Boosting the Profitability of NVRAM-based Storage Devices via the Concept of Dual-Chunking Data Deduplication

Shuo-Han Chen[1], Yu-Pei Liang[2], Yuan-Hao Chang[1], Hsin-Wen Wei[3], and Wei-Kuan Shih[2]
[1]Academia Sinica, Institute of Information Science, Taipei, Taiwan
[2]National Tsing Hua University, Department of Computer Science, Hsinchu, Taiwan
[3]Tamkang University, Department of Electrical and Computer Engineering, New Taipei City, Taiwan

Abstract— With the latest advance in the non-volatile random-access memory (NVRAM), NVRAM is widely considered as the mainstream for the next-generation storage mediums. NVRAM has numerous attractive features, which include byte addressability, limited idle energy consumption, and great read/write access speed. However, owing to the high manufacturing cost of NVRAM, the incentive of deploying NVRAM in consumer electronics is lowered due to the consideration of profitability. To resolve the profitability issue and bring the benefits of NVRAM into the design of consumer electronics, avoiding storing duplicate data on NVRAM becomes a crucial task for lowering the demand and deployment cost of NVRAM. Such observation motivates us to propose a data deduplication extended file system design (DeEXT) to boost the profitability of NVRAM via the concept of dual-chunking data deduplication while considering the characteristics of NVRAM and duplicate data content. The proposed DeEXT was then evaluated by real-world data deduplication traces with encouraging results.

I. Introduction

In recent years, non-volatile random-access memory (NVRAM) has become a reality with the efforts of both academia and industry. Markedly, Intel has been co-working with Micron for formulating a new NVRAM technology, called 3D xPoint [8], and has also released their first NVRAM-based solid state drive (SSD) for the consumer market. However, the cost of deploying NVRAM as the storage medium in consumer electronics is typically higher than that of NAND flash-based storage. For instance, the 3D xPoint-based SSD is roughly 3 times more expensive than the NAND flash-based competitor [9]. Nevertheless, NVRAM can still bring great benefits to the consumer electronics since the access speed of NVRAM is much higher that of NAND flash. Therefore, to improve the profitability of NVRAM-based electronics, avoiding storing duplicate data on NVRAM-based storage devices becomes a crucial task for lowering the cost. With such observation in mind, a data deduplication extended file system design (DeEXT) is proposed to enable dual-chunking data deduplication on NVRAM for improving the profit per storage unit. The technical difficulty of DeEXT lies in *how to integrate data deduplication into NVRAM-based storage systems while considering the duplicate data content and the wearing issue of NVRAM.*

NVRAM, such as Phase Change Memory (PCM) and Resistive RAM (ReRAM), are regarded as great alternatives for replacing hard disk-based or NAND flash-based storage devices owing to NVRAM's excellent characteristics. Table I compares the features of NVRAM, NAND flash, and HDD. More recently, 3D xPoint memory, which is a kind of PCM, has emerged as another type of NVRAM. According to [11], the performance/endurance of 3D xPoint memory is 1,000 times higher than that of conventional NAND flash. Various studies have been proposed to exploit the potential benefits of NVRAM, such as new caching mechanisms [10], performance optimization strategies [13], file system designs [23], and data updating policies [5]. From another perspective, researchers have also tried to lower the storage space usage of NVRAM via data deduplication mechanisms [20]. However, previous studies mainly focus on how to improve the data deduplication ratio[1]. Few of them have discussed *how to achieve wear-level-aware data deduplication into NVRAM-based storage systems and consider the characteristics of the duplicate data content.*

TABLE I
Comparison of memory/storage mediums [3, 4, 2].

Type	PCM	RRAM	NAND	HDD
Byte-addressable	Yes	Yes	No	No
Access Unit	64 Bytes	64 Bytes	4 KB	512 B
Endurance	10^{10}	10^{8}	10^{5}	N/A
Read energy	1 J/GB	0.25 J/GB	1.5 J/GB	65 J/GB
Write energy	6 J/GB	14.02 J/GB	17.5 J/GB	65 J/GB
Read Latency	50 ns	1.9 ns	25 μs	5 ms
Write Latency	150 ns	100 ns	500 μs	5 ms

Data deduplication is a kind of storage compression techniques and is mainly used to eliminate duplicate data copies within a storage system. Data deduplication techniques are widely adopted by cloud storage service companies, such as Dropbox and Microsoft OneDrive, due to the high duplicate nature of data within storage systems. For instance, Microsoft and EMC respectively estimate that 50% and 85% of data in their storage systems are duplicate [14, 19]. Typically, data deduplication technique labels each data chunk with a fingerprint generated by hash functions, such as MD5 and SHA-256, and compares with other fingerprints to check whether the data chunk is duplicate or not. The methods for dividing data streams into smaller data chunks include *fixed-size chunking* (FSC) [17] and *content defined chunking* (CDC) [15], where CDC can achieve higher deduplication ratio than FSC. However, since CDC actually checks the data content before chunking, the chunking overhead of CDC is higher than that of FSC.

