International Journal of Recent Trends in Engineering & Research
ISSN (ONLINE) : 2455 – 1457
IMPACT FACTOR : 4.101
National Conference on Convergence of Emerging Technologies in Computer Science & Engineering (CETCSE-2K17)

DARE: A Deduplication-Aware Resemblance Detection and Elimination Scheme for Data Reduction with Low Overheads

Dr.K.Srinivas Babu[*1] **L.Kalyani**[*2]
[*1]Professor Department of Computer Science and Engineering [*2]M.Tech

***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. One of the main challenges facing large-scale data reduction is how to maximally detect and eliminate redundancy at very low overheads. In this paper, we present DARE, a low-overhead deduplication-aware resemblance detection and elimination scheme that effectively exploits existing duplicate-adjacency information for highly efficient resemblance detection in data deduplication based backup/archiving storage systems. 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 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 only consumes about 1/4 and 1/2 respectively of the computation and indexing overheads required by the traditional super-feature approaches while detecting 2-10 percent more redundancy and achieving a higher throughput, by exploiting existing duplicate-adjacency information for resemblance detection and finding the "sweet spot" for the super-feature approach**

Keywords— Data deduplication, delta compression, storage system, index structure, performance evaluation

I. INTRODUCTION

The amount of digital data is growing explosively, as evidenced in part by an estimated amount of about 1.2 zet- tabytes and 1.8 zettabytes respectively of data produced in 2010 and 2011 [1], [2]. As a result of this "data deluge", man- aging storage and reducing its costs have become one of the most challenging and important tasks in mass storage sys- tems. According to a recent IDC study [3], almost 80 percent of corporations surveyed indicated that they were exploring data deduplication technologies in their storage systems to increase storage efficiency.

Data deduplication is an efficient data reduction approach that not only reduces storage space [4], [5], [6], [7], [8], [9], [10] by eliminating duplicate data but also mini- mizes the transmission of redundant data in low-bandwidth network environments [11], [12], [13], [14]. In general, a chunk-level data deduplication scheme splits data blocks of a data stream (e.g., backup files, databases, and virtual machine images) into multiple data chunks that are each uniquely identified and duplicate-detected by a secure SHA-1 or MD5 hash signature (also called a fingerprint) [5], [11]. Storage systems then remove duplicates of data chunks and store only one copy of them to achieve the goal of space savings.

While data deduplication has been widely deployed in storage systems for space savings, the fingerprint-based dedu- plication approaches have an inherent drawback: they often fail to detect the similar chunks that are largely identical except for a few modified bytes, because their secure hash digest will be totally different even only one byte of a data chunk was changed [4], [5], [12], [15], [16]. It becomes a big challenge when applying data deduplication to storage data- sets and workloads that have frequently modified data, which demands an effective and efficient way to eliminate redun- dancy among frequently modified and thus similar data.

Delta compression, an efficient approach to removing redundancy among similar data chunks has gained increas- ing attention in storage systems [12], [17], [18], [19], [20]. For example, if chunk A2 is similar to chunk A1 (the base- chunk), the delta compression approach calculates and then only stores the differences (delta) and mapping relation between A2 and A1. Thus, it is considered a promising tech- nique that effectively complements the fingerprint-based deduplication approaches by detecting similar data missed by the latter.

One of the main challenges facing the application of delta compression in deduplication systems is how to accurately detect the most similar candidates for delta compression with low overheads. The state-of-

International Journal of Recent Trends in Engineering & Research
ISSN (ONLINE) : 2455 – 1457
IMPACT FACTOR : 4.101
National Conference on Convergence of Emerging Technologies in Computer Science & Engineering (CETCSE-2K17)

the-art solutions [12], [15], [16], [17] detect similarity for delta compression by computing sev- eral Rabin fingerprints as features and grouping them into super-fingerprints, also referred to as super-features (SF) (detailed in Section 3.3). Nevertheless, to index a dataset of 80 TB and assuming an average chunk size of 8 KB and 16 bytes per index entry, for example, about 200 GB worth of super-feature index entries must be generated, which will still be too large to fit in memory [12]. Since the random accesses to on-disk index are much slower than that to RAM, the frequent accesses to on-disk super-features will cause the system throughput to become unacceptably low for the users [6], [12], [21].

