2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA) | 978-1-6654-7652-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/HPCA56546.2023.10071011

ESD: An ECC-assisted and Selective Deduplication for Encrypted Non-Volatile Main Memory

Chunfeng Du*, Suzhen Wu*†✉, Jiapeng Wu*, Bo Mao*, Shengzhe Wang*
*School of Informatics at Xiamen University, Xiamen, Fujian, China
†Wuhan National Laboratory for Optoelectronics, Wuhan, China
✉Corresponding Author: Suzhen Wu (suzhen@xmu.edu.cn)

***Abstract*—Reducing write data to encrypted Non-Volatile Main Memory (NVMM) can directly improve NVMM's endurance, performance, and energy efficiency. However, existing works that straightforwardly apply inline deduplication on encrypted NVMM can significantly lead to system performance degradation due to high computing, memory footprint, and index-lookup overhead to generate, store, and query the cryptographic hash (fingerprint). This paper proposes ESD, an ECC-assisted and Selective Deduplication for encrypted NVMM by exploiting both the device characteristics (ECC mechanism) and the workload characteristics (content locality). First, ESD utilizes the ECC information associated with each cache line evicted from the Last-Level Cache (LLC) as the fingerprint to identify data similarity and avoids the costly hash calculating overhead on the non-duplicate cache lines. Second, ESD leverages selective deduplication to exploit the content locality within cache lines by only storing the fingerprints with high reference counts in the memory cache to reduce the memory space overhead and avoid fingerprints NVMM_lookup operations. The experimental results show that ESD can significantly speed up the writes by up to 3.4x, 4.3x, and 2.6x, speed up the reads by up to 5.3x, 5.0x, and 2.0x, and reduce the energy consumption by up to 96.3%, 96.2%, and 56.6% than Baseline, Dedup_SHA1, and DeWrite, respectively. Meanwhile, ESD also can significantly outperform other schemes in tail latency.**

I. Introduction

Memory is a fundamental performance and energy bottleneck in almost all computing systems [37]. Data explosion and application trends requiring more capacity, bandwidth, and predictability from memory make it an even more important system bottleneck. Meanwhile, the typical main memory (e.g., DRAM) suffers from limited scalability and high power leakage [39]. Emerging Non-Volatile Main Memories (NVMM), constituted by Non-Volatile Memories technologies such as Phase Change Memory (PCM), Spin-Torque Transfer RAM (STT-RAM), and Resistive RAM (ReRAM), have been deemed the most promising substitute for the new-generation Non-Volatile Main Memory (NVMM) due to the advantage of high-density and enhanced scalability [24], [35], [54], [57]. Without loss of generality, we use PCM as a representative NVMM for research and analysis.

However, there are some fundamental issues to be faced due to media characteristics when building high-performance and energy-efficient encrypted NVMM systems [9]. First, write operations to NVMM not only lead to a higher latency (i.e., 3-8x) compared to read operations [38] but also significantly incur higher energy overhead [46], [67]. Second, NVMM has limited write endurance (e.g., 10-100 million for PCM [68]) due to different manufacturing processes, which can lead to writing penetration as soon as the number of data writes exceeds the limitation [33], [71]. Third, data stored in NVMM need to be encrypted due to the issue of security because data in NVMM is not secure if the NVMM DIMM is stolen or suffers from memory bus attacks without encryption [4], [51].

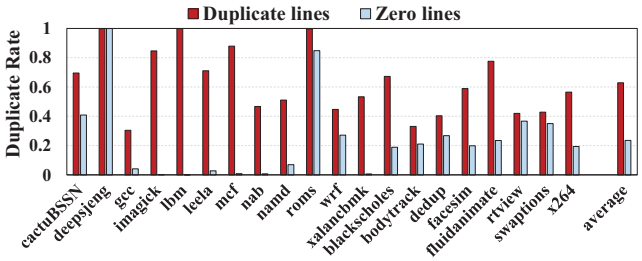


Fig. 1. Duplicate rate of cache lines *(The first 12 applications are from SPEC CPU 2017 and the rest 8 applications are from PARSEC).*

Moreover, previous studies [40], [53], [58], [73] and our analysis reveal that plenty of cache lines evicted from the LLC [30] are redundant for most real-world applications [8], [22], shown in Figure 1. It shows that the duplicate cache lines from these applications can reach 62.9% on average. This observation implies that the data deduplication technology can be utilized to effectively eliminate the duplicate cache lines [40], [73] so that the NVMM's performance, space efficiency, and endurance can be improved.

Existing studies have been proposed to combine encryption and deduplication in storage systems [6], [16], [19], [29], [63], [73], e.g., *Deduplication-after-Encryption (DaE)*, *Deduplication-before-Encryption (DbE)* and *Parallelism of Deduplication and Encryption (PDE)*. The *DaE* approach is not applicable because the same data will be changed dramatically in an encrypted NVMM system after encryption. The *PDE* approach is also not applicable because it incurs high energy overhead though the computational latency overhead of non-duplicate cache lines is hidden [73]. However, straightforwardly applying inline data deduplication before encryption is also nontrivial and can significantly degrade system performance, shown in Figure 2 in the worst case. Specifically, the reason is twofold, illustrated as follows.

High computation and energy overhead: The redundant data is detected by comparing a fingerprint (i.e., the unique identification of data). The fingerprint is usually cal-

978-1-6654-7652-2/23/$31.00 ©2023 IEEE

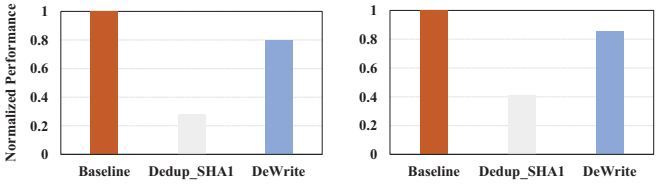


Fig. 2. The performance of different schemes normalized to baseline without deduplication in the worst case *(Left (leela) and Right (lbm))*

culated by leveraging a hash function, such as MD5 [17] or SHA1/256 [11], which introduces a high latency at hundreds of nanoseconds for a cache line. By contrast, the read and write latencies of a cache line on NVMM are usually hundreds of nanoseconds. Clearly, the deduplication-induced computational overhead is extremely high for a latency-sensitive NVMM system because existing deduplication schemes apply a hash function on all cache lines, no matter whether they are redundant or not, which adds extra hash calculating latency for all cache lines. Meanwhile, they also lead to huge energy consumption even though the latency of the hash function is hidden for the state-of-the-art DeWrite scheme [73], demonstrated in Section II-B.

High memory space and lookup overhead for fingerprints: Deduplication-based systems usually store the fingerprint information in memory and maintain a small section in the memory cache (i.e., on-chip cache in memory controller) to quickly identify redundant data. However, with the increase in device capacity, the overhead of storing fingerprint information will also become huge, which is not realistic to store such a huge fingerprint in memory, as illustrated in Section II-B. Meanwhile, this also incurs fingerprint NVMM_lookup bottleneck, similar to the fingerprint disk_lookup bottleneck in traditional disk-based deduplication system [48], because the process is on the I/O critical path.

To address these challenges, we propose ESD, an ECC-assisted and selective deduplication scheme for the encrypted NVMM system, which makes the following contributions.

- Workload analysis reveals that many cache lines are redundant from real-world applications of SPEC CPU 2017 [8] and PARSEC [22]. Most importantly, we further find the strong content locality that the 0.08% of unique cache lines are referenced by more than 1000 times but account for 42.7% total storage space before deduplication, illustrated in Section II-A
- Existing deduplication schemes for encrypted NVMM add hash computational latency and energy overhead. We propose an ECC-assisted similar cache line identification by utilizing the ECC information associated with each cache line as the fingerprint to definitively filter out non-duplicate cache lines without any hash computation and energy consumption. By exploiting the asymmetric read/write characteristics of NVMs, we further propose selective deduplication that can read these data with the same ECC values from the NVMM to perform a byte-by-byte comparison to identify and remove duplicate cache lines.
- We implemented a prototype of ESD in the gem5 [7] with NVMain [42] and evaluated its performance driven by the SPEC CPU 2017 [8] and PARSEC [22]. Experimental results show that ESD can significantly improve the system performance, compared with the Baseline scheme and deduplication-based scheme, e.g., Dedup_SHA1 and DeWrite, in terms of I/O performance, energy efficiency, and tail latency.

The rest of the paper is organized as follows. We will describe the background and motivation in Section II. In Section III, we present the design of ESD. Section IV presents our experimental results and analysis. Section V presents the related studies, and Section VI concludes the paper.

II. Background and Motivation

A. Workload Characteristics

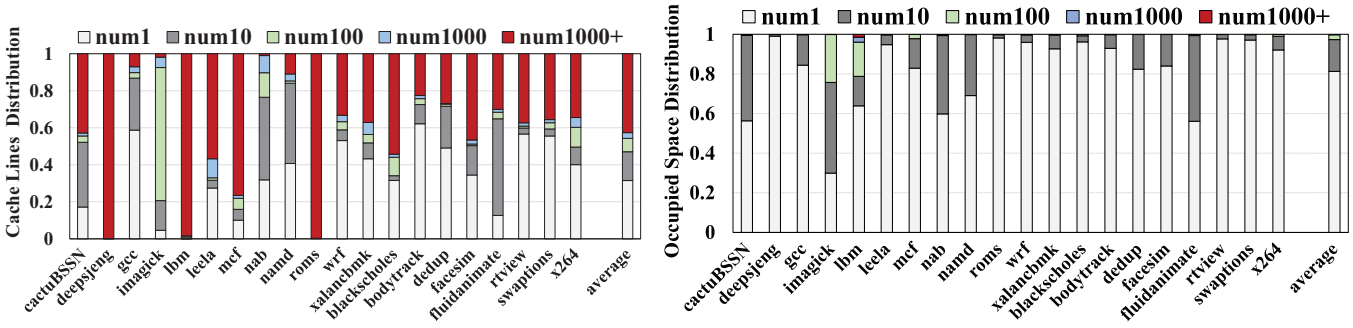
The previous studies [40], [53], [73] have revealed that moderate duplicate cache lines exist from CPU's LLC. Figure 1 shows that duplicate cache lines are existed among all the 20 applications, accounting for 33.1% to 99.9% with an average of 62.9%. We also found that for some applications, such as *deepsjeng* and *roms*, a large percentage of duplicate cache lines are contributed by the zero cache lines.