On the other hand, previous data deduplication mechanisms can be roughly classified into two major categories, including *inline deduplication* and *post-processing deduplication*. Inline deduplication eliminates redundant data before data

[1]The term "data deduplication ratio" refers to the ratio of the data amount before deduplication to the data amount after redundancy removal.

978-1-7281-4123-7/20/$31.00

are actually written to storage devices, while post-processing deduplication eliminates duplicate data chunks after data are written to the storage devices. Since inline deduplication can reduce the amount of data written to storage devices, inline deduplication is more suitable for lifetime-limited NVRAM. Based on the concept of inline deduplication, previous studies mainly focused on optimizing the deduplication ratio and fingerprint indexing[2] performance for conventional block-based [21, 22, 18] and NVRAM-based byte-addressable storage devices [20]. Although these studies can further enhance the deduplication ratio and improve indexing performance, few of them have considered the design issues of utilizing the characteristics of the duplicate data content to facilitate data deduplication design on NVRAM-based storage.

To facilitate data deduplication by considering the characteristics of duplicate data content and to improve the profitability of NVRAM, this study is a pioneer work that attempts to realize the concept of dual-chunking data deduplication on NVRAM via proposing the design of data deduplication extended file system (DeEXT). Different from previous optimization approaches, this study focuses on identifying and utilizing the inherent data deduplication potentiality[3] of a data stream to lower the chunking overhead for data with low deduplication potentiality and maintain the redundancy removal ability for data with high deduplication potentiality. To facilitate the concept of dual-chunking data deduplication within NVRAM-based file systems, the proposed DeEXT design introduces 4 different components, including a compressible check module, a dedupe extent structure, a pool-based space allocator, and a space reclamation scheme. On the other hand, in order to retain portability between different operating systems, this study chooses the fourth extended file system (ext4) as the base for enabling the concept of dual-chunking data deduplication. In addition, to take advantage of the byte-addressability feature of NVRAM within ext4, corresponding adjustments are made to the ext4 to allocate/deallocate storage space and read/write files at the byte-level granularity.

In the proposed DeEXT design, a compressible check module is firstly included for predicting the deduplication potentiality of a data stream by its compressibility. Data that are compressible tend to be uncoded data and usually have higher deduplication potentiality than incompressible data. According to the compression sampling result, CDC and FSC are applied to divide compressible and incompressible data into smaller data chunks. With this dual-chunking approach, the proposed design can achieve higher redundancy removal ratio for data with higher data deduplication potentiality, while lowering the chunking overhead for data with lower data deduplication potentiality. Furthermore, the compressibility information can also be used as a hotness indication for illustrating the update frequency of each data chunk. After deduplicating data with two different chunking methods, the proposed design manages those deduplicated data chunks via the dedupe extent structure based on the original metadata structure of ext4. Then, a pool-based space allocator is included to divide the storage space into small resource pools and accommodate data chunks within different pools based on the predicted data update frequencies of the compressible check module. In addition, the proposed design only allow sequential-write operations within each pool to ensure NVRAM cells of the same pool will have similar wearing. Finally, due to the added sequential-write constraint within each pool, the space reclamation scheme reclaims those invalid pools based on the amount of invalid data in a pool. With the designed components, the proposed DeEXT design can reduce the storage space usage by 41.84% on average and improve the read/write performance by an average of 37.76% when compared with the conventional ext4 design.

The rest of this paper is organized as follows. Section II reports the background and the motivation of this paper. Section III presents the mechanism of the proposed DeEXT design. Then, in Section IV, the proposed design was evaluated with real-world traces. Finally, Section V concludes this paper and the research remarks.