The existing solutions to the indexing issue of delta com- pression either record the resemblance information for files, instead of data chunks, so that similarity index entries can fit in the memory [22], [23], or exploit the locality of backup data streams in deduplication-based backup/archiving sys- tems, which avoids the global indexing on the disk [12], [17]. The first approach faces an implementation difficulty in large-scale data deduplication systems since it is hard to record all the resemblance or version information of files in such systems [12]. The second approach often fails to detect a significant amount of redundant data when the workloads lack locality. Another challenge facing the super-feature method is the high overhead in computing the super-fea- tures. According to a recent study of delta compression [17] and our experimental observation, the throughput of com- puting super-features is about 30 MB/s (see Section 3.3 for details), which may become a potential bottleneck for dedu- plication-based storage systems, particularly if most index entries are fit in memory or partially on SSD-based storage for which the throughput can be hundreds of MB per second or higher.

From our observation of duplicate and similar data of backup streams, we find that the non-duplicate chunks that are adjacent to duplicate ones could be considered good delta compression candidates in data deduplication systems. Thus we propose the approach of Duplicate- Adjacency based Resemblance Detection, or DupAdj for short. Exploiting this existing deduplication information (i.e., duplicate-adjacency) not only avoids the high overhead of super-feature computation but also reduces the size of index entries for resemblance detection. On the other hand, our study of the existing super-feature approaches reveals that the traditional super-feature method can be improved with fewer features per super-feature, which works very effectively on deduplication systems when combined with the aforementioned DupAdj approach.

In this paper, we propose DARE, a low-overhead Dedu- plication-Aware Resemblance detection and Elimination scheme for deduplication based backup and archiving stor- age system. The main idea of DARE is to effectively exploit existing duplicate-adjacency information to detect similar data chunks (DupAdj), refine and supplement the detection by using an improved super-feature approach (Low-Over- head Super-Feature) when the existing duplicate-adjacency information is lacking or limited. In addition, we present an analytical study of the existing super-feature approach with a mathematic model and conduct an empirical evaluation of this approach with several real-world workloads in data deduplication systems.

Our experimental evaluation results, based on real-world and synthetic backup datasets, show that DARE signifi- cantly outperforms the traditional Super-Feature approach. More specifically, the DupAdj approach achieves a similar data reduction efficiency to the pure super-feature approach and DARE detects 2-10 percent more redundant data while achieving a higher throughput of data reduction than the pure super-feature approach. Meanwhile, DARE only con- sumes about 1/4 and 1/2 respectively of the computation and indexing overheads required by the traditional super- feature approach for resemblance detection. It is important to note that our evaluation also demonstrates the superior data-restore performance of the DARE-enhanced dedupli- cation system over the deduplication-only systems via delta compression, where the former outperforms the latter by a factor of 2 (2×).

II. BACKGROUND AND MOTIVATION

In this section, we first present the necessary background knowledge about resemblance detection for data reductions in storage systems, then provide analytical and experimen- tal observations that motivate our research on resemblance detection for data reduction.

Resemblance Detection Based Data Reduction Data deduplication is becoming increasingly popular 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 com-mercial backup and archiving storage systems [6], [24], [25], [26], [27].

Previous studies on data deduplication have shown that one challenge lies in the system scalability issue of index-lookup. That is, the

International Journal of Recent Trends in Engineering & Research
ISSN (ONLINE) : 2455 – 1457
IMPACT FACTOR : 4.101
National Conference on Convergence of Emerging Technologies in Computer Science & Engineering (CETCSE-2K17)

fingerprints of a multi-TB-scale storage system will be too large to fit in memory and must be moved to the disk, which causes long latencies of random disk I/Os for fingerprint index-lookup. Most existing solutions to this problem aim to make full use of RAM, by putting only the hot fingerprints into RAM to reduce accesses to on-disk index. DDFS [6] and Sparse Indexing [25] attempt to avoid the disk bottleneck for deduplication indexing by exploiting the inherent locality of the backup streams and preserving this locality in the memory to increase cache hit ratio. Locality here means that the chunks of a backup stream will appear in approx- imately the same order in each full backup with a high probability. Extreme Binning [28] and SiLo [29] exploits similarity-only and similarity & locality of the backup data streams respectively to minimize RAM overhead for deduplication index-lookup. ChunkStash puts the finger- print index on SSD by means of a memory-efficient data structure called cuckoo hash to accelerate index-lookup for data deduplication [21].