Most importantly, we found that the duplicate cache line distributions are skewed within all 20 applications. This phenomenon is called content locality in this paper and represents that a small part of unique cache lines is referenced many times in a deduplication-based NVMM system. Figure 3 shows the cache line distribution before deduplication and the occupied space distribution after deduplication based on the 20 applications, where the $*$ in the $num*$ indicates how many times the unique cache line is referenced or written to the NVMM. Meanwhile, the reference count in the deduplication system denotes how many cache lines are referenced to the unique cache line. For example, the $num1$ denotes that the cache line is unique and is only written once. The $num10$ denotes that the cache line is written from 2 to 10 times, and so for the $num100$ and $num1000$. The $num1000+$ denotes that the cache line is written by more than 1000 times. From these two figures, we can learn that high content locality exists for these applications. For example, these cache lines with reference count 1000 and above only account for 0.08% of the total unique cache lines, but they occupy a large percentage of the total storage data volumes by 42.7% on average before the deduplication is applied to the NVMM system.

In summary, the workload characteristics of memory accesses indicate that inline deduplication can be utilized for the NVMM system to reduce the write traffic. Moreover, when designing the deduplication-based NVMM system, the content locality represented by the reference count should be carefully considered and exploited to alleviate the deduplication-induced computational and memory overhead.

B. Challenges in Deduplication-based NVMM

The preliminary studies reveal that applying inline deduplication can decrease the system performance in the worst case,



(a) The cache line distribution before deduplication.

(b) The occupied space distribution after deduplication.

Fig. 3. The cache line distribution before deduplication and the occupied space distribution after deduplication.

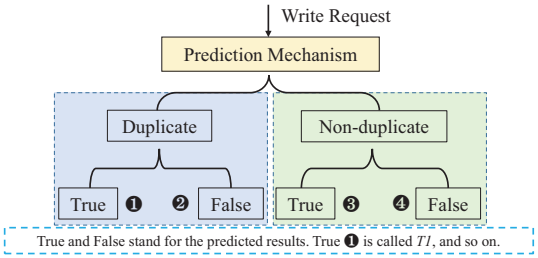


Fig. 4. The prediction mechanism in NVMM system.

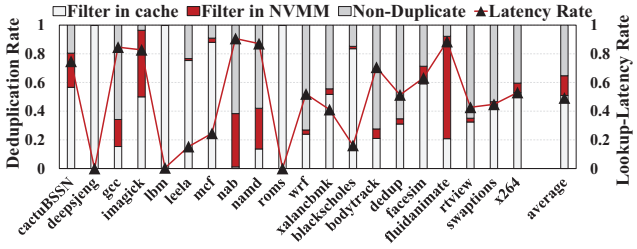


Fig. 5. The rate of duplicate cache lines filtered by the fingerprints stored in memory cache and in NVMM, and the fingerprints NVMM_lookup latency rate of writes.

though the duplicate cache lines can be eliminated significantly for saving space [40], [53], [73]. To improve the efficiency of NVMM systems with inline deduplication, there are two challenges remaining to be faced and illustrated as follows.

The first challenge is the high computational and energy overhead. Traditional inline deduplication technologies use the fingerprint (e.g., MD5 or SHA1) to identify duplicate data, which leads to significant computational latency and energy overhead [59], [73] because they generate the hash-based fingerprints for all cache lines. Though the CRC algorithm and parallel mechanism have been exploited [73], existing problems cannot be ignored. First, the computation overhead from the CRC algorithm needs for all cache lines, not only for non-duplicate cache lines. The reason is that the parallel encrypting mechanism is only for non-duplicate cache lines in the DeWrite scheme. Second, even with strictly guaranteed correct predictions, the energy overhead for all cache lines due to CRC computation cannot be overlapped, not to mention significantly adding hash computational overhead and wasting the energy resource in the case of incorrect predictions.

Figure 4 shows the prediction mechanism for the deduplication-based NVMM system. *T1*, *F2*, *T3* and *F4* stand for the predicted results. When the result of the prediction shows that the cache line is duplicated (e.g., *T1* and *F2*), all the cache lines need to be computed for fingerprint by a hash algorithm. The increased time is nontrivial, especially for applications with moderate deduplication rates and high write ratios. For example, the maximum time increases by up to 450.7ms for the *lbm* application in DeWrite, even though the number of cache lines is only *million-level*. Meanwhile, only when the result of the prediction is non-duplicated the parallel encryption mechanism will be triggered (e.g., *T3* and *F4*). The benefits of the parallel encryption mechanism are obvious because it can hide the latency of CRC computation and read for comparison. However, if the result of the prediction is wrong (i.e., F2 and F4), it will inevitably lead to bad situations. First, for the F2, the total latency includes CRC computation, fingerprint lookup, reads for comparison, and encryption. Second, for the F4, it will lead to resource waste, e.g., extra cryptographic operations, which incur additional latency and energy consumption per cache line, even though the latency of deduplication logic will be hidden. Thus, how to quickly and efficiently perform the deduplication is critically important for encrypted NVMM.

The second challenge is the memory footprint and lookup overhead in NVMM for the deduplication-induced metadata. For deduplication systems, the fingerprint index is important metadata that occupies a large memory capacity. For example, the memory footprint required to store the metadata of the DeWrite [73] is 6.25% of the total unique storage space for a cache line. But it increases to 25% when the cache line size is 64 bytes which is widely used in previous studies [14], [66]. Meanwhile, storing the whole fingerprints in NVMM and only maintaining a small number of fingerprints in the memory cache will introduce the fingerprints NVMM_lookup bottleneck, which also affects the system performance, besides the non-negligible NVMM space overhead. Therefore, how to effectively place metadata and improve the efficiency of data access is a question worth thinking about.

Figure 5 shows the rate of duplicate cache lines filtered by the fingerprints stored in the memory cache and NVMM, respectively, along with the fingerprint NVMM_lookup induced performance overhead for the NVMM system. We can see that an average of 51.0% duplicate cache lines can be filtered by the fingerprints stored in the memory cache, and an average

of 13.7% duplicate cache lines are filtered by the fingerprints stored in NVMM. Moreover, to filter these 13.7% duplicate cache lines, the extra fingerprint NVMM_lookup processes degrade the performance by up to 90.7% with an average of 49.2%. However, spending the most system resources (NVMM space and performance overhead) to gain the extra 13.7% benefits (duplicate cache lines) is not an efficient design choice in computer architecture. As a result, though inline deduplication is a useful and effective way to improve performance for the encrypted NVMM, some challenges remain to effectively incorporate inline deduplication technology.

C. Motivation

Although the hash computational latency of fingerprints in parallel mechanisms can be hidden for data that are predicted to be non-redundant, it has been the focus of attention [19], [29] on how to perfectly combine encryption and deduplication to achieve secure deduplication enabled NVMM system. However, the *DaE* is not applicable due to the strong diffusion effect caused by encryption that makes the data change significantly before and after [6], [16]. The *PDE* requires computing fingerprint values of all redundant data, while the energy consumption from all hash computational fingerprints cannot be ignored, not to mention the dramatic increase in latency and energy consumption in the case of failed predictions [73]. Inspired by these observations, we believe that if fingerprints are generated without extra computation overhead, then deduplication and encryption procedures can be effectively integrated.

Error Correction Code (ECC) provides error detection and correction abilities in case of memory bit errors [26], [41], [44], which is widely used in computer systems. Meanwhile, the ECC values can be used in a fine-grained cache line, i.e., Per-Word, e.g., the 8-Byte word is matched with an 8-bit ECC. Each 8-Byte word generates an 8-bit ECC, and a 64-Byte cache line generates a 64-bit ECC [64]. Many processors with large capacity caches have ECC-protected L2 and L3 (also known as the last-level cache) caches [1], [12], [65]. For example, Qualcomm Centriq 2400 Processor has 512KB L2 cache with ECC and a distributed 60MB L3 Cache (12 x 5MB) with ECC [43], AMD EPYC processors have internal ECC-enabled L2 and L3 caches [3], and Intel Xeon Processors have internal ECC logic to detect errors for L2 and L3 caches [20], [25]. Most importantly, existing works demonstrate that the memory controller computes the ECC for each cache line and sends the ECC on specific bits along with the data [36], [47], [64], when cache lines are evicted to NVMM from LLC.

In general, the ECC information from the memory controller within a processor chip is considered absolutely secure and cannot be observed or manipulated by an attacker by any means [62]. Besides the error detecting and correcting abilities, ECC can be used for data similarity identification. For example, if two ECC values are different, their corresponding cache lines are different. Otherwise, their cache lines are the same with a high possibility, though not 100%. The corresponding cache line is further fetched to the memory for a byte-by-byte comparison [59]. Inspired by the fact that ECC can effectively identify data similarity, we can exploit it to filter non-duplicate cache lines in deduplication-based NVMM quickly.

In summary, both the media characteristics (ECC mechanism) and the workload characteristics (content locality) should be carefully considered and exploited. Specifically, for the hash computational overhead, ESD uses the existing ECC values associated with cache lines rather than hash algorithms (i.e., MD5 and SHA1) to calculate the fingerprints for filtering the non-duplicate cache lines. To efficiently decrease the storage overhead for storing fingerprints, ESD only stores those fingerprints with high reference counts in the memory cache and conducts selective deduplication.

