Combining Deduplication and Delta Compression to Achieve Low-Overhead Data Reduction on Backup Datasets

Wen Xia[†,*] xia@hust.edu.cn
Hong Jiang[‡] jiang@cse.unl.edu
Dan Feng[†,✉] dfeng@hust.edu.cn
Lei Tian[‡] tian@cse.unl.edu

[†] *Wuhan National Laboratory for Optoelectronics, Wuhan, China*
[†] *School of Computer, Huazhong University of Science and Technology, Wuhan, China*
[‡] *Dept. of Computer Science and Engineering,University of Nebraska-Lincoln, Lincoln, NE, USA*

Abstract: Data reduction has become increasingly important in storage systems due to the explosive growth of digital data in the world that has ushered in the big data era. In this paper, we present DARE, a Deduplication-Aware Resemblance detection and Elimination scheme for compressing backup datasets that effectively combines data deduplication and delta compression to achieve high data reduction efficiency at low overhead. The main idea behind DARE is to employ a scheme, call Duplicate-Adjacency based Resemblance Detection (*DupAdj*), by considering any two data chunks to be similar (i.e., candidates for delta compression) if their respective adjacent data chunks are found to be duplicate in a deduplication system, and then further enhance the resemblance detection efficiency by an improved super-feature approach. Our experimental results based on real-world and synthetic backup datasets show that DARE achieves an additional data reduction by a factor of more than 2 (2X) on top of deduplication with very low overhead while nearly doubling the data restore performance of deduplication-only systems by supplementing delta compression to deduplication.

1 Introduction

Data deduplication is a dictionary based data reduction approach popular in the backup/archiving storage area due to its demonstrated ability to effectively to compress backup/archiving datasets by a factor of 4-40X [1, 2]. In general, a chunk-level data deduplication scheme splits data blocks of a data stream (e.g., backup files and databases) into multiple data chunks of average size 8K or 4K with each being uniquely identified and duplicate-detected by a secure SHA-1 or MD5 hash signature (also called a fingerprint) [3, 4, 5, 1]. This secure fingerprint-based deduplication technique eliminates redundancy at the chunk or file level and thus scales better than the traditional LZ77 and Huffman coding based GZ compression [6, 4].

Delta compression, however, has been gaining increasing attention in recent years for its ability to remove redundancy among non-duplicate but very similar data files and chunks, for which the data deduplication technology often fails to identify and eliminate [7, 2]. For example, if chunk A_2 is similar to chunk A_1 (the base-chunk), the delta compression approach calculates and then only stores the differences (delta $\Delta_{1,2}$) and the mapping information between A_2 and A_1 [8]. Thus, it is considered a promising technique to effectively complement and supplement the fingerprint-based deduplication approaches by detecting and compressing similar data missed by the latter.

In this paper, we propose DARE, a low-overhead Deduplication-Aware Resemblance detection and Elimination scheme that effectively combines data deduplication and delta compression to achieve high data reduction efficiency at low overhead. The main contributions include:

- A "DupAdj" approach is proposed to exploit existing duplicate-adjacency information after deduplication to detect similar data chunks for delta compression. Specifically, due to locality of similar data in backup datasets, the non-duplicate chunks that are adjacent to the duplicate

[*] Part of the work was done when the first author was a visiting student at University of Nebraska-Lincoln.

1068-0314/14 $31.00 © 2014 IEEE
DOI 10.1109/DCC.2014.38

Table 1: Duplicate detection vs. resemblance detection for data reduction.

	Duplicate Detection	Resemblance Detection
Objects	Duplicate data	Similar data
Granularity	Chunk-level	Byte-level
Representative Methods	Secure-Fingerprint based Deduplication	Super-Feature based Delta Compression
Scalability	Strong	Weak
Representative Systems	LBFS [3], Venti[9], DDFS [5]	REBL[7], DERD[10], SIDC[2]

ones are considered good delta compression candidates for further data reduction.

- A theoretical and empirical study of the traditional super-feature approach is conducted, which suggests that an improved resemblance detection for further delta compression is possible when the aforementioned existing duplicate-adjacency information is lacking or limited.
- An investigation into the restoration of deduplicated and delta compressed backup data suggests that delta compression has the potential to improve the data-restore performance of deduplication-only systems by further removing redundancy after deduplication and thus enlarging the logical space of the restoration cache.
- Our experimental evaluation results, based on real-world and synthetic backup datasets, show that DARE only consumes about 1/4 and 1/2 respectively of the computation and indexing overhead required by the traditional super-feature approach for resemblance detection while achieving a superior data reduction performance.

The rest of the paper is organized as follows. Section 2 presents the background for this work. The architecture, key data structures, and data reduction schemes of DARE are described in Section 3. We present and discuss the experimental evaluation of the DARE prototype in Section 4. Section 5 draws conclusions and provides directions for future work.