TABLE 1
Comparisons between Duplicate Detection and Resemblance Detection for Data Reduction Systems

	Duplicate Detection	Resemblance Detection
Objects	Duplicate data	Similar data
Granularity	Chunk-level	Byte-level
Rep. Methods	Secure-Fingerprint based Deduplication	Super-Feature based Delta Compression
Scalability	Strong	Weak
Rep. Systems	LBFS [11], Venti[5], DDFS [6]	REBL[16], DERD[15], SIDC[12]

Another challenge for data deduplication is how to maxi- mally detect and eliminate data redundancy in storage sys- tems by determining appropriate data chunking schemes. 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 [9], [11], [30]. Re-chunking approaches were also proposed to divide those non-dupli- cate chunks into smaller ones to expose and detect more redundancy [31], [32], [33].

Resemblance detection with delta compression[15], [16], [26], as another approach to data reduction in storage sys- tems, was proposed more than 10 years ago but was later overshadowed by fingerprint-based deduplication [6], [24], [25] due to the former's scalability issue. Table 1 compares these two data reduction approaches. Resemblance detec- tion 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 mass storage systems.

REBL[16] and DERD [15] are typical super-feature-based resemblance detection approaches for data reduction. They compute the features of the data stream (e.g., Rabin Finger- prints [34]) and group features into super-features to cap- ture the resemblance of data and then delta compress the data. TAPER [35] presents a Bloom-Filter solution that measures the similar files based on the chunk fingerprints recorded in Bloom Filters. All these approaches require high computation and indexing overheads for resemblance detection. As a result, the simpler and faster deduplication method has become a more popular data reduction approach in the last five years [6], [7], [8].

Nevertheless, resemblance detection is gaining increas- ing tractions in storage systems because of its ability to capture and eliminate data redundancy among similar but non-duplicate data chunks that effectively comple- ments fingerprint-based deduplication. Difference Engine

[20] employs Xdelta [23] to further eliminate memory redundancy and thus enlarge the logical RAM space in VM environments. I-CASH [18] delta compresses similar data to enlarge the logical space of SSD caches. Shilane et al. [12] proposed a stream-informed delta compression (SIDC) approach to reducing similar data transmission and thus accelerating data replication in a WAN

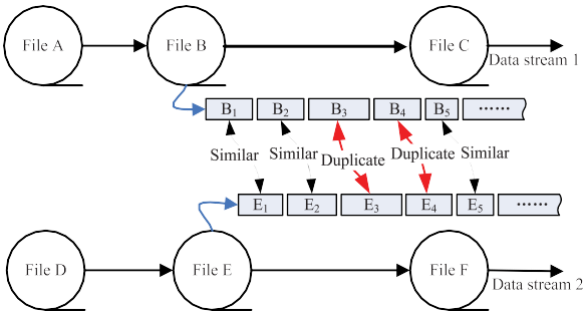


Fig. 1. A conceptual illustration of the duplicate adjacency. The non- duplicate chunks adjacent to duplicate ones are considered potentially similar and thus good delta compression candidates.

environment. This approach is super-feature based and complements the chunk-level deduplication by only detecting resemblance among non-duplicate chunks in the cache that preserves the backup stream

International Journal of Recent Trends in Engineering & Research
ISSN (ONLINE) : 2455 – 1457
IMPACT FACTOR : 4.101
National Conference on Convergence of Emerging Technologies in Computer Science & Engineering (CETCSE-2K17)

locality. It avoids the costly global indexing, at a limited loss of resemblance detection. While the combined detection of duplicate and resemblance promises to achieve a superior data reduction performance, challenges of relatively high computation and indexing overheads stemming from resemblance detection remain [17].