III. The Design of ESD

A. Architecture Overview of ESD

ESD works with other modules and locates inside the memory controller on the CPU-side, which is considered absolutely secure [62]. The design objective of ESD is to completely eliminate the computational overhead, reduce metadata occupancy, and eliminate NVMM_lookup overhead for fingerprints, by applying ECC-based similarity identification and selective deduplication for encrypted NVMM. Specifically, ESD first intercepts existing ECC values associated with cache lines as the fingerprints to identify data similarity and filter the non-duplicate cache lines on the critical write path. Then for these cache lines which have a high probability to be redundant, ESD presents a selective deduplication scheme to merely remove these cache lines with high reference counts on the write path by using a reference-count-based fingerprint cache. Although not all duplicate cache lines can be eliminated, ESD can eliminate a majority of the duplicate cache lines with slight memory space without the computational overhead and fingerprints NVMM_lookup overhead. To defend against a series of data attacks, e.g., hardware stolen and data leakage, the written data must be encrypted when evicted from the CPU-side to the memory bus. Inspired by Section II-C, ESD leverages the counter model encryption (CME) algorithm for encrypting cache lines because the majority of redundant data have been eliminated.

Figure 6 shows the architecture overview of ESD which mainly consists of three functional modules, e.g., the ECC-based Similarity Identification, the Selective Deduplication, and the Deduplication Metadata. Different from the traditional deduplication schemes, ESD does not require costly hash functions to generate fingerprints but piggybacks on the ECC information associated with cache lines as the similarity identification of a cache line. The specific functions of each module are described as follows.

When a cache line is evicted from the LLC, ESD first obtains the ECC values associated with the corresponding cache lines and passes them to the ECC-based Similarity Identification module. The ECC-based Similarity Identification module is responsible for detecting non-duplicate cache lines by checking their ECC values within the ECC-based fingerprint cache. However, for these cache lines which are

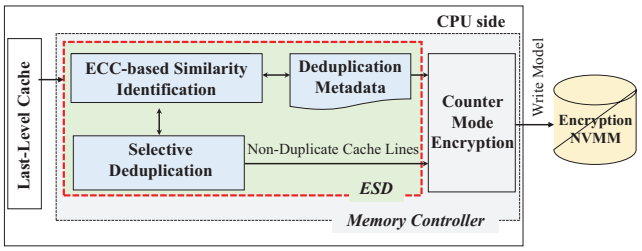


Fig. 6. The architecture overview of ESD.

not filtered out, collisions can occur since different cache lines may generate the same ECC. Though the probability is low, we cannot remove these cache lines based on their ECC values and need to further compare their content. Therefore, when the ECC value hits the ECC-based fingerprint cache, the ECC-based Similarity Identification module passes it to the Selective Deduplication module.

The Selective Deduplication module is primarily responsible for conducting selective deduplication by maintaining the fingerprint items with high reference counts and removing identical cache lines when both the ECC-based fingerprint and the cache line content are identical. The cache line is considered a non-duplicate cache line and be written to the NVMM as long as its fingerprint or content is different from the existing one.

The Deduplication Metadata module is mainly responsible for managing the metadata used in ESD and storing them in the memory cache and NVMM, which undoubtedly increases the memory space overhead. The Deduplication Metadata includes two main data structures, the ECC-based Fingerprint Index Table (EFIT) and the Address Mapping Table (AMT), which are described in the following Section III-B. The EFIT is stored in the memory controller cache for filtering the non-duplicate cache lines and identifying the data similarity. Meanwhile, the AMT is stored fully in NVMM and the hot items are buffered in the memory controller cache.

B. Data Structures

ESD mainly maintains two data structures to support the ECC-based data similarity identification and selective deduplication, EFIT and AMT, demonstrated in Figure 7.

The EFIT records ECC-based fingerprint items for deduplication processes. Each entry in EFIT includes four variables, shown as $\langle ECC, Addr\_base, Addr\_offsets, referH \rangle$. The *ECC* is obtained from the memory controller when the corresponding cache line is evicted from LLC. The *Addr_base* occupies 4 bytes which present the base addressing space. However, with the release of the Intel Optane PM device [21], the total capacity of NVMM could reach 1.5 TB. The 4 bytes addressing space is not sufficient in practice because the total addressing space is only 256 GB based on 64 bytes cache line size, which is a constant cache line size granularity generated by the CPU core and widely used in the previous studies [4], [10], [14], [66]. Therefore, ESD adopts a simple tradeoff that adds *Addr_offsets* as the offset of addressing space, which means that the real physical address includes

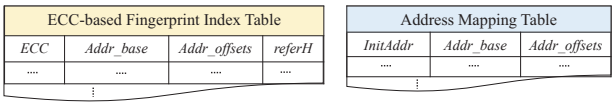


Fig. 7. Two data structures of ESD.

two sections, e.g., *Addr_base* (4-byte) and *Addr_offsets* (1-byte). To get the physical address, ESD first shifts left 8-bit of *Addr_base* and then plus *Addr_offsets*. The 40-bit physical address could address up to 64 TB of total memory space, which is enough for current servers. The *referH* indicates the number of remapping to the same physical address. It is set to be 1 byte which is enough because more than 99.9% of the reference counts are less than 1000, as shown in Figure 3(b). When the reference counts increase more than the limited number, ESD will consider the cache line as a new cache line and write it to NVMM.

The AMT records the mapping relationship between the logical address to its real physical address in deduplication-based NVMM, shown as $\langle initAddr, Addr\_base, Addr\_offsets \rangle$. It is a many-to-one relationship and updated on the write path. The *initAddr* denotes the logical address of a cache line that is allocated from the user when it is evicted from the CPU. The real physical address also consists of two sections (i.e., the *Addr_base*, *Addr_offsets*) and has the same meaning as that in the EFIT. ESD maintains the AMT in NVMM and buffers the hot items (e.g., the items with high reference counts) in the memory cache to provide fast query performance.

Identifying the similarity of cache lines and filtering out non-duplicate cache lines with the help of fingerprints is a crucial step in a deduplication-based NVMM system. With the similarity identification capability of the ECC value in the EFIT, ESD can effectively filter the non-duplicate cache lines and identify the similarity of cache lines without generating dramatic latency overhead, illustrated in Section III-C. For similar cache lines, ESD performs a further byte-by-byte comparison to determine the redundancy of cache lines, illustrated in Section III-D.

C. ECC-based Similarity Identification

For a typical deduplication system, the fingerprint of data is used as a unique identifier to detect and filter duplicate data to eliminate redundancy and save efficient storage space. However, although the typical hash functions for calculating fingerprint, e.g., MD5 [17] or SHA1 [11] could be considered a unique identification for data, their calculating latency is extremely high, e.g., 312ns for MD5 [73] and 321ns for SHA1, which can cause cascade blocking and significantly lengthen the write delays in deduplication-based storage systems, especially for the NVMM with ultra-low latency. DeWrite [73] scheme utilizes the lightweight CRC to detect similar cache lines and leverage a parallel method for eliminating the computational overhead. However, DeWrite needs to strictly depend on the result of prediction, which can also incur a heavy overhead [27]. The reason consists of two sections. First, if the result of the prediction is true, DeWrite needs to calculate the CRC for all duplicate cache lines though

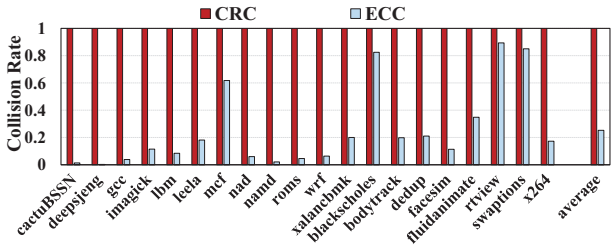


Fig. 8. The comparison of different collision probabilities, normalized to the CRC-based method.

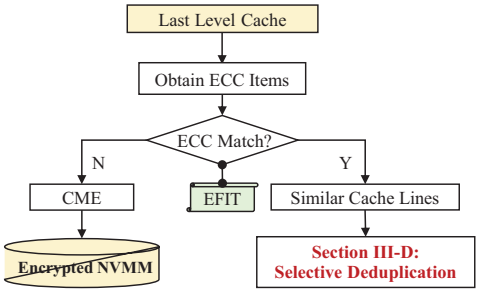


Fig. 9. ECC-based similarity identification workflow.

these redundancy cache lines will be removed. Second, if the result of the prediction is false, DeWrite will lead to two extreme cases, e.g., CRC calculation and encryption becomes a serial operation or many duplicate cache lines for invalid encryption. In addition, CRC also needs extra hardware logic support, besides the high hash collision rate demonstrated in Figure 8. By contrast, ESD uses the already existing ECC value associated with each cache line as the fingerprint to definitively filter out the non-duplicate cache lines to prevent costly hash computing and identify the similarity of cache lines. Hence, the latency of the hash calculation can be completely eliminated on the critical write path in ESD.

Figure 9 shows the workflow of the ECC-based similarity identification. When cache lines are evicted from the LLC of the CPU, ESD can obtain the ECC items from the memory controller. Due to the association with the cache line, the overhead of obtaining ECC is negligible and does not compromise the original error-checking function of ECC. Then, ESD will search the EFIT to check whether the ECC items exist within EFIT or not. If the ECC items are matched in the EFIT, these cache lines with these ECC items will be considered similar cache lines due to the collision probability, i.e., there could be the same data with the same ECC items in NVMM, and a further byte-by-byte comparison is needed based on the data in NVMM to determine whether the cache line is duplicate. Hence, the result of ECC-based similarity identification of cache lines will pass to the Selective Deduplication module. In this process, the latency of the deduplication logic is negligible. First. ESD can save the latency overhead of computing fingerprints. Second, the process of metadata lookup only happens in the cache, since ESD executes selective deduplication, illustrated in III-D. Meanwhile, ESD will add the address mapping item to AMT and cache relative mapping items in the cache based on user requests.