2 Related Work

Data deduplication is gaining increasing traction in data-intensive storage systems as one of the most efficient data reduction approaches in recent years. Fingerprint-based deduplication techniques eliminate duplicate chunks by checking their secure fingerprints (i.e., SHA-1/SHA-256 signatures), which has been widely used in commercial backup and archiving storage systems [5, 1].

One of the main challenges facing data deduplication is how to maximally detect and eliminate data redundancy in storage systems at low overhead. In order to find more redundant data, the Content-Defined Chunking (CDC) approach was proposed in LBFS to find the proper cut-point of each chunk in the files and address the boundary-shift problem [3].

Resemblance detection with delta compression [7, 10, 11], as another approach to data reduction in storage systems, was proposed more than ten years ago but was subsequently overshadowed by fingerprint-based deduplication [5, 1] due to the former's poor scalability. Table 1 compares these two data reduction approaches. Resemblance detection detects redundancy among similar data at the byte level while duplicate detection finds totally identical data at the chunk level, which makes the latter much more scalable than the former in large-scale storage systems.

REBL[7] and DERD [10] are typical super-feature based resemblance detection approaches for data reduction. They compute the features of the data stream (e.g., Rabin Fingerprints [12]) and group features into super-features to capture the resemblance of data and then delta compress these data. All these approaches require high computation and indexing overhead for resemblance detection. As a result, the simpler and faster deduplication method has become a more popular data reduction technology in the recent five years [1].

Nevertheless, resemblance detection is gaining increasing traction in storage systems because of its ability to capture and eliminate data redundancy among similar but non-duplicate data chunks,

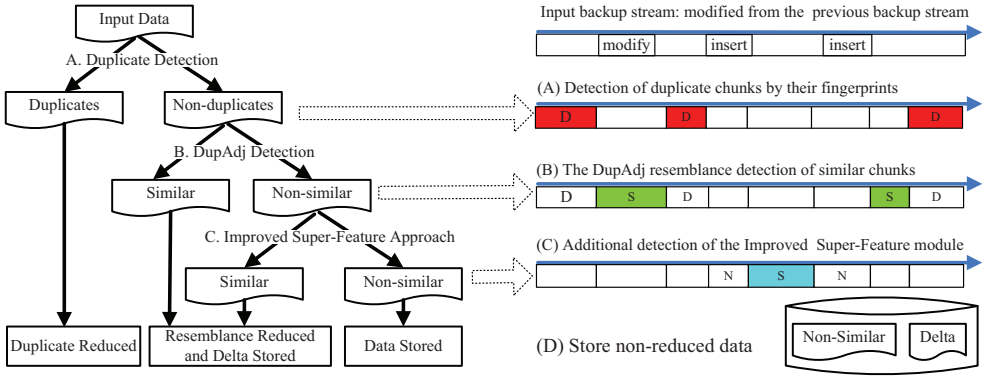


Figure 1: The data reduction workflow of DARE, showing an example of resemblance detection for delta compression first by the DupAdj approach and then by the super-feature approach. 'D', 'S' and 'N' here refer to a duplicate chunk, a similar chunk and a chunk that is neither duplicate nor similar, respectively.

which effectively complements and supplements fingerprint-based deduplication. Difference Engine [13] and I-CASH [14] make full use of the delta compression technology to eliminate redundancy in memory pages and SSD caches respectively. Shilane et al. [2] proposed a stream-informed delta compression approach to reducing similar data transmission and thus accelerating data replication in a WAN environment. This approach is also super-feature based and complements deduplication by only detecting resemblance among non-duplicate chunks in the cache that preserves the backup stream locality. It avoids the costly global indexing, at a limited loss of data reduction.

3 Design and Implementation

3.1 An Overview of the DARE Data Reduction Approach

DARE is designed to improve resemblance detection for additional data reduction in deduplication-based backup/archiving storage systems. Figure 1 shows a detailed case of the workflow of the DARE system. For an incoming backup stream, DARE goes through the following four key steps:

1. **Duplicate Detection.** The data stream is first chunked by the CDC approach [3], fingerprinted by SHA-1, duplicate-detected, and then grouped into container of sequential chunks to preserve the backup-stream locality [5].
2. **Resemblance Detection.** The DupAdj resemblance detection module in DARE first detects duplicate-adjacent chunks in the containers formed in Step (1) (see Section 3.2). After that, DARE's improved super-feature module further detects similar chunks in the remaining non-duplicate and non-similar chunks that may have been missed by the DupAdj detection module when the duplicate-adjacency information is lacking or weak (see Section 3.3).
3. **Delta Compression.** For each of the resembling chunks detected in Step (2), DARE reads its base-chunk, then delta encodes their differences by the Xdelta algorithm [8]. In order to reduce the number of disk reads, an LRU and locality-preserved cache is implemented here to prefetch the base-chunks in the form of locality-preserved containers.
4. **Storage Management.** The data NOT reduced, i.e., non-similar or delta chunks, will be stored as containers into the disk. The file mapping information among the duplicate chunks, resembling chunks, and non-similar chunks will also be recorded as the file recipes to facilitate future data restore operations in DARE.