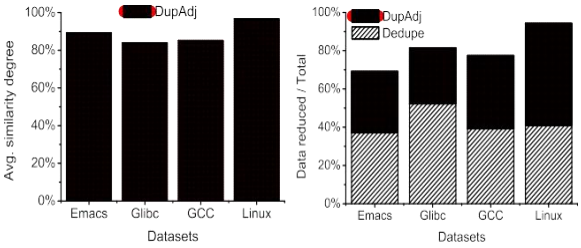
Note that SIDC [12] is the most related work to DARE. Different from SIDC that implements traditional super- feature based delta compression in a stream-informed (i.e., locality preserved) cache, DARE first employs a duplicate-adjacency based resemblance detection scheme (see Section 3.2) and then an improved super-feature based approach (see Section 3.3) to jointly and more effec- tively reduce the indexing and computation overheads for delta compression.

Fact of Duplicate Adjacency

As discussed in Section 2.1, 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. For those non-duplicate chunks that are location-adjacent to known duplicate data chunks in a deduplication system, it is intuitive and quite possible that only a few bytes of them are modified from the last backup, making them potentially excellent delta compression candidates.

Fig. 1 illustrates a case of duplicate data chunks and their immediate non-duplicate neighbors. As mentioned above, our intuition is that the latter are highly likely to be similar and thus good delta compression candidates. Specifically, since chunks B3 & B4 are duplicates of chunks E3 & E4 in Fig. 1 respectively, their immediate neighbors, the chunk-pairs B1 & E1, B2 & E2, and B5 & E5, are then considered good delta compression candidates, which is consistent with the aforementioned backup-stream locality [6], [12], [25], [29], [36].

If we can make full use of the existing knowledge about duplicate data chunks in a deduplication system, it is pos- sible for us to detect similar chunks without the overheads of computing and storing features & super-features and then accessing their on-disk index. Fig. 2 shows important preliminary results of this duplicate-adjacency-based resemblance detection approach, called DupAdj, on sev- eral real-world datasets whose workload characteristics are detailed in Table 2 in Section 4.1. First, the similarity degree (i.e., dela compressed size) of the DupAdj-detected chunks tends to be very high, on average, about 84-96 per- cent on the four backup datasets as shown in Fig. 2a. Sec- ond, by exploiting this duplicate adjacency information, the DupAdj-based post deduplication delta compression approach can further detect and eliminate about 30-50 per- cent redundancy from the non-duplicate but duplicate-adjacent chunks as shown in Fig. 2b. Hence, this DupAdj approach, detailed in Section 3.2, is very effective for detecting possible similar chunks and then delta encoding them to further remove redundancy in deduplication- based backup systems while significantly simplifying the resemblance detection process.



(a) Similarity degree of the DupAdj detected chunks

(b) Redundancy reduced by DupAdj-based delta compression

Fig. 2. A study of redundancy elimination on the four real-world tarred datasets by 8KB-level deduplication and then DupAdj-based delta compression.

Rethinking of the Super-Feature Approaches Similar data, like duplicate data, are in wide existence in backup systems [8], [17]. Meister and Brinkmann [37] find that small semantic changes on documents may result in big modifications in the binary representation of files, and delta compression is more effective in eliminating redun- dancy in such cases. To support delta compression, resem- blance detection will be required for selecting suitable similar candidates.

Some early research on resemblance detection of near- duplicate or similar files was performed by Broder for search engine results [38], [39] and Manber for filesystems [40]. REBL [16] and DERD [15] used an efficient super- feature approach to eliminating redundancy with delta compression in the early data reduction systems. How- ever, their super-feature approaches are arguably very different from the most recently required resemblance detection in the current large-scale storage systems in that:

- The datasets used in REBL and DERD are more likely of primary storage workloads and only several hundreds of MB in size, in contrast to typical data- sets of deduplication sytems that are usually of the TB/PB scale [8], [12], [41].

- The chunk size tested in their papers is in the 1-4 KB range and hashing region of each feature is of 4-22 bytes, in contrast to typical deduplication systems that adopt the larger chunk size of 8 KB and

International Journal of Recent Trends in Engineering & Research
ISSN (ONLINE) : 2455 – 1457
IMPACT FACTOR : 4.101
National Conference on Convergence of Emerging Technologies in Computer Science & Engineering (CETCSE-2K17)

larger feature hashing region of 32 or 48 bytes (i.e., CDC sliding windows) [6], [8], [11].