To accelerate the read/write processes, the ECC-based fingerprints should be stored in the memory cache as much as possible. However, it's not realistic to store all the fingerprint items in the memory cache, and the whole fingerprints should be stored in the NVMM in the deduplication-based NVMM systems, such as the traditional Baseline scheme and the DeWrite [73] scheme. Different from the full deduplication method used in the traditional Baseline scheme and the DeWrite scheme, ESD only stores the ECC-based fingerprints with high reference counts in the memory cache and performs a selective deduplication method for the NVMM system, illustrated in Section III-D. Thus, not only can the latency of accessing the fingerprints stored in NVMM on the critical write path be eliminated, but also the memory space overhead can be decreased by ESD.

D. Selective Deduplication

By eliminating duplicate cache lines, inline deduplication not only improves NVMM space efficiency but also reduces the write traffic. However, the fingerprint NVMM_Lookup overhead during the deduplication could cause significant performance degradation by up to 50% or more, as illustrated in Figure 5. The reason is that the traditional Dedup-SHA1 scheme and the DeWrite scheme perform full deduplication, even though the reference count of cache lines is only 1. Figure 10 shows a comparison of full deduplication (Dedup-SHA1 and DeWrite) and selective deduplication (ESD). Both the Dedup-SHA1 scheme and DeWrite scheme store the whole fingerprints in the NVMM. Once the fingerprint is not in the fingerprint cache, read accesses of the fingerprints stored in the NVMM are performed to determine whether the cache line is similar or not. These extra fingerprint NVMM_lookup operations only contribute a little to the overall redundancy elimination by less than 13.7%. However, they induce a huge performance overhead on the write path by up to 90.7% with an average of 50.0%, as shown in Figure 5.

Based on this observation, in order to avoid the fingerprints NVMM_lookup overhead, ESD employs selective deduplication by only checking the ECC items within the ECC-based fingerprint cache on the CPU-side which is considered absolutely secure [62], If the ECC item does not exist, ESD considers the cache line as a non-duplicate cache line and writes it to the NVMM system shown in Figure 9. Meanwhile, ESD will write the fingerprint item to EFIT in the memory cache and add the address mapping to AMT if the memory cache is not full. Otherwise, ESD will determine if a replacement needs to occur and add the corresponding entry.

Though ESD may miss detecting some duplicate cache lines, the fingerprint NVMM_lookup operations are avoided. In order to improve the hit ratio of the ECC-based fingerprint cache, ESD leverages a Least Reference Count Used (LRCU) cache strategy that mainly exploits the content locality represented by the reference count. When cache replacement occurs, the LRCU policy prioritizes replacing ECC-based fingerprints with a reference count of 1 so that ECC-based fingerprints with a reference count of 2 and larger can be kept in the memory controller cache to detect many more similar

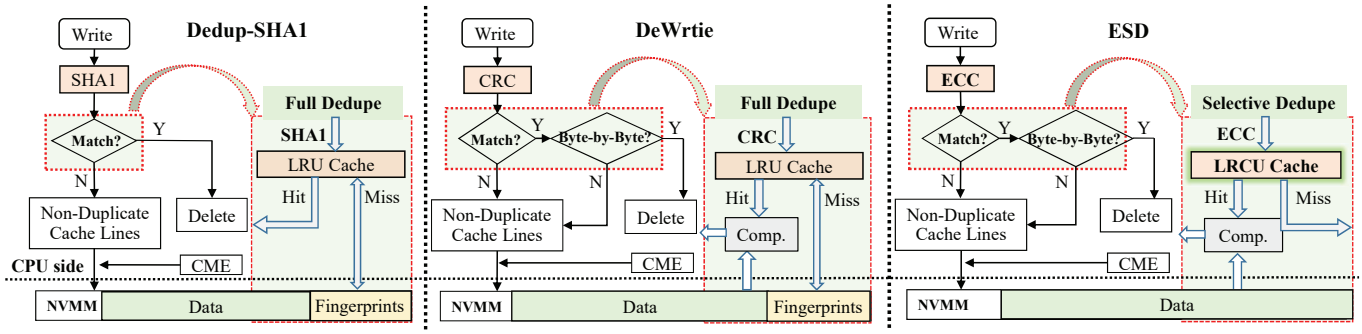


Fig. 10. A comparison of full deduplication (Dedup_SHA1 and DeWrite) and selective deduplication (ESD).

cache lines. Moreover, ESD performs a regular refresh of all cache items in the cache, i.e., subtracting a fixed value to ensure that the cache items are updated in real-time.

For selective deduplication, one design choice should be clarified and designed. If a cache line is referenced many times and exceeds the limits of *referH*, this cache line should not be updated when the same cache line arrives. In this case, ESD needs to rewrite the new cache line and update the AMT. However, the probability of this case is extremely low because less than 0.08% cache lines have a reference count of more than 1000, illustrated in Figure 3(b). Hence, ESD sets 1 Byte for *referH*, which is enough for recording the reference count.

E. Security and Consistency Issues

Security and consistency issues for ESD include three aspects: (1) Plaintext-generated ECC cannot cause a serious security vulnerability, (2) There is no data loss possibility caused by deduplication-induced write eliminations, and (3) The key data structures should be safely stored persistently.

First, Similar to the previous studies [62], [70], [71], the trust base of ESD is only the processor chip. Any data stored outside the processor chip is considered vulnerable to external attacks. ECC items are associated with plaintext cache lines before flushing them to the NVMM in ESD. Storing these ECC items can reveal the original cache line and thus cause a serious security vulnerability. However, since these ECC items are only stored in the memory controller cache on the CPU-side, they are absolutely secure and cannot be manipulated by attackers by any means [62], [73].

Second, traditional MD5/SHA-based deduplication schemes may eliminate non-duplicate data blocks that are identified as duplicate data blocks due to the hash collision possibility, which can lead to data loss. In the ESD scheme, the ECC-based fingerprints are only used as similarity identification to filter the non-duplicate cache lines. For these similar cache lines, ESD further conducts a byte-by-byte comparison to make sure whether they are identical or not. Moreover, though ESD only eliminates a major part of duplicate cache lines, the undetected duplicate cache lines cannot result in data loss or any correctness issues.

Third, different from the DeWrite scheme, which needs to maintain the consistency of fingerprints stored in the NVMM, ESD only stores the fingerprints of high reference counts in the memory controller cache and avoids the consistency issue. Besides the extended asynchronous DRAM refresh (eADR) equipped in the new Optane PMem 200 series [21] can be used, other persistence methods such as battery-backed write cache [4] and leveraging programming primitive enabled write back [34] are orthogonal to ESD.

F. Discussion

In lieu of Optane PM, Intel pivots towards Compute Express Link (CXL) technology [15], which allows attaching volatile and non-volatile memory to a CPU over a CXL-capable PCIe bus. This would accomplish many of the same goals as Optane PM without the costs of developing Optane PM technology [23], [28], [52]. However, ESD is a lightweight ECC-based selective deduplication scheme in the memory controller between LLC and NVMM that improves system performance and memory space efficiency. Therefore, ESD and CXL are orthogonal. They share the same goal to improve the system's memory performance and utilization.

IV. Performance Evaluation

A. Experimental Setup and Methodology

To evaluate the performance of ESD, we implement the prototype of ESD in the gem5 simulator [7] with NVMain [42]. The gem5 is a system cycle-accurate simulator, and The NVMain is an NVM-based main memory simulator, which has been widely used in previous studies to accurately simulate the timing, cycle times, and energy consumption of the NVMM system. Table I shows the system configuration parameters exploited in the experimental simulators. The cache on the CPU-side comprises three levels which consist of the private caches (e.g., L1 and L2) and the shared L3 cache [61], [72]. The size of the cache line evicted from the CPU-core is 64 Bytes which is consistent with that in the real-system [4], [10], [55], [66]. The metadata cache is in the memory controller [70], [72]. The PCM parameters are configured to be consistent with that used in the previous studies [5], [32], [61].

To comprehensively evaluate the efficiency of ESD, we implement and evaluate the other related deduplication-based NVMM schemes, e.g., the Baseline without deduplication, the traditional SHA1-based deduplication (Dedup_SHA1) and the DeWrite [73]. The Dedup_SHA1 uses the SHA1-based fingerprint to detect and filter duplicate cache lines. For experimental evaluation, we use 20 applications with default settings that represent different areas in real-world applications, including

TABLE I
THE SYSTEM CONFIGURATIONS IN THE SIMULATORS.

Processor and Cache	
CPU	8 cores, X86-64 processor, 2GHZ Clock
L1 cache (private)	32KB, 8-way with 64B cache line, 2-cycle latency
L2 cache (private)	256KB, 8-way with 64B cache line, 8-cycle latency
L3 cache (shared)	16MB, 8-way with 64B cache line, 25-cycle latency
Cache Line Size	64B
Main Memory (PCM) for the System	
Capacity	16GB
PCM Latency	Read/Write: 75ns/150ns [5], [32]
PCM Energy	Read/Write: 1.49nJ/6.75nJ [13]
Metadata Cache	EFIT (512 KB), AMT (512 KB)

12 applications from the SPEC CPU 2017 [8] and 8 applications from the PARSEC [22]. For example, these applications from the SPEC CPU 2017 consist of various types, e.g., integer throughput (int_rate), integer speed (int_speed), floating-point throughput (fp_rate), and floating-point speed (fp_speed), and these applications from the PARSEC comprise a set of multi-threads benchmark tests. These 20 applications are also used to evaluate memory performance in the previous studies [4], [51], [66]. For all deduplication-based schemes, the NVMM system is warmed up by issuing about 1 billion requests in the initialization phase before each evaluation. We mainly compare the ESD scheme with the Dedup_SHA1 scheme and the DeWrite scheme. To make a comprehensive comparison in terms of the write endurance, performance, energy, and tail latency, the Baseline system is configured without deduplication.

B. Endurance

One design objective of ESD is to reduce the write traffic to NVMM and enhance its endurance by eliminating these duplicate cache lines. Figure 11 shows the write reductions by the different deduplication schemes, normalized to the Baseline scheme without deduplication. We can see that ESD reduces cache line writes by 47.8% on average across all 20 applications and by up to 99.9% for the *deepsjeng* and *roms* applications. ESD also can eliminate more than 80.0% of cache line writes, though some applications contain lots of non-zero duplicate writes, e.g., *lbm, mcf* and *roms*.