For the restore operation, DARE will first read the referenced file recipes and then read the duplicate and non-similar chunks one by one from the referenced containers on disk according to the mapping information in the file recipes. For the resembling chunks, DARE needs to read both their delta data and base-chunks and then delta decode them.

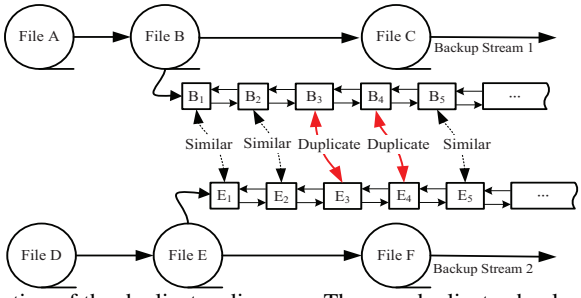


Figure 2: A conceptual illustration of the duplicate adjacency. The non-duplicate chunks adjacent to duplicate ones are considered potentially similar and thus good delta compression candidates.

3.2 DupAdj: Duplicate-Adjacency based Resemblance Detection

As a salient feature of DARE, the DupAdj approach detects resemblance by exploiting the existing duplicate-adjacency information of a deduplication system. This is based on our observation that the modified chunks may be very similar to their previous versions in a backup system while unmodified chunks will remain duplicate and are easily identified by the deduplication process.

Figure 2 illustrates a case of duplicate data chunks and their immediate non-duplicate neighbors. DARE records the backup-stream locality of chunk sequence by a *doubly-linked list*, which allows an efficient search of the duplicate-adjacent chunks for resemblance detection by traversing to prior or next chunks on the list, as shown in Figure 2. When the DupAdj Detection module of DARE processes an input file, it will traverse all the chunks by the aforementioned *doubly-linked list* to find the already duplicate-detected chunks. If chunk E_m of the input file E was detected to be a duplicate of chunk B_n of file B, DARE will traverse the doubly-linked list of B_n in both directions (e.g., E_{m+1} & B_{n+1} and E_{m-1} & B_{n-1}) in search of potentially similar chunk pairs between files E and B, until a dissimilar chunk or an already detected duplicate or similar chunk is found.

Since the DupAdj detection approach only adds a doubly-linked list to an existing deduplication system, DARE avoids the computation and indexing overhead of the conventional super-feature approach for resemblance detection in data reduction systems. In case where the duplicate-adjacency information is lacking or limited, DARE uses an improved super-feature approach to further detecting and eliminating resemblance as discussed in the next subsection.

3.3 Theoretical Analysis of Super-Feature based Resemblance Detection

As mentioned in Section 2, traditional super-feature approaches generate features by Rabin fingerprints and group these features into super-features to detect resemblance for data reduction [7, 10]. For example, $Feature_i$ of a chunk (length = N) is uniquely generated with a randomly pre-defined value pair m_i & a_i and N Rabin fingerprints (as used in Content-Defined Chunking [3]) as follows:

$$Feature_i = Max_{j=1}^{N}\{(m_i * Rabin_j + a_i)\, mod\, 2^{32}\} \quad (1)$$

A super-feature of this chunk, $SFeature_x$, can then be calculated by several features as follows:

$$SFeature_x = Rabin(Feature_{x*k}, ..., Feature_{x*k+k-1}) \quad (2)$$

For example, to generate two super-features with k=4 features each, we must first generate 8 features, namely, features 0...3 for $SFeature_1$ and features 4...7 for $SFeature_2$. For similar chunks that differ only in a tiny fraction of bytes, most of their features will be identical due to the random distribution of the chunk's maximal-feature positions [15]. Thus two data chunks can be considered very similar if any one of their super-features matches. The state-of-the-art studies [7, 2] recommend

the use of 4 or more features to generate a super-feature to minimize false positives of resemblance detection for delta compression.