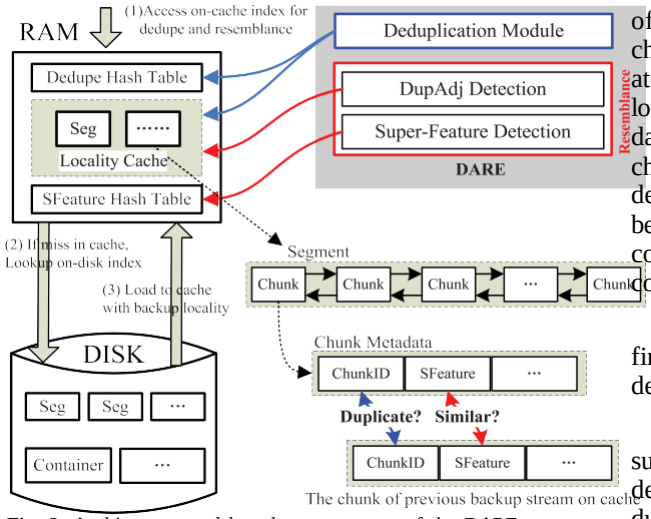


Fig. 3. Architecture and key data structures of the DARE system that combines duplicate detection and resemblance detection for data reduction.

- The post-deduplication chunks tend to be more fre- quently modified in backup systems, which may make the resemblance of these non-duplicate and less similar chunks more difficult to detect.

III. DESIGN AND IMPLEMENTATION

In this section, we will first describe the architecture and key data structures of DARE, followed by detailed discussions of its design and implementation issues.

Architecture Overview

DARE is designed to improve resemblance detection for additional data reduction in deduplication-based backup/ archiving storage systems. As shown in Fig. 3, the DARE architecture consists of three functional modules, namely, the Deduplication module, the DupAdj Detection module, and the improved super-feature module. In addition, there are five key data structures in DARE, namely, Dedupe Hash Table, SFeature Hash Table, Locality Cache, Container, Seg- ment, and Chunk, which are defined below:

- A chunk is the atomic unit for data reduction. The non-duplicate chunks, identified by their SHA-1 fin- gerprints, will be prepared for resemblance detection in DARE.

- A container is the fixed-size storage unit that stores sequential and NOT reduced data, such as non-duplicate & non-similar or delta chunks, for better storage performance by using large I/Os [6], [36].

- A segment consists of the metadata of a number of sequential chunks (e.g., 1 MB size), such as the chunk fingerprints, size, etc., which serves as the atomic unit in preserving the backup-stream logi- cal locality [36] for data reduction. Here DARE uses a data structure of doubly-linked list to record the chunk adjacency information for the DupAdj detection. Note that the SFeature in the segment may be unnecessary if the DupAdj module has already confirmed this chunk as being similar for delta compression.

- Dedupe hash table serves to index fingerprints for duplicate detection for the deduplication module.

- SFeature hash table serves to index the super-features after the DupAdj resemblance detection. It manages the super-features of non-duplicate and non-similar chunks.

- Locality cache contains the recently accessed data seg- ments and thus preserves the backup-stream locality in memory, to reduce accesses to the on-disk index from either duplicate detection or resemblance detection.

Here we describe a general workflow of DARE. For the input data stream, DARE will first detect duplicate chunks by the Deduplication module. Any of the many existing dedu- plication approaches [36] can be implemented here and the preservation of the backup-stream logical locality in the seg- ments is required for further resemblance detection. For each non-duplicate chunk, DARE will first use its DupAdj Detection module (see Section 3.2) to quickly determine whether it is a delta compression candidate. If it is not a can- didate, DARE will then compute its features and super-fea- tures, using its improved Super-Feature Detection module (see Section 3.3), to further detect resemblance for data reduction.

Because DARE adopts a caching scheme that exploits the backup-stream logical locality [36] in a way similar to the Sparse Indexing [25], SiLo [29], and BLC [42] approaches, the indexing hit ratio in the locality cache for both the deduplication and resemblance detection mod- ules will be very high. Upon a miss in the locality cache, DARE will load the missing segment from the latest backup to the RAM with the LRU replacement policy. It is noteworthy that, after deduplication, the cached seg- ments that have preserved the logical-locality of chunks, including the adjacency information of the duplicate-detected chunks, will be further exploited by DARE to detect possi- ble resemblance among the

International Journal of Recent Trends in Engineering & Research
ISSN (ONLINE) : 2455 – 1457
IMPACT FACTOR : 4.101
National Conference on Convergence of Emerging Technologies in Computer Science & Engineering (CETCSE-2K17)

non-duplicate data chunks, as detailed in the next section.

