore and garbage collection in deduplication-based backup systems via exploiting history information. In *Proceedings of Annual Technical Conference*, 2014.

[9] Y. Fu, N. Xiao, X. Liao, and F. Liu. Application-aware client-side data reduction and encryption of personal data in cloud backup services. *Journal of Computer Science and Technology*, 28(6), November 2013.

[10] Yinjin Fu, Hong Jiang, and Nong Xiao. A Scalable Inline Cluster Deduplication Framework for Big Data Protection. In *Proceedings of International Conference on Middleware*, 2012.

[11] A. Gharaibeh, C. Constantinescu, M. Lu, A. Sharma, R. Routray, P. Sarkar, D. Pease, and M. Ripeanu. DedupT: Deduplication for tape systems. Technical report, IBM, 2014.

[12] Jhon Gratz and David Reinsel. The digital universe decade - are you ready? IDC White Paper, www.idc.com, 2010.

[13] F. Guo and P. Efstathopoulos. Building a high-performance deduplication system. In *Proceedings of the USENIX Annual Technical Conference*, 2011.

[14] M. Jianting. A deduplication-based data archiving system. In *Proceedings of the International Conference on Image, Vision and Computing (ICIVC)*, 2012.

[15] K. Jin and E. Miller. The effectiveness of deduplication on virtual machine disk images. In *Proceedings of the Israeli Experimental Systems Conference (SYSTOR)*, 2009.

[16] R. Koller and R. Rangaswami. I/O deduplication: Utilizing content similarity to improve I/O performance. In *Proceedings of the Eighth USENIX Conference on File and Storage Technologies (FAST '10)*, 2010.

[17] M. Li, C. Qin, and P. Lee. Cdstore: Toward reliable, secure, and cost-efficient cloud storage via convergent dispersal. In *USENIX Annual Technical Conference*, 2015.

[18] M. Lillibridge and K. Eshghi. Improving restore speed for backup systems that use inline chunk-based deduplication. In *Proceedings of the Eleventh USENIX Conference on File and Storage Technologies (FAST '13)*, 2013.

[19] 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 Seventh USENIX Conference on File and Storage Technologies (FAST '09)*, 2009.

[20] X. Lin, F. Douglis, J. Li, X. Li, R. Ricci, S. Smaldone, and G. Wallace. Metadata considered harmful ... to deduplication. In *HotStorage'15*, 2015.

[21] X. Lin, M. Hibler, E. Eide, and R. Ricci. Using deduplicating storage fot efficient disk image deployment. In *Proceedings of IEEE International Conference on Software Testing, Verification and Validation*, 2015.

[22] M. Lu, D. Chambliss, J. Glider, and C. Constantinescu. Insights for data reduction in primary storage: A practical analysis. In *Proceedings of the Israeli Experimental Systems Conference (SYSTOR)*, 2012.

[23] D. Meister and A. Brinkmann. dedupv1: Improving deduplication throughput using solid state drives (SSD). In *Proceedings of the MSST Conference*, 2010.

[24] D. Meister, A. Brinkmann, and T. Suss. File recipe compression in data deduplication systems. In *Proceedings of the Eleventh USENIX Conference on File and Storage Technologies (FAST '13)*, 2013.

[25] D. Meister, J. Kaiser, A. Brinkmann, T. Cortes, M. Kuhn, and J. Kunkel. A study on data deduplication in HPC storage systems. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, 2012.

[26] D. Meyer and W. Bolosky. A study of practical deduplication. In *Proceedings of the Ninth USENIX Conference on File and Storage Technologies (FAST '11)*, 2011.

[27] N. Park and D. Lilja. Characterizing datasets for data deduplication in backup applications. In *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC)*, 2010.

[28] C. Olofson R. Villars and M. Eastwood. Big data: What it is and why you should care. A White Paper from www.idc.com, June 2011.

[29] V. Tarasov, A. Mudrankitony, W. Buik, P. Shilane, G. Kuenning, and E. Zadok. Generating realistic datasets for deduplication analysis. In *Proceedings of the USENIX Annual Technical Conference*, 2012.

[30] C. Ungureanu, B. Atkin, A. Aranya, S. Gokhale, S. Rago, G. Calkowski, C. Dubnicki, and A. Bohra. HydraFS: a high-throughput file system for the HYDRAstor content-addressable storage system. In *Proceedings of the Eighth USENIX Conference on File and Storage Technologies (FAST '10)*, 2010.

[31] C. Vaughn, C. Miller, O. Ekenta, H. Sun, M. Bhadkamkar, P. Efstathopoulos, and E. Kardes. Soothsayer: Predicting capacity usage in backup storage systems. In *In Proceedings of MASCOTS conference*, 2015.

[32] G. Wallace, F. Douglis, H. Qian, P. Shilane, S. Smaldone, M. Chamness, and W. Hsu. Characteristics of backup workloads in production systems. In *Proceedings of the Tenth USENIX Conference on File and Storage Technologies (FAST '12)*, 2012.

[33] W. Xia, H. Jiang, D. Feng, and Y. Hua. SiLo: A similarity-locality based near-exact deduplication scheme with low RAM overhead and high throughput. In *Proceedings of the USENIX Annual Technical Conference*, 2011.

[34] Y. Zhou, D. Feng, W. Xia, M. Fu, F. Huang, Y. Zhang, and C. Li. Secdep: A user-aware efficient fine-grained secure dedupication scheme with multi-level key management. In *Proceedings of International Conference on Massive Storage Systems and Technology*, 2015.

[35] 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*, 2008.