However, our theoretical analysis and experimental observations suggest that the probability of false positives resulting from feature collision is extremely low but increasing the number of features per super-feature actually decreases the efficiency of resemblance detection. First, the false positives of 64-bit Rabin fingerprints tend to be very low as discussed in previous studies [12, 16]. This means that two chunks will have the same content of a hashing region (32 or 48 bytes) with a very high probability if they have the same Rabin fingerprint. Next, the probability of two similar chunks having the same feature is highly dependent upon their similarity degree according to Broder's theorem [17]. The less similar two data chunks are to each other, the smaller the probability of them having the same feature. Thus, the probability of two data chunks S_1 and S_2 being detected as resembling to each other by N features can be computed as follows.

$$Pr[\bigcap_{i=1}^{N} Max_i(H(S_1)) = Max_i(H(S_2))] = \{\frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}\}^N = \gamma^N \tag{3}$$

This probability is clearly decreasing as a function of the number of features used in a super-feature, as indicated by the above probability expression. Nevertheless, all recent studies on delta compression suggest to increase the number of super-features [2]. If any one of the super-features of two data chunks matches, the two chunks are considered similar to each other. Thus, the probability of resemblance detection, expressed as $1-(1-\gamma^N)^M$, can be increased by the number of super-features, M.

For simplicity, assume that the similarity degree γ follows a uniform distribution in the range [0, 1] (note that the actual distribution may be much more complicated in real workloads), the expected value of resemblance detection can be expressed as a function of the number of features per super-feature and the number of super-features under the aforementioned assumption as:

$$\int_0^1 x(1-(1-x^N)^M)dx = \sum_{i=1}^{M} C_M^i(-1)^{i+1}\frac{1}{N*i+2} \tag{4}$$

This expression of resemblance detection suggests that the larger the number N of features used in obtaining a super-feature is, the less capable the super-feature is of resemblance detection. On the other hand, the larger the number M of super-features is, the more resemblance can be detected and the more redundancy can be eliminated.

Thus, DARE employs an improved super-feature approach with fewer features per super-feature to effectively complementing the DupAdj resemblance detection as discussed in Section 3.1. And our experimental results suggest that two features per super-feature appear to hit the "sweet spot" of resemblance detection in deduplication systems in terms of cost effectiveness.

4 Performance Evaluation

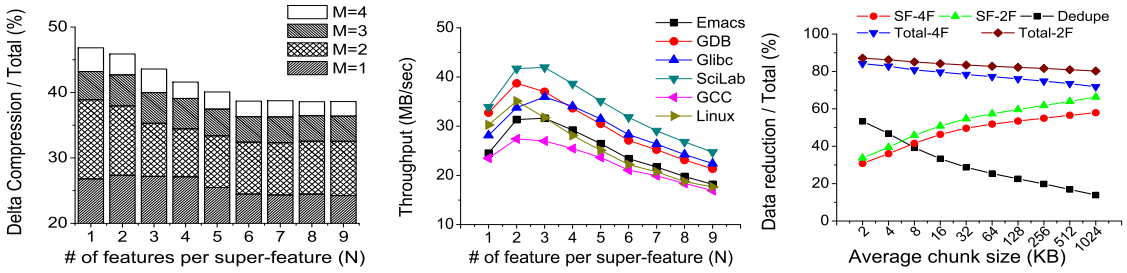
4.1 Experimental Setup

Platform of the DARE Prototype. We have implemented a prototype of DARE and tested it on the Ubuntu 12.04 operating system running on a quad-core Xeon E5606 processor at 2.13 GHz, with a 16GB RAM, a 14TB RAID6 disk array that consists of sixteen 1TB disks, and a 120GB SSD of KINGSTON SVP200S37A120G.

Configurations for Data Reduction. DARE employs the widely used Rabin algorithm [12, 16] and SHA-1 hash function [5] respectively for the chunking and fingerprinting processes in data deduplication, with an average chunk size of 8KB. For the resemblance detection, DARE adopts

Table 2: Workload characteristics of the six real-world and two synthetic backup datasets.

Datasets	Emacs	GDB	Glibc	SciLab	GCC	Linux	Freq	Less
versions	8	10	35	10	20	40	20	30
Size	1.15 GB	1.37 GB	3.18 GB	4.94 GB	8.91 GB	16.8 GB	857 GB	1372 GB



(a) Compression vs. features and SFs (b) Throughput vs. features (c) Data reduction vs. avg. chunk size

Figure 3: An empirical study of the super-feature based resemblance detection approach for data reduction. (a) Data reduction of the super-feature approach as a function of the number N of features per super-feature and the number M of super-features (shaded segments on each bar). (b) Data-reduction throughput of the four super-feature based approaches as a function of the number N of features per super-feature. (c) Data reduction as a function of the average chunk size, by deduplication and the 2-features-per-SF (SF-2F) and the 4-features-per-SF (SF-4F). Total-2F and Total-4F refer to Dedupe + SF-2F and Dedupe + SF-4F respectively. Note that results of subfigures (a) and (c) are evaluated on the GCC dataset and results of other datasets are similar to GCC thus are omitted due to space limit.

a CDC sliding window size of 48 bytes to generate features and uses the Xdelta algorithm [8] to compress the detected similar chunks.