II. Background and Motivation

NVRAM has emerged as a promising storage medium for both the industry and consumer electrics owing to its attractive features. When compared with NAND flash, NVRAM has the advantages of byte-addressability, shorter access latency, and higher endurance. On the other hand, when compared with DRAM, NVRAM also has the advantage of higher density. Such characteristics have increased the incentive of coexisting NVRAM and DRAM on the memory bus so as to exploit the benefits of byte-addressable NVRAM. The architecture of coexisting NVRAM and DRAM on the memory bus has inspired numerous excellent designs. For example, a log-structured file system, called NOVA [23], has been proposed to exploit the fast random access feature of NVRAM and to enhance the degree of access concurrency. In additional, Chen et al. [5] also proposed the idea of union page cache to boost the file access performance for the conventional block-based file system on NVRAM storage devices. From another perspective, researchers also tried to boost the data deduplication ratio or performance of data deduplication algorithm on NVRAM. For instance, Wang et al. [20] proposed an inline deduplication algorithm, called NV-Dedup, to improve data deduplication performance on NVRAM via managing the deduplication metadata with finer granularity and consistency consideration.

To enforce inline deduplication, chunking is the first and the most critical step. To chunk a data stream into small data chunks for deduplication, FSC is a fast and easy way but may face the issue of low data deduplication ratio owing to the boundary-shift problem. For instance, if a few bytes are inserted or removed from a data stream, the boundaries identified by FSC will be shifted, thus lowering the number of deduplicated data chunks. To resolve this boundary-shift problem, CDC is proposed to utilize a sliding window for calculating the fingerprint. A chunking boundary will then be declared if the calculated fingerprint meets certain conditions. An example is give in Figure 1 to illustrate the difference between FSC and CDC. As shown in Figure 1(a), to chunk an updated file, FSC cannot effectively identify the original

[2]Fingerprint indexing searches the existing pool of fingerprint for identifying identical fingerprints.

[3]In this study, the term "deduplication potentiality" refers to the potential amount of duplicate data that can be deduplicated from a data stream.

deduplicated data chunk and will generate new deduplicated data chunks, thus lowering the deduplication ratio. On the other hand, in Figure 1(b), the CDC method can still identify the original boundary and those deduplicated data chunks, such as C_1, C_3 and C_4, the content of which are not changed. As a result, CDC can achieve higher deduplication ratio than FSC. However, since CDC actually checks the data contents of each data stream or file, the chunking latency of CDC is longer than that of FSC. Besides the chunking latency, the fingerprint calculation latency and fingerprint indexing latency could also affect the performance of an inline deduplication storage system. To resolve the latency issue of inline deduplication, Xia et al. [22] proposed the FastCDC approach to simplify the procedure and enhance the performance of the latest Gear-based CDC. In addition, parallelizing techniques [7] have been identified to speed the fingerprint calculation process. Furthermore, other studies [1] explored the methods for speeding up the indexing performance in large-scale deduplication storage systems. *However, previous works have not considered the possibility of utilizing the characteristics of duplicate data content to shorten the latency and lower the management overhead of inline deduplication.*

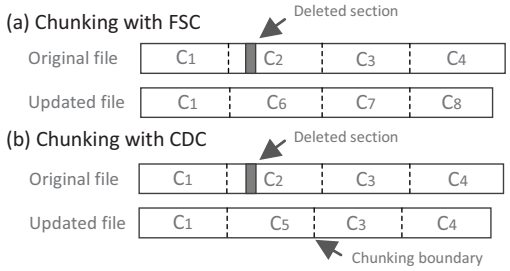

Fig. 1. Comparison of the FSC and CDC methods.

To resolve above issues, this study proposes the concept of dual-chunking data deduplication to draw the benefits from both the FCS and CDC approaches. The key observation of dual-chunking data deduplication concept is that applying CDC for chunking data with lower deduplication potentiality could cause unnecessary latency and fingerprint management overhead for an inline deduplication mechanism. Therefore, the proposed concept firstly utilizes FCS to lower the chunking latency and fingerprint indexing overhead for those data that have lower deduplication potentiality. Data with low deduplication potentiality are usually coded data, including video, image, and compressed data. On the other hand, CDC is used to enhance the deduplication ratio for those data with higher deduplication potentiality, including uncoded raw data and text files. In the end, the technical difficulty lies on *how to realize the concept of dual-chunking data deduplication for considering the characteristics of deduplicated data content while utilizing the feature and improving the profitability of NVRAM-based storage devices.* To realize the concept of dual-chunking data deduplication and improve the profitability of NVRAM-based storages, we propose the design of data deduplication extended file system (DeEXT) to predict the deduplication potentiality of each data stream via compression sampling and utilize the metadata structure of extended file system for fingerprint indexing. The details of the proposed DeEXT design are described in Section III.