Duplicate-Adjacency Based Resemblance Detection

As a salient feature of DARE, the DupAdj approach detects resemblance by exploiting existing duplicate-adjacency information of a deduplication system. The main idea behind this approach is to consider chunk pairs closely adjacent to any confirmed duplicate-chunk pair between two data streams as resembling pairs and thus candidates for delta compression, as conceptually illustrated in Fig. 1.

According to the description of the DARE data structures in Fig. 3, DARE records the backup-stream logical locality of chunk sequence by a doubly-linked list, which allows an efficient search of the duplicate-adjacent chunks for resem- blance detection by traversing to prior or next chunks on the list, as shown in Fig. 1. When the DupAdj Detection module of DARE processes an input segment, it will traverse all the chunks by the aforementioned doubly-linked list to find the already duplicate-detected chunks. If chunk Am of the input segment A was detected to be a duplicate of chunk Bn of segment B, DARE will traverse the doubly- linked list of Bn in both directions (e.g., Amþ1 & Bnþ1 and Am—1 & Bn—1) in search of potentially similar chunk pairs between segments A and B, until a dissimilar chunk or an already detected duplicate or similar chunk is found. Note that the detected chunks here are considered dissimilar (i.e., NOT similar) to others if their similarity degree (i.e., dela compressed size) is smaller than a predefined threshold, such as 0.25, a false positive for resemblance detection. Actually, the similarity degree of the DupAdj-detected chunks tends to be very high, larger than 0.88, as shown in Fig. 2 in Section 2.2.

In general, the overheads for the DupAdj based approach are twofold:

- Memory overhead: Each chunk will be associated with two pointers (about 8 or 16 Bytes) for building the doubly-linked list when DARE loads the segment into the locality cache. But when the segment is evicted from the cache, the doubly-linked list will be immediately freed. Therefore, this RAM memory overhead is arguably negligible given the total capacity of the locality cache.

- Computation overhead: Confirming the similarity degree of the DupAdj-detected chunks may intro- duce additional but ommitted computation over- head. First, the delta encoding results for the confirmed resembling (i.e., similar) chunks will be directly used as the final delta chunk for storage. Second, the actual extra computation overhead occurs when the DupAdj-detected chunks are NOT similar, which is a very rare event as discussed in the previous paragraph.

In all, the DupAdj detection approach only adds a dou- bly-linked list to an existing deduplication system, DARE avoids the computation and indexing overheads of the con- ventional super-feature approach. In case where the dupli- cate-adjacency information is lacking, limited, or interrupted due to operations such as file content inser- tions/deletions or new file appending, DARE will use an improved super-feature approach to further detect and eliminate resemblance as discussed in the next section.

IV. CONCLUSION

In this paper, we present DARE, a deduplication-aware, low-overhead resemblance detection and elimination scheme for data reduction in backup/archiving storage systems. DARE uses a novel approach, DupAdj, which exploits the duplicate-adjacency information for efficient resemblance detection in existing deduplication systems, and employs an improved super-feature approach to fur- ther detecting resemblance when the duplicate-adjacency information is lacking or limited.

Results from experiments driven by real-world and syn- thetic backup datasets suggest that DARE can be a powerful and efficient tool for maximizing data reduction by further detecting resembling data with low overheads. Specifically, DARE only consumes about 1/4 and 1/2 respectively of the computation and indexing overheads required by the tradi- tional super-feature approaches while detecting 2-10 per- cent more redundancy and achieving a higher throughput. Furthermore, the DARE-enhanced data reduction approach is shown to be capable of improving the data-restore perfor- mance, speeding up the deduplication-only approach by a factor of 2(2X) by employing delta compression to further eliminate redundancy and effectively enlarge the logical space of the restoration cache.

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

REFERENCES

[1] The data deluge [Online]. Available: http://econ.st/fzkuDq

International Journal of Recent Trends in Engineering & Research
ISSN (ONLINE) : 2455 – 1457
IMPACT FACTOR : 4.101
National Conference on Convergence of Emerging Technologies in Computer Science & Engineering (CETCSE-2K17)

[2] J. Gantz and D. Reinsel, "Extracting value from chaos," IDC Rev., vol. 1142, pp. 1–12, 2011.