Datasets. Six well-known open-source projects representing typical backup workloads of deduplication and resemblance detection are used in this evaluation as shown in Table 2. These datasets consist of large tarred files representing sets of source code files or objects concatenated together by backup software [1, 2]. In order to assess the scalability of DARE, we generate two larger synthetic backup datasets according to the principles of synthesizing datasets outlined in recent studies [18, 19]. We obtain the first version of the datasets from our research group of 16 users with 192K files and totaling 42GB, mutate the data by the operations of "modify", "delete" and "new" for 20%, 1%, and 1% of the files respectively in the "Freq" dataset and 10%, 0.5%, and 0.5% of the files respectively in the "Less" dataset, and then concatenate individual files to the tarred files. The file modifications are also applied in the beginning, middle, and end of the files as suggested in [18].

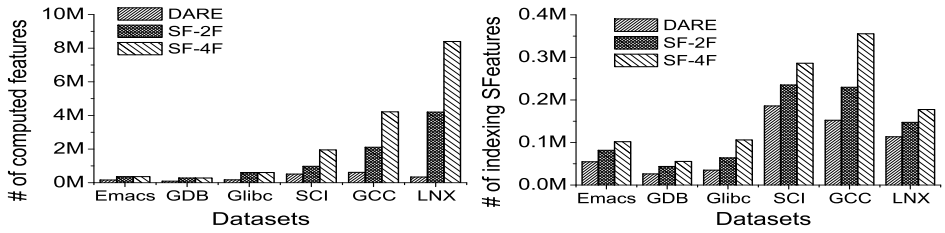
4.2 An Empirical Study of the Super-Feature Approach

We first examine the impact of the number of features per SF and the number of SFs used in resemblance detection via an evaluation driven by a real-world dataset. In Figure 3(a), we plot the trend of data reduction of the traditional super-feature based approach as a function of the number N of features per SF and the number M of SFs. We find a general tendency of redundancy elimination being a decreasing function of 'N' and an increasing function of 'M', which is consistent with the results of our mathematical analysis presented in Section 3.3. That is, the more SFs are used, the more resemblance can be exposed to eliminate more redundancy. On the other hand, the more features used in generating each SF, the less redundancy will be eliminated, because the probability of more features in an SF being identical is smaller than that of fewer features in an SF.

To evaluate the overall performance impact of the number of features per SF, we plot in Figure 3(b) the data-reduction throughput of the super-feature based approaches running on the RAID as a function of the number of features per SF. We find that the one-feature-per-SF approach has a lower throughput than the two- or three-features-per-SF approaches where the highest throughputs

Table 3: A comparison among Deduplication, DupAdj, DARE, SF-2F, and SF-4F approaches in the data reduction measure under six real-world datasets, and both tarred and untarred versions for a total of 12 backup datasets.

Datasets	Tarred						UnTarred					
	Emacs	GDB	Glibc	SciLab	GCC	Linux	Emacs	GDB	Glibc	SciLab	GCC	Linux
Versions	8	10	35	10	20	40	8	10	35	10	20	40
Dedupe	37.1%	48.7%	52.2%	56.9%	39.1%	40.9%	43.5%	70.6%	87.9%	77.5%	83.5%	96.7%
DupAdj	32.1%	33.5%	29.2%	19.5%	38.2%	53.4%	29.6%	10.9%	2.9%	5.2%	7.2%	0.7%
DARE	41.0%	40.8%	36.9%	25.2%	46.7%	54.1%	37.7%	18.2%	7.3%	10.4%	9.9%	1.0%
SF-2F	33.7%	36.4%	35.3%	22.6%	45.2%	54.4%	31.7%	16.5%	6.6%	9.6%	9.1%	0.9%
SF-4F	28.2%	33.4%	30.4%	18.8%	40.6%	53.5%	28.0%	14.2%	5.7%	8.1%	7.3%	0.6%



(a) Computation Overhead (b) Indexing Overhead

Figure 4: A comparison among DARE, SF-2F, and SF-4F in terms of computation and memory (indexing) overhead.

are achieved. This is because the former detects more similar chunks than the latter and thus induces more random I/Os in reading the base-chunks of the resemblance-detected chunks for delta compression. On the other hand, while approaches based on four or more features per super-feature detect less resemblance, they incur higher computation latency (overhead), which lowers the throughput.

Figure 3(c) shows the data reduction results of deduplication and delta compression approaches as a function of the average chunk size. It demonstrates that delta compression is very efficient in supplementing deduplication for data reduction. The larger the average chunk size is, the less duplicate data are detected. But the resemblance detection approach can detect almost all the redundant data (e.g., similar data) that deduplication fails to identify, regardless of the average chunk size.