Meanwhile, we also observe that these write operations reduced by ESD (on average 47.8%) are less than the cache line writes reduced by the Dedup_SHA1 and the DeWrite schemes, by 18.3% and 18.2% on average for the Dedup_SHA1 scheme and DeWrite scheme, respectively. The reason is that ESD leverages an ECC-assisted and selective deduplication method, which exploits the content locality by only storing the ECC-based fingerprints with high reference counts in the cache. Hence, ESD can eliminate these duplicate cache lines but miss some duplicate cache lines whose reference counts are not large enough (i.e., only 18.3%).

By contrast, both the Dedup_SHA1 scheme and the DeWrite scheme use the full deduplication method that aims to detect and eliminate all duplicate cache lines. As a result, both of them need to store all fingerprints in the memory cache and NVMM, which incurs extra NVMM space and access

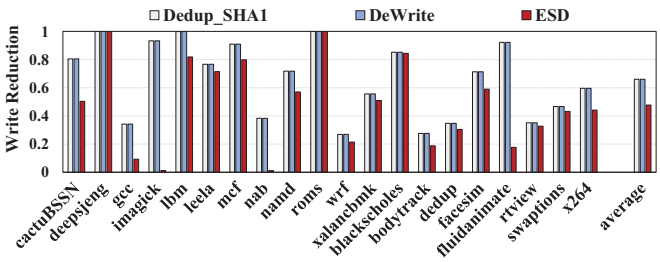


Fig. 11. Write reductions by different schemes.

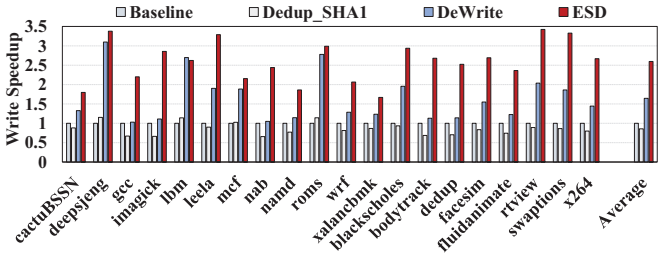


Fig. 12. Write speedup normalized to the Baseline.

overhead. However, storing full fingerprints contributes a little to redundancy detection and elimination but incurs huge space and performance overhead. The reason is that a large number of fingerprint items with a reference count value of 1 are rarely accessed and queried after they are stored in NVMM.

C. System Performance

Write Speedup: Write speedup is denoted as the write latency of the Baseline scheme divided by the other schemes, e.g., the Dedu_SHA1 scheme, the DeWrite scheme, and the ESD scheme. Figure 12 shows that the ESD scheme speeds up the system's write performance for all applications compared to the Baseline system by eliminating approximately 47.8% of data redundancy, with up to 3.4x. Meanwhile, the ESD can further speed the system's write performance by up to 4.3x and 2.6x for Dedup_SHA1 and DeWrite, respectively. For lbm, due to the content locality and accurate prediction, the write performance of DeWrite is superior to ESD. However, Dedup_SHA1 can speed up the system's write performance only for a few applications, e.g., *deepsjeng*, *lbm*, and *roms*.

ESD reduces the write latency from the following two aspects. First, ESD eliminates the computational latency of costly hash calculations because it employs the existing ECC information on the critical write path, though the reduced write traffic is less than full deduplication schemes (e.g., Dedup_SHA1 and DeWrite). Second, ESD only stores part of the fingerprints in a cache which avoids the fingerprint NVMM_lookup overhead as that in the Baseline and the DeWrite schemes. By only checking the ECC-based fingerprints in memory, the fingerprint lookup process could be significantly accelerated on the critical write path. Hence, in terms of the write performance, ESD outperforms both the latest scheme (i.e., DeWrite) and the Dedup_SHA1 scheme.

Read Speedup: Read speedup is denoted as the read latency of the Baseline scheme divided by the other schemes, e.g., the Dedu_SHA1 scheme, the DeWrite scheme, and the ESD scheme. Figure 13 shows that ESD can speed up the read performance for all applications compared to the Baseline

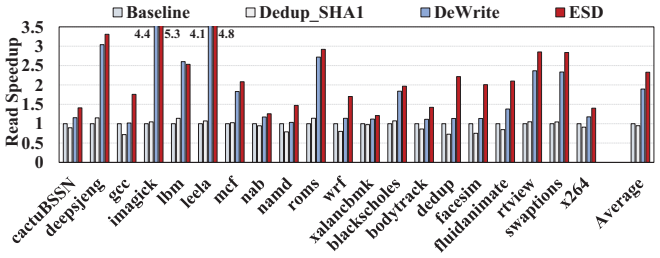


Fig. 13. Read speedup normalized to the Baseline.

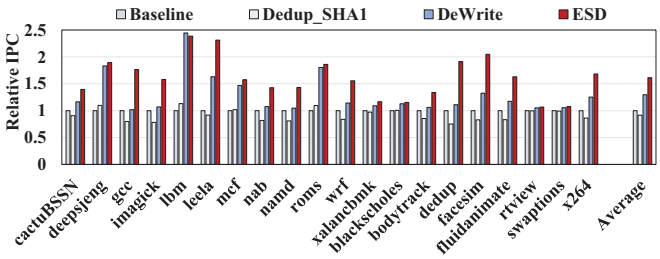


Fig. 14. IPC improvements normalized to the Baseline.

system, with up to 5.3x. Meanwhile, the ESD further speeds the read performance by up to 5x and 2x for Dedup_SHA1 and DeWrite, respectively. For lbm, due to the content locality and accurate prediction, the read performance of DeWrite is superior to ESD. However, Dedup_SHA1 can degrade system performance in most applications and speeds up the read performance compared with the Baseline scheme only in a few applications, e.g., *deepsjeng*, *leela*, *lbm*, *roms*.

ESD reduces the read latency from the following two aspects. First, ESD reduces a majority of duplicate cache line writes, which allows more device resources to serve read requests, such as the I/O bus and memory banks. Second, ESD can significantly eliminate duplicate cache lines that will be written to NVMM. Therefore, the read/write interference is significantly alleviated, which also reduces the waiting times for reading operations in the queue.

Instructions Per Cycle: By eliminating the duplicate cache line writes and speeding up reads, ESD can improve the overall NVMM system performance in terms of the Instructions Per Cycle (IPC). Figure 14 shows the relative IPC for different schemes, normalized to that of the Baseline scheme. The observation shows that the ESD scheme speeds up the read performance for all applications compared to the Baseline system by up to 2.4x. Meanwhile, the ESD further speeds the IPC by up to 2.5x and 1.8x for Dedup_SHA1 and DeWrite, respectively. However, Dedup_SHA1 decrease the IPC value in most applications and speeds up the IPC compared with the Baseline scheme only in a few applications, such as *deepsjeng*, *lbm*, *leela*, *roms*.

D. Tail Latency

The tail latency is an important performance metric in large-scale storage systems, e.g., Google, Facebook, and Amazon [18], [31]. The tail latency also reflects the Quality of Service (QoS) of a storage system [50], [69], especially for the latency-sensitive memory systems which are being gradually replaced by NVMM.

We investigate the latency distributions of the Dedup_SHA1, DeWrite, and ESD schemes across 20 applications. Figure 15 shows the tail latency results of 8 applications that are selected from SPEC CPU 2017 and PARSEC, e.g., *gcc*, *leela*, *bodytrack*, *dedup*, *facesim*, *fluidanimate*, *wrf* and *x264*. We can observe that the ESD scheme has much shorter tail latencies than the Dedup_SHA1 scheme, which is a SHA1-based deduplication approach, and the DeWrite scheme, which is the state-of-the-art deduplication scheme designed for NVMM.

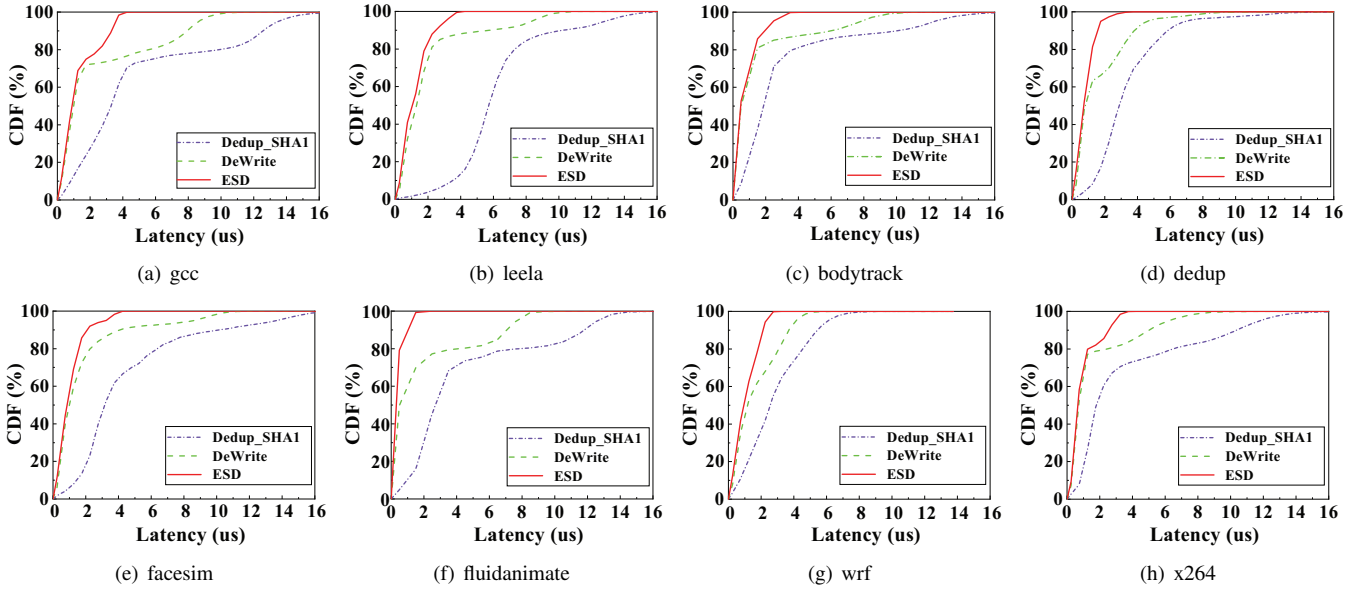
The reason that ESD significantly reduces the tail latency is illustrated as follows. First, ESD is an ECC-assisted deduplication scheme that eliminates the computational latency generated by the expensive hash calculation function. Hence, the hash calculation latency is directly removed from the critical write path. Second, ESD leverages an ECC-assisted and selective deduplication approach which can also remove a majority of duplicate writes to further reduce the latency of both reads and writes, even though ESD reduces fewer duplicate cache lines than the other schemes by about 18.3%. Third, compared with the ESD scheme, the Dedup_SHA1 scheme and the DeWrite scheme need a lot of extra costly operations, including the hash calculations and fingerprint NVMM_lookup operations on the critical write path, which significantly increase the write latency. Hence, the ESD scheme has the lowest tail latency compared with the Dedup_SHA1 scheme and the DeWrite scheme.