III. Data Deduplication Extended File System

A. Overview

In this section, we present the data deduplication extended file system (DeEXT) for improving the profitability of NVRAM-based storage by utilizing the concept of dual-chunking data deduplication to remove data redundancy and lower the chunking overhead. To the best of our knowledge, this work is a pioneer work that aims to consider the characteristics of duplicate data content while integrating data deduplication mechanisms into NVRAM-based storage systems. The goal of the DeEXT is to adaptively utilize different chunking methods, such as FSC and CDC, for data with different inherent deduplication potentiality. On the other hand, to retain portability between different systems, the proposed DeEXT is designed and implemented based on the conventional ext4 file system. The system architecture of the proposed DeEXT can be illustrated as Figure 2.

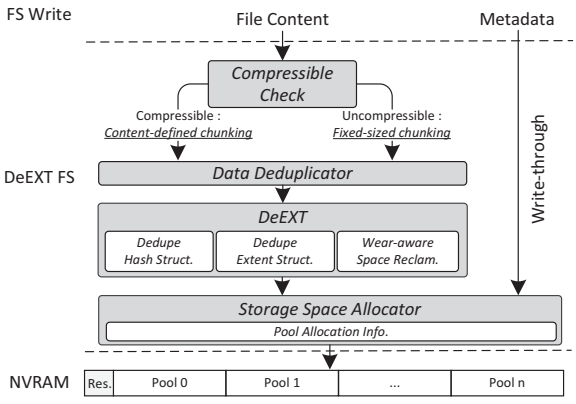

Fig. 2. The system architecture of DeEXT.

As shown in Figure 2, the proposed DeEXT design utilizes the compressible check module [12] for inferring the deduplication potentiality of each data stream via the results of compression sampling. Based on the predicted deduplication potentiality of each data stream, data streams are sent to the data deduplicator for chunking and deduplicating. For instance, data with high deduplication potentiality will be chunked with the CDC method for achieving higher deduplication ratio, while data that are less likely to be deduplicated will be chunked with the FSC method to lower the chunking and fingerprint indexing overhead. Note that the DeEXT design performs compression ratio sampling without actually compressing the whole data content, thus inducing limited decompression overhead. Then, based on the extent structure of ext4, the proposed DeEXT firstly lowers the fingerprint indexing overhead of those deduplicated data chunks via a dedupe extent structure (see Section III.III-B), which is designed based on the original extended structure of ext4. Then, a pool-based space allocator (see Section III.III-C) is included to store deduplicated data chunks within different resource pools with the consideration of data update frequencies; meanwhile, a sequential-write constraint is included to ensure that NVRAM cells of the same resource pool will have similar wearing. Finally, due to the sequential-write constraint used for ensuring similar wearing, a space reclamation scheme (see Section III.III-D) is designed to reclaim invalid space based on the amount of invalid data in each resource pool.

With the aforementioned designed components, the proposed DeEXT design can effectively integrate data deduplication into NVRAM-based storage systems while considering duplicate data content and the wearing of NVRAM.

B. Dedupe Extent Structure

To manage the data chunks of two different chunking methods, the proposed DeEXT design merges the management of deduplicated data chunks and fingerprints into the design of extended file system. Therefore, the DeEXT design manages deduplicated data chunks by following the management approach of conventional file data in extended file systems. The difference is that the original extent structure is now replaced with the dedupe extent structure, including dedupe header and dedupe index. The details of the dedupe extent structure are summarized in Figure 3.

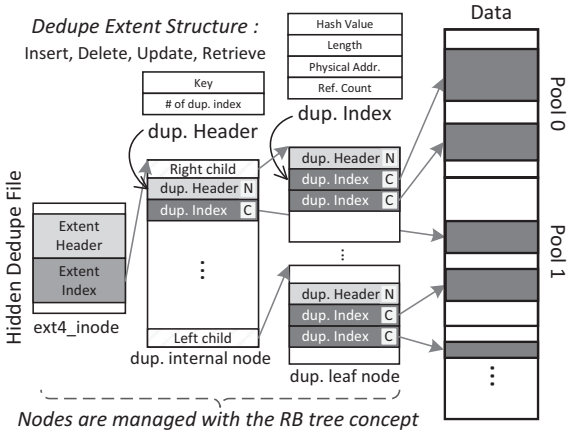

Fig. 3. The dedupe extent structure of DeEXT.