4.3 Deduplication-Aware Resemblance Detection

In this subsection, we evaluate DARE's resemblance detection and elimination schemes and compare DARE with the super-feature-only approaches based on 3 SFs with two-features per SF (SF-2F) and 3 SFs with four-features per SF (SF-4F, the same configuration as SIDC [2]). Note that DARE's resemblance detection here is thus DupAdj supplemented by SF-2F, where SF-2F is applied only to chunks that DupAdj has failed to detect as similar, as discussed in Section 3.3.

Table 3 shows the additional data reduction on top of the conventional deduplication (Dedupe) achieved by the four resemblance detection schemes, DupAdj, DARE, SF-2F, and SF-4F, on the six real-world backup datasets (both tarred and untarred). Generally, the DupAdj approach achieves a resemblance-detection efficiency similar to the SF-4F approach and DARE detects about 2-6% and 3-10% more redundancy than the SF-2F and SF-4F approaches respectively. As indicated in Figure 3(a) and discussed in Section 4.2, SF-2F is more effective than SF-4F for resemblance detection. Thus DARE detects the most resemblance by combining the DupAdj and SF-2F resemblance detection approaches. In fact, our evaluation results suggest that the average similarity degree of the DupAdj-detected chunks is 0.895 while that of the SF-2F- and SF-4F-detected chunks are 0.892 and 0.934 on the above six datasets, which demonstrates that DupAdj is very effective and efficient in detecting resemblance among the post-deduplication chunks with a very low false-positive rate.

Figure 4 shows the computation and indexing overhead incurred by the three resemblance detec-

Table 4: A comparison among the duplicate-detection and resemblance-detection approaches in terms of data reduction efficiency on the two synthesized larger backup datasets Freq and Less. The "+" ("*") sign in front of a reduction percentage (factor) indicates the "additional" post-deduplication data reduction.

Datasets	Freq		Less	
Exact Dedupe	84.6%	6.5X	91.2%	11.4X
SiLo Dedupe [20]	84.4%	6.4X	91.2%	11.4X
Cached Index of DARE (DupAdj+SF-2F)	+9.28%	*2.5X	+5.01%	*2.3X
Cached Index of SF-2F	+7.01%	*1.8X	+4.31%	*2.0X
Cached Index of SF-4F (SIDC [2])	+5.32%	*1.5X	+3.27%	*1.6X
Full Index of SF-2F	+9.61%	*2.6X	+5.04%	*2.3X

tion schemes. Obviously, SF-4F, which computes more features but detects less resemblance, consumes the most amounts of computation and indexing resources for resemblance detection. DARE uses the same super-feature parameters as SF-2F but incurs only half of the computation and indexing overheads of the SF-2F approach because of DupAdj's very effective pre-screening of similar chunks by exploiting existing duplicate-adjacency information after data deduplication.

4.4 Scalability of DARE Data Reduction

In order to better evaluate the scalability of DARE on larger backup datasets, we have implemented the schemes of Stream-Informed Delta Compression (SIDC) [2] in SiLo [20], a memory-efficient near-exact deduplication system that exploits the backup-stream similarity and locality. As introduced in Section 2.1, SIDC only detects resemblance in the backup-stream locality-preserved cache that can reduce the indexing overhead of SFs and scales well in large-scale deduplication system. Thus we employ their method to assess the scalability of different resemblance detection schemes and implement SiLo with a 20MB locality cache (similar to SIDC [2]) with a segment size of 1MB. The stream-informed approaches are denoted by the "Cached/Cac." prefix in Table 4 and Figure 5.

The first column under a dataset (Freq or Less) in Table 4 shows the percentages of data reduction and the second column shows the data reduction factor (the ratio of before/after data reduction). Since SiLo achieves nearly 99% deduplication efficiency of the exact deduplication (i.e., full index in memory) while requiring substantially less memory overhead (about 1/250 of the exact deduplication approach) for fingerprint indexing [20], we only discuss the results of resemblance detection after SiLo's deduplication here in Table 4.

As shown in Table 4, resemblance detection further reduces the storage space, after the deduplication process, by a factor of about 2.5, meaning that we can save about 60% of the post-deduplication storage space by resemblance-detecting post-deduplication chunks. DARE achieves a superior data reduction efficiency on both datasets while the cached SF-2F and SF-4F schemes detect less resemblance, which is consistent with the results on our real-world datasets summarized in Table 3. Table 4 also shows that the cached SF-2F and SF-4F schemes fail to detect a noticeable amount of resemblance due to the sometimes weak or lack thereof locality in the backup stream as the limitations of the stream-informed cache approach [2].