E. Energy Consumption

Energy consumption is an important performance metric of system efficiency, which includes read/write energy, encryption-induced energy, and deduplication-induced computing energy. We refer to the energy consumption model for CRC and SHA1 [56]. Figure 16 shows the energy consumption results, normalized to the Baseline. We can observe that the ESD can reduce the energy consumption across 20 applications, with up to 69.3%, 69.2%, and 56.6% compared to Baseline, Dedup_SHA1, and DeWrite, respectively. The reason is twofold. First, for the deduplication logic, ESD exploits the existing ECC information to eliminate computational energy consumption. However, for the Dedup_SHA1 and the DeWrite schemes, the energy consumption of calculating the SHA1 and CRC fingerprints by the CPU-core cannot be negligible. Second, ESD uses the selective deduplication method, which avoids the NVMM-based fingerprint accesses to further reduce energy consumption, compared to the full deduplication method, which incurs a large number of extra NVMM accesses to fetch the fingerprints. As a result, ESD can avoid fingerprint-induced energy consumption, including both the fingerprint computation for all cache lines and searching for the cache-missed cache lines, during deduplication-based NVMM systems.

F. Write Latency Profile

To understand the impact of latency on critical write path after adding deduplication for different schemes, the write



(a) gcc (b) leela (c) bodytrack (d) dedup

(e) facesim (f) fluidanimate (g) wrf (h) x264

Fig. 15. The CDF of write latency.

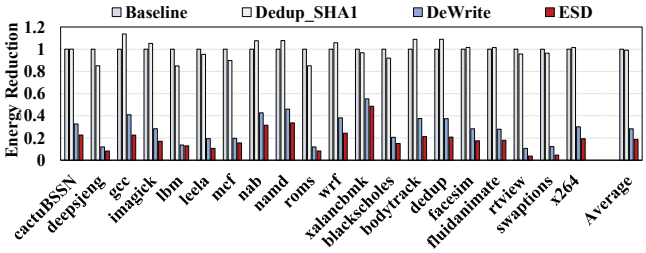


Fig. 16. The energy consumption results normalized to the Baseline.

latency can be divided into four parts: the fingerprints computation, the fingerprints NVMM_lookup, reading the similar cache lines, and writing the unique cache lines. Figure 17 shows the comparison and profile of write latency distributions for different deduplication schemes.

First, we observe that most of the write latency is caused by the fingerprint computation by approximately 80% in the Dedup_SHA1 scheme. It implies that SHA1-based fingerprints are not suitable for deduplication-enabled low-latency NVMM systems. Even for the lightweight CRC-based fingerprints used in DeWrite, the fingerprint computation contributes about 10% to write latency. Therefore, these results confirm that our ECC-assisted non-duplicate cache line filter is feasible for a deduplication-enabled low-latency NVMM system.

Second, we observe that approximately 12% and 23% of the write latency is spent on the fingerprints NVMM_lookup process in the Dedup_SHA1 and the DeWrite schemes, respectively. Both the Dedup_SHA1 and the DeWrite schemes use the full deduplication method by storing the full fingerprints in the NVMM and trying to eliminate all the duplicate cache lines, which can incur extra fingerprints NVMM_lookup operations. However, as illustrated before, these extra fingerprints NVMM_lookup operations contribute little by less than 18.3% to the redundancy elimination but incur significant performance and space overhead which should be optimized.

By contrast, we can see that ESD can eliminate the fingerprints computation because it employs the already existing ECC information to identify the data similarity without any computational overhead. Moreover, ESD also can eliminate the fingerprints NVMM_lookup overhead by only storing these ECC-based fingerprints with high reference counts in the cache and conducting the selective deduplication method. Though selective deduplication may miss eliminating approximately 18.3% duplicate cache lines, it avoids the significant performance and space overhead induced by storing the full fingerprints for full deduplication methods. We can see that the write latency is dominated by the reads and writes of the cache lines in the ESD scheme. As a result, the ESD scheme can provide higher performance than the Dedup_SHA1 and DeWrite schemes by eliminating the fingerprints computational and NVMM_lookup operations.

G. Sensitivity Analysis and Space Overhead

The cache size of the EFIT directly affects the deduplication efficiency and the performance of the ESD scheme. Figure 18 shows the cache hit rates for EFIT and AMT under different cache sizes in ESD. Figure 18(a) shows that the cache hit rates first increase along with the increase of the EFIT cache size. However, when the cache size increases over 512KB, the hit rate does not increase significantly, e.g., the hit rate only increases by 0.25% and 0.16% from 512KB to 1024KB, and by 0.03% and 0.02% from 1024KB to 2048KB with and without LRCU, respectively. Meanwhile, Figure 18(b) shows that the cache hit rates keep increasing as the AMT cache size gets progressively larger. The cache hit rate continues to increase, but the increasing percentage is getting much smaller after the cache size reaches 512KB, e.g., the cache hit rate only increases by 0.5% from 512KB to 1024KB, and by 0.6% from 1024KB to 2048KB. These results indicate that further increasing the cache size by storing more ECC items in the memory cache does not help to detect and eliminate more duplicate cache lines. With a larger cache size, more ECC

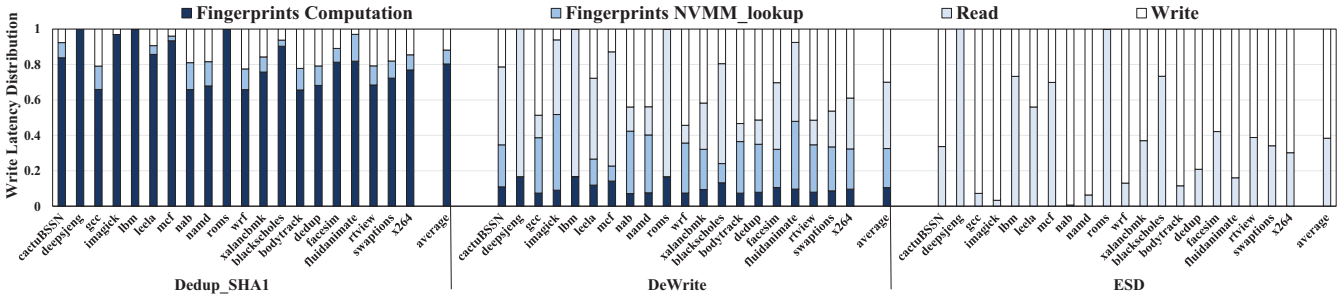
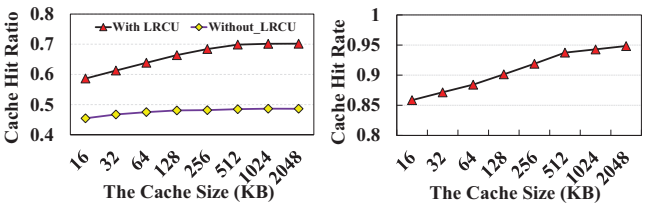


Fig. 17. The comparison and profile of write latency distributions for different deduplication schemes.



(a) The EFIT Cache (b) The AMT Cache

Fig. 18. The cache hit rates for EFIT (with and without LRCU) and AMT under different cache sizes.

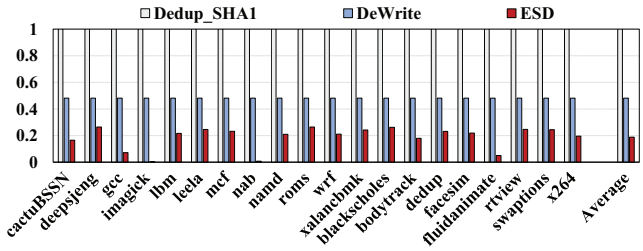


Fig. 19. The metadata overhead of different schemes normalized to the Dedup_SHA1 scheme.

items can be stored in memory, and more duplicate cache lines can be detected. However, with a cache size of 512KB for EFIT, ESD can already detect a majority of duplicate cache lines by ECC-based fingerprints stored in the memory cache. Hence, these results also validate that the selective deduplication method used in the ESD scheme is feasible.

The memory footprint of metadata is a major part of space overhead. The fingerprint metadata store can incur the NVMM space overhead in the Dedup_SHA1 and the DeWrite schemes, both of which use the full deduplication method. The DeWrite scheme needs to maintain (16 Bytes + 3 bits) for each physical cache line, resulting in a metadata space overhead of about 25.59%. For the Dedup_SHA1 scheme with 160 bits SHA1 fingerprint, the metadata space overhead is much larger than the DeWrite scheme. By contrast, ESD only stores fingerprints with high reference counts in the memory cache for selective deduplication and maintains the AMT table in the NVMM, thus completely eliminating the fingerprint store-induced NVMM space overhead. Figure 19 shows the metadata overhead of different schemes normalized to the Dedup_SHA1 scheme. Compared with the other two schemes, ESD significantly reduces metadata space overhead by 81.2% and 60.9%, respectively.