As shown in Figure 3, by following the file management approach of conventional ext4, DeEXT manages the deduplicated data chunks of each chunking method as an *ext4_inode* instance, also known as the hidden dedupe file in this study. Then, within the hidden dedupe file, multiple *dedupe indexes* are utilized to track deduplicated data chunks, their fingerprint, and reference count. The instance of *dedupe indexes* are then grouped into *dedupe internal nodes* and *dedupe leaf nodes*, which are managed by Red Black (RB) Tree structure to enhance the performance of fingerprint indexing. In other words, the proposed DeEXT design can lower the overhead of deduplication metadata by managing the deduplicated data chunks and their fingerprints with a single dedupe extent structure. Note that the *hidden dedupe file* are hidden in the metadata of DeEXT design and invisible to front-end users.

To link user files with those deduplicated data chunks in the hidden dedupe files, modifications are also made to the original *ext4_inode* structure of user files, which is illustrated in Figure 4. As illustrated in the figure, the *extent indexes* of conventional *ext4_inode* are altered for pointing to the *dedupe indexes* of hidden dedupe files, instead of the physical location of the data chunks in the resource pools. With this approach, the proposed DeEXT can relocate deduplicated data chunks without updating all the *extent indexes* of visible system files for pointing to a new physical location of the deduplicated data chunk. On the other hand, in the proposed DeEXT, instead of allowing random-write operation, DeEXT only allows sequential-write operations to avoid excessive wearing issue on NVRAM-based storage devices (see Section III.III-C).

C. Pool-based Space Allocator

To consider the wearing issue of NVRAM within the design of DeEXT, a pool-based space allocator is included to divide the storage space into multiple resource pools for ensuring NVRAM cells of a single pool will have similar wearing. To enforce the resource pools design, the DeEXT introduces the virtual block group to store all the file-system-related metadata. The physical location of this virtual block group can be adjusted to avoid the issue of uneven wear. Following the design of resource pools, the pool-based space allocator also considers the data update-frequency when allocating resource pools for incoming write requests. The design of resource pools is summarized in Figure 5.

As shown in Figure 5(a), the virtual block group is initialized to the beginning location of the storage space and the rest storage space is divided into multiple resource pools. To facilitate the management of resource pools, pool lists are included in the pool-based space allocator. As illustrated in Figure 5(b), a free pool list is utilized to track those free pools, which are sorted according to the accumulated write count of each pool. In the free list, to consider the wearing condition of each pool, the free list is virtually divided into 4 regions, including youngest, young, old, and oldest, for serving write request with different update frequency. Then, the resource pools of youngest and young regions will be allocated for data with high update frequency. On the other hand, the resource pools of old and oldest region will be allocated for serving data with lower update frequency for prolonging the lifetime of pools with larger write count.

In order to differentiate data chunk of different update frequency, 4 additional lists, which are shown in Figure 6, are included in the included pool-based space allocator. The four included lists are new CDC, updated CDC, new FSC and updated FSC. These four lists are used to accommodate newly-written CDC data chunk, updated CDC data chunk, newly-written FSC data chunk, and updated FSC data chunk. In addition, by storing CDC and FSC data chunks in different resource pools, the proposed DeEXT can achieve higher space utilization by eliminating the alignment problem between CDC and FSC data chunks since the length of CDC data chunk may vary while the FSC data chunk has fixed data length. Furthermore, storing data of similar update frequency

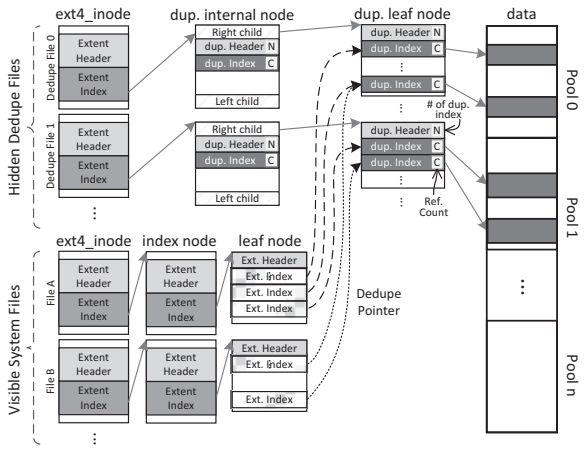

Fig. 4. The overall extent structure of DeEXT.