To further understand the scalability of DARE, Figure 5 shows the percentages of resemblance detected by its DupAdj detection and by the SF-2F and SF-4F approaches showing the individual contributions by the 1^{st} SF, the 2^{nd} SF, and the 3^{rd} SF, on the two synthesized datasets. DupAdj is very effective and efficient in detecting resemblance in the deduplication system with abundant duplicate-adjacency information (i.e., dedupe factor > 2), leaving DARE's improved super-feature approach very little (smaller than 1%), if any, additional resemblance to detect (see the DARE bars).

Figure 6 shows that DARE achieves the highest throughputs among all the approaches compared running on both RAID-structured HDDs and the SSD. Cached SF-4F has the lowest throughput

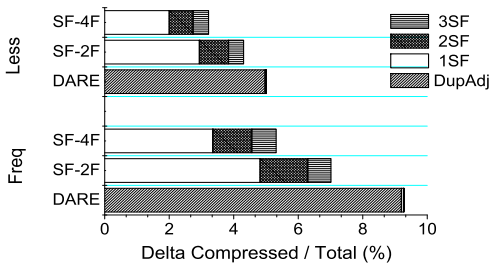
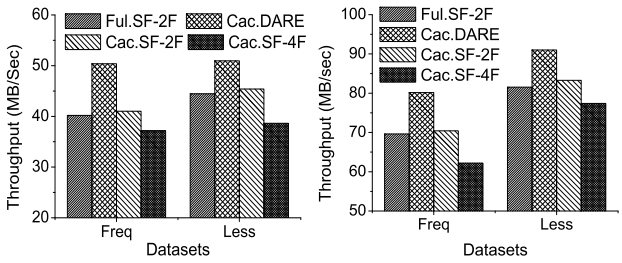
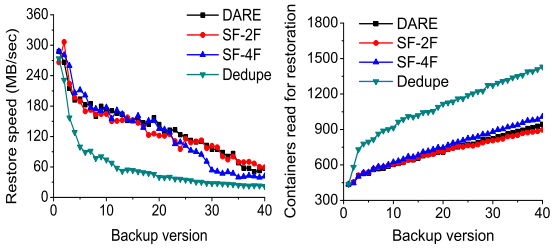


Figure 5: Percentages of data reduced by DupAdj, and the 1^{st} SF, 2^{nd} SF, 3^{rd} SF of the super-feature approach respectively in the stream-informed DARE, SF-2F, and SF-4F approaches.



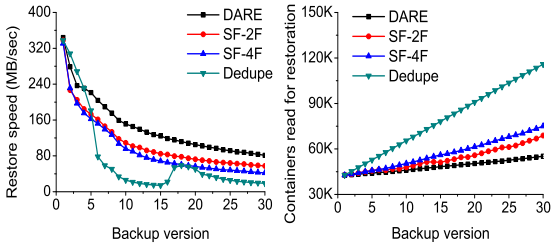
(a) Throughput on RAID (b) Throughput on SSD

Figure 6: Throughputs of four resemblance detection enhanced data reduction approaches on the two synthesized datasets.



(a) Restoration throughput (b) Containers read

Figure 7: Data-restore performance versus backup version on the Linux dataset with an LRU cache of size 256MB.



(a) Restoration throughput (b) Containers read

Figure 8: Data-restore performance versus backup version on the Less dataset with an LRU cache of size 512MB.

because it incurs the largest computation overhead for resemblance detection. It is noteworthy that DARE's average data-reduction throughput on RAID, at 50MB/s, is much lower than DARE's average throughput of 85MB/s on SSD. The root cause of RAID's inferior data-reduction performance (in Figure 6(a)) mainly lies in the random reads of the base-chunks. In general, DARE achieves superior performance of both throughput and data reduction efficiency among all the resemblance detection approaches as indicated in Figure 6 and Table 4.

4.5 Restoration Performance

Intuitively, delta compression should slow down the data-restore performance of a data-reduction system since it needs to restore the resembling chunks by two reads, one for the delta data and the other for the base-chunk, and then delta decode them. But in our evaluation of the restore operations for resembling chunks, we find that the speed of delta decode (i.e., Xdelta [8]) tends to be very fast, about 1GB/s in the DARE system. Another interesting observation is that, for a restoration cache of a given size with a delta chunk $\Delta_{i,k}$ and its based chunk C_i, DARE actually caches more logical content of the two chunks C_i and C_k than a deduplication-only system and thus improves the data-restore performance by virtual of the enlarged restoration cache due to delta compression.

Figures 7(a) and 8(a) show that DARE on average doubles the data-restore speed of the deduplication-only system (both running on the RAID). Figures 7(b) and 8(b) clearly show that the reason lies in the fact that DARE reads half as many containers for restoration as the deduplication-only system. The superior data-restore performance of SF-2F and SF-4F to the deduplication-only system is attributed to their data reduction efficiency (see Tables 3 and 4). Since the restore-performance for the other six datasets is similar to and consistent with that of the Less dataset, they are omitted to save space. The sudden increase in the data-restore performance of the deduplication-only approach at the backup version 17 (Figure 8(a)), we observe, is due to the fact that most of the backed-up sources targeted for restoration are from the current and recent backups and thus have fewer random reads for restoration.