[3] L. DuBois, M. Amaldas, and E. Sheppard, "Key considerations as deduplication evolves into primary storage," White Paper 223310, Framingham, MA, USA: IDC, Mar. 2011.

[4] W. J. Bolosky, S. Corbin, D. Goebel, and J. R. Douceur, "Single instance storage in windows 2000," in Proc. 4th USENIX Windows Syst. Symp., Aug. 2000, pp. 13–24.

[5] S. Quinlan and S. Dorward, "Venti: A new approach to archival storage," in Proc. USENIX Conf. File Storage Technol., Jan. 2002, pp. 89–101.

[6] B. Zhu, K. Li, and R. H. Patterson, "Avoiding the disk bottleneck in the data domain deduplication file system," in Proc. 6th USE- NIX Conf. File Storage Technol., Feb. 2008, vol. 8, pp. 1–14.

[7] D. T. Meyer and W. J. Bolosky, "A study of practical deduplication," ACM Trans. Storage, vol. 7, no. 4, p. 14, 2012.

[8] G. Wallace, F. Douglis, H. Qian, P. Shilane, S. Smaldone, M. Chamness, and W. Hsu, "Characteristics of backup workloads in production systems," in Proc. 10th USENIX Conf. File Storage Tech- nol., Feb. 2012, pp. 33–48.

[9] A. El-Shimi, R. Kalach, A. Kumar, A. Ottean, J. Li, and S. Sengupta, "Primary data deduplication-large scale study and system design," in Proc. Conf. USENIX Annu. Tech. Conf., Jun. 2012, pp. 285–296.

[10] L. L. You, K. T. Pollack, and D. D. Long, "Deep store: An archival storage system architecture," in Proc. 21st Int. Conf. Data Eng., Apr. 2005, pp. 804–815.

[11] A. Muthitacharoen, B. Chen, and D. Mazieres, "A low-bandwidth network file system," in Proc. ACM Symp. Oper. Syst. Principles., Oct. 2001, pp. 1–14.

[12] P. Shilane, M. Huang, G. Wallace, and W. Hsu, "WAN optimized replication of backup datasets using stream-informed delta compression," in Proc. 10th USENIX Conf. File Storage Technol., Feb. 2012, pp. 49–64.

[13] S. Al-Kiswany, D. Subhraveti, P. Sarkar, and M. Ripeanu, "Vmflock: Virtual machine co-migration for the cloud," in Proc. 20th Int. Symp. High Perform. Distrib. Comput., Jun. 2011, pp. 159–170.

[14] X. Zhang, Z. Huo, J. Ma, and D. Meng, "Exploiting data dedupli- cation to accelerate live virtual machine migration," in Proc. IEEE Int. Conf. Cluster Comput., Sep. 2010, pp. 88–96.

[15] F. Douglis and A. Iyengar, "Application-specific delta-encoding via resemblance detection," in Proc. USENIX Annu. Tech. Conf., General Track, Jun. 2003, pp. 113–126.

[16] P. Kulkarni, F. Douglis, J. D. LaVoie, and J. M. Tracey, "Redundancy elimination within large collections of files," in Proc. USENIX Annu. Tech. Conf., Jun. 2012, pp. 59–72.

[17] P. Shilane, G. Wallace, M. Huang, and W. Hsu, "Delta compressed and deduplicated storage using stream-informed locality," in Proc. 4th USENIX Conf. Hot Topics Storage File Syst., Jun. 2012, pp. 201–214.

[18] Q. Yang and J. Ren, "I-cash: Intelligently coupled array of SSD and HDD," in Proc. 17th IEEE Int. Symp. High Perform. Comput. Archit., Feb. 2011, pp. 278–289.

[19] G. Wu and X. He, "Delta-FTL: Improving SSD lifetime via exploit- ing content locality," in Proc. 7th ACM Eur. Conf. Comput. Syst., Apr. 2012, pp. 253–266.

[20] 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. 5th Symp. Oper. Syst. Design Implementation., Dec. 2008, pp. 309–322.