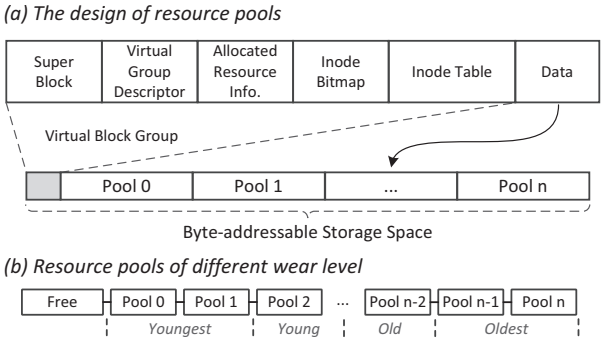

Fig. 5. The design of resource pools.

in the same pool can also benefit the procedure of invalid space reclamation since data of similar update frequency tend to be invalidated at the similar time point and can lower the overhead of live-data copies during invalid space reclamation. To reclaim those invalid space in each pool, the proposed DeEXT includes a space reclamation scheme to relocate valid data and reclaim invalid space (see Section III.III-D).

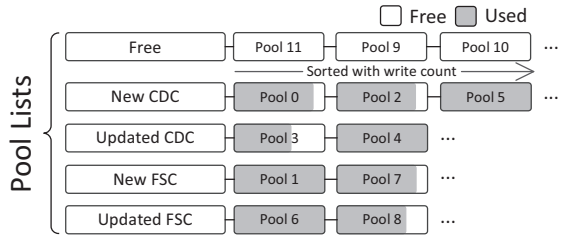

Fig. 6. Resource pool lists.

D. Space Reclamation Scheme

Due to the added sequential-write constraint within each pool, the proposed DeEXT design includes a space reclamation scheme to reclaim invalid resource pools based on the amount of invalid data in a resource pool. When DeEXT design runs out of free pools, the space reclamation scheme is initialized to reclaim space from all resource pool lists. Resource pools are selected as victim resource pools based on their invalid space ratio, which can be calculated as $Size\ of\ valid\ data/Size\ of\ a\ pool$. The procedure of reclaiming resource pools is summarized in Figure 7.

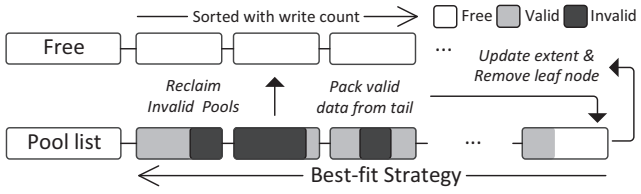

Fig. 7. Reclamation of resource pools.

As shown in Figure 7, when a resource pool is selected as a victim resource pool, the valid data of the victim resource pool will be packed to the tail pool of the same list. During packing those valid data, the corresponding extent entries maintained in the dedupe extent structure need to be updated accordingly. Therefore, the included space reclamation scheme also constructs a temporal data chunk information structure of each pool list based on the maintained dedupe extent structure. When a resource pool is selected as the victim resource pool, the extent entries of those valid data chunks can be updated when the valid data chunks are packed into other resource pool.

IV. Performance Evaluation

A. Experiment Setup

In this section, experiments are conducted to evaluate the capability of the proposed DeEXT regarding the storage space usage and the access performance of different percentage of compressible file data. In this paper, the proposed DeEXT is implemented based on the structure of ext4 and evaluated on an extended file system simulator [6]. The experimental results of DeEXT are then compared with the results of conventional ext4. To evaluate the performance of DeEXT and its compared designs, real-world deduplication traces collected by Microsoft Research are used in this evaluation. The deduplication traces are collected via scanning around 850 file systems and chunking data with FSC of 16 KB chunk size and CDC of 16 KB statistical mean chunk. Then, a salted MD5 is utilized as the hash function for generating the fingerprint for each data chunk. As for the parameters setting, we refer to PCM [4] as a case study because PCM is one of the commercialized NVRAMs. The latency of the included compressible check module is also considered by referring to the Incompressible Data Predictor (IDP) [16] module proposed by Park et al. The experimental I/O latency settings of the studied PCM and compressible check module are summarized in Table II. Note that this experiment setup assumes the percentage of compressible file data is between 0% and 100%.

TABLE II
Experimental parameters [4, 16].

Item	Specification	Unit
Storage size	512	GBs
Read latency	50	ns/byte
Write latency	150	ns/byte
Compressible check latency	165	ns/4 KB
Chunk size of FSC	16	KB
Statistical mean chunk of CDC	16	KB

B. Experimental Results

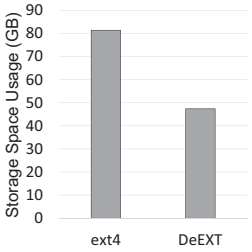

Fig. 8. Storage space usage comparison.