V. Related Work

Data deduplication is an effective way to improve memory space efficiency by removing duplicate cache lines [2], [45], [49], [60]. Recent studies have shown that inline deduplication can effectively reduce the write traffic to NVMs [40], [53], [73]. For example, NV-Dedup [53] applies the inline deduplication algorithm to the NVM-oriented file systems to deduplicate data while achieving high performance. It maintains a CPU and NVM-favored metadata table and uses a lightweight scheme for metadata consistency. DeWrite [73] leverages the intrinsic read/write asymmetry of NVMs and the lightweight CRC hashing algorithm by performing full deduplication to reduce all the duplicate cache lines to NVMs. Base and Compressed Difference (BCD) deduplication [40] effectively utilizes the partial matches among cache lines through an inter-block diff compression with deduplication to increase the effective capacity of main memory.

However, all these schemes perform full deduplication for NVMs, which tries to eliminate all duplicate cache lines (DeWrite [73] and BCD [40]) or data blocks (NV-Dedup [53]). As a result, the computational and memory space overhead induced by full deduplication is significant and leads to performance degradation for NVMMs. Different from these studies, ESD first exploits the ECC within cache lines to identify the data similarity, thus eliminating the costly computational overhead. It further performs selective deduplication for NVMs by only storing the fingerprints with a high reference count in memory to exploit the content locality.

VI. Conclusion

The workload analysis and preliminary evaluations reveal that existing deduplication-based studies would degrade the NVMM system performance. We propose ESD that exploits both the media characteristics (ECC mechanism) to identify the duplicate cache lines and the workload characteristics (content locality) to implement selective deduplication in deduplication-based NVMM. Our experimental evaluation demonstrates that ESD significantly outperforms the existing SHA1-based and DeWrite approaches.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant No. U22A2027, No. 61972325, and No. 61872305, Open Research Projects of Zhejiang Lab (No. 2021DA0AM01/002), and Open Project Program of Wuhan National Laboratory for Optoelectronics No. 2021WNLOKF011.

REFERENCES

[1] I. Alam, C. Schoeny, L. Dolecek, and P. Gupta, "Parity++: Lightweight Error Correction for Last Level Caches," in *Proceedings of the 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSNW'18)*, Los Angeles, CA, USA, Jun. 2018.

[2] A. Albalawi, V. Vassilakis, and R. Calinescu, "Memory Deduplication as A Protective Factor in Virtualized Systems," in *Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS'21)*, Kamakura, Japan, Jun. 2021.

[3] AMD EPYC Processors, " https://www.amd.com/zh-hans/processors/epyc-server-cpu-family," Sep. 2021.

[4] A. Awad, P. Manadhata, S. Haber, Y. Solihin, and W. Horne, "Silent Shredder: Zero-cost Shredding for Secure Non-volatile Main Memory Controllers," in *Proceedings of the 21st International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'16)*, New Orleans, LA, USA, Apr. 2016.

[5] A. Awad, M. Ye, Y. Solihin, L. Njilla, and K. Zubair, "Triad-nvm: Persistency for Integrity-protected and Encrypted Non-volatile Memories," in *Proceedings of the 46th International Symposium on Computer Architecture (ISCA'19)*, Phoenix, AZ, USA, Jun. 2019.

[6] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-Locked Encryption and Secure Deduplication," in *Proceedings of Annual international conference on the theory and applications of cryptographic techniques (EUROCRYPT'13)*, Berlin, Heidelberg, Sep. 2013.

[7] N. Binkert, B. Beckmann, G. Black, S. Reinhardt, A. Saidi, A. Basu, J. Hestness, R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, D. Hill, and A. Wood, "The Gem5 Simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, May 2011.

[8] J. Bucek, K. Lange, and J. Kistowski, "SPEC CPU2017: Next-generation Compute Benchmark," in *Proceedings of the 9th ACM/SPEC International Conference on Performance Engineering (ICPE'18)*, Berlin, Germany, Apr. 2018.

[9] M. Cai, C. Coats, and J. Huang, "Hoop: Efficient Hardware-assisted Out-of-place Update for Non-Volatile Memory," in *Proceedings of the 47th International Symposium on Computer Architecture (ISCA'20)*, Virtual Event, May 2020.

[10] D. Carvalho and A. Seznec, "Conciliating Speed and Efficiency on Cache Compressors," in *Proceedings of the 39th IEEE International Conference on Computer Design (ICCD'21)*, Virtual Event, Oct. 2021.

[11] R. Chaves, L. Sousa, N. Sklavos, A. Fournaris, G. Kalogeridou, P. Kitsos, and F. Sheikh, "Secure Hashing: SHA-1, SHA-2, and SHA-3," *Circuits and Systems for Security and Privacy*, pp. 105–132, May 2016.

[12] L. Chen, Y. Cao, and Z. Zhang, "Free ECC: An Efficient Error Protection for Compressed Last-level Caches," in *Proceedings of the 31st International Conference on Computer Design (ICCD'13)*, Phoenix, AZ, USA, May 2013.

[13] Z. Chen, Y. Zhang, and N. Xiao, "ExtraCC: Improving Performance of Secure NVM with Extra Counters and ECC," in *Proceedings of the 36th International Conference on Massive Storage Systems and Technology (MSST'20)*, Virtual Event, Oct. 2020.

[14] S. Cho and H. Lee, "Flip-N-Write: A Simple Deterministic Technique to Improve PRAM Write Performance, Energy and Endurance," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'09)*, New York, USA, Dec. 2009.

[15] Compute Express Link (CXL) of Intel, " https://www.intel.cn/content/www/cn/zh/products/details/fpga/intellectual-property/interface-protocols/cxl-ip.html," Mar. 2022.

[16] L. Cox, C. Murray, and B. Noble, "Pastiche: Making Backup Cheap and Easy," in *Proceedings of 5th Symposium on Operating Systems Design and Implementation (OSDI'02)*, Boston, MA, Dec. 2002.

[17] G. De, B. Larry, A. Sison, and R. Medina, "MD5 Secured Cryptographic Hash Value," in *Proceedings of the 2018 International Conference on Machine Learning and Machine Intelligence (MLMI'18)*, Ha Noi, Viet Nam, Sep. 2018.

[18] J. Dean and L. Barroso, "The Tail at Scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, Feb. 2013.

[19] J. Douceur, A. Adya, W. Bolosky, D. Simon, and M. Theimer, "Reclaiming Space from Duplicate Files in A Serverless Distributed File System," in *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, Vienna, Austria, Jul. 2002.

[20] ECC for L2 Cache Data Memory, " https://www.intel.com/content/www/us/en/docs/programmable/683360/18-0/ecc-for-l2-cache-data-memory.html," Mar. 2022.

[21] Extended Asynchronous DRAM Refresh (eADR), " https://www.intel.cn/content/www/cn/zh/products/docs/memory-storage/optane-persistent-memory/optane-persistent-memory-200-series-brief.html," Mar. 2021.

[22] M. Gebhart, J. Hestness, E. Fatehi, P. Gratz, and S. Keckler, "Running PARSEC 2.1 on M5," *Technical Report TR-09-32, The University of Texas at Austin, Department of Computer Science*, Oct. 2009.

[23] D. Gouk, S. Lee, M. Kwon, and M. Jung, "Direct Access, High-Performance Memory Disaggregation with DirectCXL," in *Proceedings of the 2022 USENIX Annual Technical Conference (ATC'22)*, Carlsbad, CA, Jul. 2022.

[24] S. Ikeda, K. Miura, H. Yamamoto, K. Mizunuma, H. Gan, M. Endo, S. Kanai, J. Hayakawa, F. Matsukura, and H. Ohno, "A perpendicular-anisotropy CoFeB–MgO Magnetic Tunnel Junction," *Nature Materials*, vol. 9, no. 9, pp. 721–724, Jul. 2010.

[25] Intel Xeon Processor E5 Family, " https://www.intel.com/content/www/us/en/processors/xeon/xeon-e5-family-spec-update.html," May 2020.

[26] L. Jaulmes, M. Moreto, M. Valero, and M. Casas, "A Vulnerability Factor for ECC-protected Memory," in *Proceedings of the 25th International Symposium on On-Line Testing and Robust System Design (IOLTS'19)*, Rhodes Island, Greece, Jul. 2019.

[27] A. Joglekar, M. Kounavis, and F. Berry, "A Scalable and High Performance Software ISCSI Implementation," in *Proceedings of the 4th Conference on USENIX Conference on File and Storage Technologies (FAST'05)*, San Francisco, CA, USA, Dec. 2005.

[28] M. Jung, "Hello Bytes, Bye Blocks: PCIe Storage Meets Compute Express Link for Memory Expansion (CXL-SSD)," in *Proceedings of the 14th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage'22)*, Virtual Event, Jun. 2022.

[29] S. Keelveedhi, M. Bellare, and T. Ristenpart, "DupLESS: Server-Aided Encryption for Deduplicated Storage," in *Proceedings of the 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques (Security'13)*, Washington, D.C., Aug. 2013.

[30] K. Korgaonkar, I. Bhati, H. Liu, J. Gaur, S. Manipatruni, S. Subramoney, T. Karnik, S. Swanson, I. Young, and H. Wang, "Density Tradeoffs of Non-Volatile Memory as A Replacement for SRAM Based Last Level Cache," in *Proceedings of the 45th Annual International Symposium on Computer Architecture (ISCA'18)*, Los Angeles, CA, USA, Jun. 2018.

[31] A. Lange, T. Kepe, and M. Sunye, "Performance Interference on Key-Value Stores in Multi-tenant Environments: When Block Size and Write Requests Matter," in *Proceedings of the 12th ACM/SPEC International Conference on Performance Engineering (ICPE'21)*, Virtual Event, Apr. 2021.