5 Conclusion and Future Work

In this paper, we present DARE, a deduplication-aware, low-overhead resemblance detection and elimination scheme for delta compression on the top of deduplication on backup datasets. DARE uses a novel resemblance detection approach, DupAdj, which exploits the duplicate-adjacency information for efficient resemblance detection in existing deduplication systems, and employs an improved super-feature approach to further detecting resemblance when the duplicate-adjacency information is lacking or limited.

Our preliminary results on the data-restore performance suggest that supplementing delta compression to deduplication can effectively enlarge the logical space of the restoration cache, but the data fragmentation in data reduction systems remains a serious problem [19]. We plan to further study and improve the data-restore performance of storage systems based on deduplication and delta compression in our future work.

Acknowledgments

This work was supported in part by National Basic Research 973 Program of China under Grant No. 2011CB302301; NSFC No. 61025008, 61173043, and 61232004; 863 Project 2013AA013203; US NSF under Grants IIS-0916859, CCF-0937993, CNS-1116606, and CNS-1016609. This work was also supported by Key Laboratory of Information Storage System, Ministry of Education, China.

References

[1] G. Wallace, F. Douglis, H. Qian, P. Shilane, S. Smaldone, M. Chamness, and W. Hsu, "Characteristics of backup workloads in production systems," in *Proc. USENIX FAST*, 2012.

[2] P. Shilane, M. Huang, G. Wallace, and W. Hsu, "WAN optimized replication of backup datasets using stream-informed delta compression," in *Proc. USENIX FAST*, 2012.

[3] A. Muthitacharoen, B. Chen, and D. Mazieres, "A low-bandwidth network file system," in *Proc. ACM SOSP*, 2001.

[4] C. Constantinescu, J. Glider, and D. Chambliss, "Mixing deduplication and compression on active data sets," in *Data Compression Conference (DCC), 2011*. IEEE, 2011, pp. 393–402.

[5] B. Zhu, K. Li, and H. Patterson, "Avoiding the disk bottleneck in the data domain deduplication file system," in *Proc. USENIX FAST*. USENIX Association, 2003.

[6] J. Gailly and M. Adler, "The gzip compressor," http://www.gzip.org/, 1991.

[7] P. Kulkarni, F. Douglis, J. LaVoie, and J. Tracey, "Redundancy elimination within large collections of files," in *USENIX Annual Technical Conference*. USENIX Association, 2004.

[8] J. MacDonald, "File system support for delta compression." Masters thesis. Department of Electrical Engineering and Computer Science, University of California at Berkeley., 2000.

[9] S. Quinlan and S. Dorward, "Venti: a new approach to archival storage," in *Proc. USENIX FAST*, 2002.

[10] F. Douglis and A. Iyengar, "Application-specific delta-encoding via resemblance detection," in *Proc. USENIX FAST*. USENIX Association, 2003.

[11] L. Aronovich, R. Asher, E. Bachmat, H. Bitner, M. Hirsch, and S. Klein, "The design of a similarity based deduplication system," in *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference*. ACM, 2009.

[12] M. Rabin, *Fingerprinting by random polynomials*. Center for Research in Computing Techn., Aiken Computation Laboratory, Univ., 1981.

[13] D. Gupta, S. Lee, M. Vrable, S. Savage, A. C. Snoeren, G. Varghese, G. M. Voelker, and A. Vahdat, "Difference engine: harnessing memory redundancy in virtual machines," in *Proc. USENIX OSDI*, 2008.

[14] Q. Yang and J. Ren, "I-CASH: Intelligently coupled array of ssd and hdd," in *Proc. IEEE HPCA*, 2011.

[15] A. Broder, "Identifying and filtering near-duplicate documents," in *Combinatorial Pattern Matching*, 2000.

[16] ——, "Some applications of Rabins fingerprinting method," in *Sequences II: Methods in Communications, Security, and Computer Science*, 1993.

[17] ——, "On the resemblance and containment of documents," in *Compression and Complexity of Sequences 1997*.

[18] V. Tarasov, A. Mudrankit, W. Buik, P. Shilane, G. Kuenning, and E. Zadok, "Generating realistic datasets for deduplication analysis," in *USENIX Annual Technical Conference*, 2012.

[19] M. Lillibridge, K. Eshghi, and D. Bhagwat, "Improving restore speed for backup systems that use inline chunk-based deduplication," in *Proc. USENIX FAST*, 2013.

[20] 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 *USENIX Annual Technical Conference*, 2011.