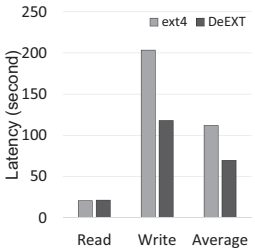

Fig. 9. Access latency comparison.

In this section, real-world deduplication traces were used to conduct the experiments to reflect the real-world usage scenario. Figure 8 compares the average storage space usage of the conventional ext4 and the proposed DeEXT deign with 50% of compressible data. As shown in the figure, the DeEXT design can effectively reduce the storage space usage

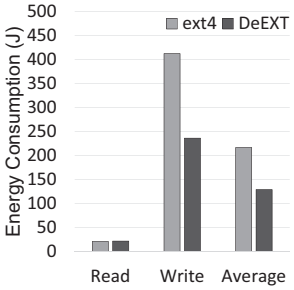

Fig. 10. Energy consumption comparison.

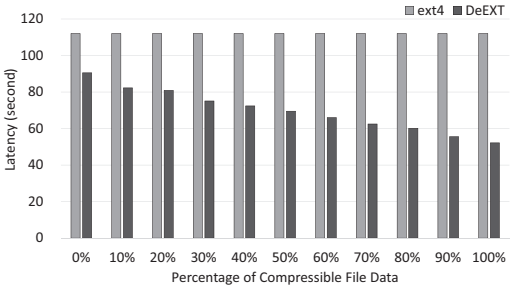

Fig. 11. Access latency comparison with different percentage of compressible file data.

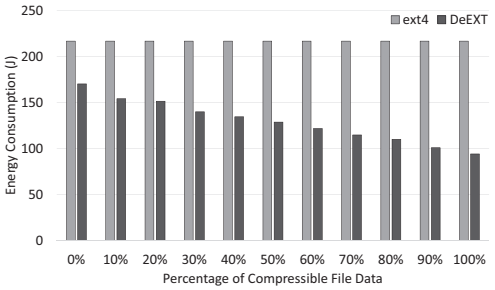

Fig. 12. Energy consumption comparison with different percentage of compressible file data.

by 41.84% when compared with the conventional ext4 design. On the other hand, Figures 9 and 10 show the comparison of average access latency and the energy consumption of the DeEXT and conventional ext4. As the results show, owing to the lower write traffic of data deduplication, the DeEXT design can enhance the access latency by 37.76% and reduce the energy consumption by 40.47%. It is worth noting that, even though DeEXT requires extra read operations for indexing through the dedupe extent structure to retrieve the file data from hidden dedupe file, DeEXT can maintain similar read performance when compared with conventional ext4 because DeEXT can effectively lower the fingerprint indexing overhead with the dual-chunking data deduplication concept.

On the other hand, the access latency and energy consumption results with different percentage of compressible data are summarized in Figures 11 and 12. As shown in both figures, even with 0% of compressible data, DeEXT can still achieve 21.49% and 19.21% of access latency and energy consumption reduction by utilizing only the FSC method.

V. Conclusion

To resolve the profitability issue of NVRAM via data deduplication, this study proposes the concept of dual-chunking data deduplication for lowering the management overhead for data with lower deduplication ratio and ensure effective redundancy removal for data with higher deduplication ratio. To realize the proposed concept, a new data deduplication extended file system, DeEXT, is proposed. The proposed DeEXT introduces a compressible check module for predicting the deduplication ratio of data streams and applying suitable chunking method for each data stream. In addition, a pool-based space allocator is used to store deduplicated data chunk with small resource pools with data hotness consideration. Finally, a space reclamation scheme is used to reclaim invalid space while balancing the wearing of each resource pool. With those introduced components of DeEXT, the proposed DeEXT can lower the storage space usage by an average of 41.84% with 50% of compressible data when compared with the conventional ext4 design. In addition, DeEXT can also improve the access performance by 37.76% on average due to the lowered amount of write traffic.

Acknowledgment

This work was supported in part by Academia Sinica under grant no. AS-CDA-107-M05 and Ministry of Science and Technology under grant nos. 107-2221-E-032-005-MY3, 107-2923-E-001-001-MY3, 108-2218-E-002-048, 108-2221-E-001-001-MY3, and 108-2221-E-001-004-MY3.

References

[1] *ChunkStash: Speeding up Inline Storage Deduplication using Flash Memory*. USENIX, June 2010.

[2] J. Arulraj, A. Pavlo, and S. R. Dulloor. Let's talk about storage and recovery methods for non-volatile memory database systems. *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 2015.

[3] M. T. Chang, P. Rosenfeld, S. L. Lu, and B. Jacob. Technology comparison for large last-level caches (l3cs): Low-leakage sram, low write-energy stt-ram, and refresh-optimized edram. In *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*, 2013.

[4] S. Chen, P. B. Gibbons, and S. Nath. Rethinking database algorithms for phase change memory. *Conference on Innovative Data Systems Research*, 2011.

[5] S.-H. Chen, T.-Y. Chen, Y.-H. Chang, H.-W. Wei, and W.-K. Shih. Enabling union page cache to boost file access performance of nvram-based storage device. In *Proceedings of the 55th Annual Design Automation Conference*, DAC '18, 2018.

[6] T. Y. Chen, Y. H. Chang, S. H. Chen, C. C. Kuo, M. C. Yang, H. W. Wei, and W. K. Shih. wrjfs: A write-reduction journaling file system for byte-addressable nvram. *IEEE Transactions on Computers*, 2018.

[7] F. Guo and P. Efstathopoulos. Building a high-performance deduplication system. In *Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference*, USENIXATC'11, 2011.

[8] Intel. Intel Optane Technology. http://www.intel.com/content/www/us/en/architecture-and-technology/non-volatile-memory.html?wapkw=3d+xpoint. Online; accessed 02 July 2018.

[9] Intel. Intel SSD 760p Series. https://ark.intel.com/products/134582/Intel-SSD-760p-Series-512GB-M_2-80mm-PCIe-3_0-x4-3D2-TLC. Online; accessed 02 July 2018.

[10] G. Jia, G. Han, J. Jiang, and L. Liu. Dynamic adaptive replacement policy in shared last-level cache of dram/pcm hybrid memory for big data storage. *IEEE Transactions on Industrial Informatics*, 2017.

[11] K. Vättö and I. Cutress and R. Smith. Analyzing Intel-Micron 3D XPoint: The Next Generation Non-Volatile Memory. https://www.anandtech.com/show/9470/intel-and-micron-announce-3d-xpoint-nonvolatile-memory-technology\\-1000x-higher-performance-endurance-than-nand. Online; accessed 02 July 2018.

[12] J. B. Khan. Method to detect uncompressible data in mass storage device, 1 2013.

[13] P. Li, D. R. Chakrabarti, C. Ding, and L. Yuan. Adaptive software caching for efficient nvram data persistence. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2017.

[14] D. T. Meyer and W. J. Bolosky. A study of practical deduplication. In *Proceedings of the 9th USENIX Conference on File and Stroage Technologies*, FAST'11, 2011.

[15] A. Muthitacharoen, B. Chen, and D. Mazières. A low-bandwidth network file system. In *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*, SOSP '01, 2001.

[16] Y. Park and J. s. Kim. zftl: power-efficient data compression support for nand flash-based consumer electronics devices. *IEEE Transactions on Consumer Electronics*, 2011.

[17] S. Quinlan and S. Dorward. Venti a new approach to archival data storage. In *Proceedings of the 1st USENIX Conference on File and Storage Technologies*, FAST '02, 2002.

[18] K. Srinivasan, T. Bisson, G. Goodson, and K. Voruganti. idedup latency-aware, inline data deduplication for primary storage. In *2012 USENIX Annual Technical Conference (USENIX ATC 12)*, 2012.

[19] 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 10th USENIX Conference on File and Storage Technologies*, FAST'12, 2012.

[20] C. Wang, Q. Wei, J. Yang, C. Chen, Y. Yang, and M. Xue. Nv-dedup high-performance inline deduplication for non-volatile memory. *IEEE Transactions on Computers*, 2018.

[21] 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 2011 USENIX Conference on USENIX Annual Technical Conference*, USENIXATC'11, pages 26–28, Berkeley, CA, USA, 2011. USENIX Association.

[22] W. Xia, Y. Zhou, H. Jiang, D. Feng, Y. Hua, Y. Hu, Q. Liu, and Y. Zhang. Fastcdc a fast and efficient content-defined chunking approach for data deduplication. In *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, 2016.

[23] J. Xu and S. Swanson. NOVA: A log-structured file system for hybrid volatile/non-volatile main memories. In *14th USENIX Conference on File and Storage Technologies (FAST 16)*, 2016.