[32] B. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting Phase Change Memory as A Scalable Dram Alternative," in *Proceedings of the 36th Annual International Symposium on Computer Architecture (ISCA'09)*, Austin, Texas, USA, Jun. 2009.

[33] T. Lehman, A. Hilton, and B. Lee, "PoisonIvy: Safe Speculation for Secure Memory," in *Proceedings of the 49th International Symposium on Microarchitecture (MICRO'16)*, Boston, MA, USA, Oct. 2016.

[34] S. Liu, A. Kolli, J. Ren, and S. Khan, "Crash Consistency in Encrypted Non-volatile Main Memory Systems," in *Proceedings of the 24th IEEE International Symposium on High Performance Computer Architecture (HPCA'18)*, Vienna, Austria, Feb. 2018.

[35] R. Maddah, S. Seyedzadeh, and R. Melhem, "CAFO: Cost Aware Flip Optimization for Asymmetric Memories," in *Proceedings of the 21st International Symposium on High-Performance Computer Architecture (HPCA'15)*, Seoul, South Korea, Feb. 2015.

[36] A. Malek, E. Vasilakis, V. Papaefstathiou, P. Trancoso, and I. Sourdis, "Odd-ECC: On-Demand DRAM Error Correcting Codes," in *Proceedings of the 2017 International Symposium on Memory Systems (MEMSYS'17)*, Alexandria, Virginia, Oct. 2017.

[37] O. Mutlu, "Memory Scaling: A Systems Architecture Perspective," in *Proceedings of the 5th IEEE International Memory Workshop (IMW'13)*, Monterey, CA, USA, May 2013.

[38] P. Nair, C. Chou, B. Rajendran, and M. Qureshi, "Reducing Read Latency of Phase Change Memory via Early Read and Turbo Read," in *Proceedings of the 21st International Symposium on High-Performance Computer Architecture (HPCA'15)*, Burlingame, CA, USA, Feb. 2015.

[39] P. Nair, D. Kim, and M. Qureshi, "ArchShield: Architectural Framework for Assisting DRAM Scaling by Tolerating High Error Rates," in *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA'13)*, New York, USA, Jun. 2013.

[40] S. Park, I. Kang, Y. Moon, J. H. Ahn, and G. Suh, "BCD Deduplication: Effective Memory Compression Using Partial Cache-line Deduplication," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'21)*, Lausanne, Switzerland, Feb. 2021.

[41] M. Patel, J. Kim, T. Shahroodi, H. Hassan, and O. Mutlu, "Bit-exact Ecc Recovery (BEER): Determining DRAM on-die ECC Functions by Exploiting DRAM Data Retention Characteristics," in *Proceedings of the 53rd International Symposium on Microarchitecture (MICRO'20)*, Virtual Event, Sep. 2020.

[42] M. Poremba, T. Zhang, and Y. Xie, "NVMain 2.0: A User-friendly Memory Simulator to Model Non-Volatile Memory Systems," *IEEE Computer Architecture Letters*, vol. 14, no. 2, pp. 140–143, Feb. 2015.

[43] Qualcomm Centriq 2400 Processor, "https://www.qualcomm.com/news/onq/2017/10/05/qualcomm-centriq-2400-designed-scalability-and-throughput-performance," Oct. 2017.

[44] G. Saileshwar, P. Nair, P. Ramrakhyani, W. Elsasser, and M. Qureshi, "Synergy: Rethinking Secure-Memory Design for Error-Correcting Memories," in *Proceedings of the 24th IEEE International Symposium on High Performance Computer Architecture (HPCA'18)*, Vienna, Austria, Feb. 2018.

[45] D. Saxena, T. Ji, A. Singhvi, J. Khalid, and A. Akella, "Memory Deduplication for Serverless Computing with Medes," in *Proceedings of the 17th European Conference on Computer Systems (EuroSys'22)*, RENNES, France, Apr. 2022.

[46] S. Schechter, G. Loh, K. Strauss, and D. Burger, "Use ECP, Not ECC, for Hard Failures in Resistive Memories," *ACM SIGARCH Computer Architecture News*, vol. 38, no. 3, pp. 141–152, Jun. 2010.

[47] S. Silvestro, H. Liu, T. Zhang, C. Jung, D. Lee, and T. Liu, "Sampler: PMU-Based Sampling to Detect Memory Errors Latent in Production Software," in *Proceedings of the 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'18)*, Fukuoka, Japan, Oct. 2018.

[48] K. Srinivasan, T. Bisson, G. Goodson, and K. Voruganti, "iDedup: Latency-aware, Inline Data Deduplication for Primary Storage," in *Proceedings of the 10th USENIX Conference on File and Storage Technologies (FAST'12)*, San Jose, CA, USA, Feb. 2012.

[49] J. Stevenson, *Fine-grain In-Memory Deduplication for Large Scale Workloads*. Stanford University, 2013.

[50] C. Sun, D. Moal, Q. Wang, R. Mateescu, F. Blagojevic, M. Lueker-Boden, C. Guyot, Z. Bandic, and D. Vucinic, "Latency Tails of Byte-addressable Non-volatile Memories in Systems," in *Proceedings of the 2017 IEEE International Memory Workshop (IMW'17)*, Monterey, CA, USA, May 2017.

[51] S. Swami, J. Rakshit, and K. Mohanram, "SECRET: Smartly Encrypted Energy Efficient Non-volatile Memories," in *Proceedings of the 53rd Design Automation Conference (DAC'16)*, Austin, Texas, USA, Jun. 2016.

[52] D. Waddington, M. Hershcovitch, and C. Dickey, "PyMM: Heterogeneous Memory Programming for Python Data Science," in *Proceedings of the 11th Workshop on Programming Languages and Operating Systems (PLOS'05)*, Virtual Event, Oct. 2021.

[53] 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*, vol. 67, no. 5, pp. 658–671, May 2018.

[54] R. Wang, L. Jiang, Y. Zhang, and J. Yang, "SD-PCM: Constructing Reliable Super Dense Phase Change Memory under Write Disturbance," in *Proceedings of the 20th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'15)*, Istanbul Turkey, Mar. 2015.

[55] Z. Wang, X. Liu, J. Yang, T. Michailidis, S. Swanson, and J. Zhao, "Characterizing and Modeling Non-volatile Memory Systems," in *Proceedings of the 53rd International Symposium on Microarchitecture (MICRO'20)*, Athens, Greece, Oct. 2020.

[56] B. Westermann, D. Gligoroski, and S. Knapskog, "Comparison of the Power Consumption of the 2nd Round SHA-3 Candidates," in *International Conference on ICT Innovations (ICCS'10)*, Ohrid Macedonia, Sep. 2010.

[57] P. Wong, H. Lee, Y. Shimeng, C. YuSheng, W. Yi, C. Pang-Shiu, L. Byoungil, C. Frederick, and T. Ming-Jinn, "Metal–Oxide RRAM," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951–1970, May 2012.

[58] S. Wu, J. Wu, Z. Shen, Z. Zhang, Z. Wang, and B. Mao, "SimiEncode: A Similarity-based Encoding Scheme to Improve Performance and Lifetime of Non-Volatile Main Memory," in *Proceedings of the 39th IEEE International Conference on Computer Design (ICCD'21)*, Virtual Event, Oct. 2021.

[59] S. Wu, J. Zhou, W. Zhu, H. Jiang, Z. Huang, Z. Shen, and B. Mao, "EaD: A Collision-free and High Performance Deduplication Scheme for Flash Storage Systems," in *Proceedings of the IEEE 38th International Conference on Computer Design (ICCD'20)*, Virtual Event, Oct. 2020.

[60] J. Xiao, Z. Xu, H. Huang, and H. Wang, "Security Implications of Memory Deduplication in A Virtualized Environment," in *Proceedings of the 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'13)*, Budapest, Hungary, Jun. 2013.

[61] C. Xu, D. Niu, N. Muralimanohar, R. Balasubramonian, T. Zhang, S. Yu, and Y. Xie, "Overcoming the Challenges of Crossbar Resistive Memory Architectures," in *Proceedings of the IEEE 21st International Symposium on High Performance Computer Architecture (HPCA'15)*, Burlingame, CA, USA, Feb. 2015.

[62] F. Yang, Y. Lu, Y. Chen, H. Mao, and J. Shu, "No Compromises: Secure NVM with Crash Consistency, Write-Efficiency and High-performance," in *Proceedings of the 56th ACM/IEEE Design Automation Conference (DAC'19)*, Las Vegas, NV, USA, Jun. 2019.

[63] Z. Yang, J. Li, and P. P. C. Lee, "Secure and Lightweight Deduplicated Storage via Shielded Deduplication-Before-Encryption," in *Proceedings of 2022 USENIX Annual Technical Conference (ATC'22)*, Carlsbad, CA, Jul. 2022.

[64] M. Ye, C. Hughes, and A. Awad, "Osiris: A Low-Cost Mechanism to Enable Restoration of Secure Non-Volatile Memories," in *Proceedings of the 51st International Symposium on Microarchitecture (MICRO'18)*, Fukuoka, Japan, Oct. 2018.

[65] D. H. Yoon and M. Erez, "Memory Mapped ECC: Low-cost Error Protection for Last Level Caches," in *Proceedings of the 36th Annual International Symposium on Computer Architecture (ISCA'09)*, Austin, Texas, USA, Jun. 2009.

[66] V. Young, P. Nair, and M. Qureshi, "DEUCE: Write-efficient Encryption for Non-volatile Memories," *ACM SIGARCH Computer Architecture News*, vol. 43, no. 1, pp. 33–44, Mar. 2015.

[67] J. Yue and Y. Zhu, "Accelerating Write by Exploiting PCM Asymmetries," in *Proceedings of the IEEE 19th International Symposium on High Performance Computer Architecture (HPCA'13)*, Phoenix, AZ, USA, Feb. 2013.

[68] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology," in *Proceedings of the 36th Annual International Symposium on Computer Architecture (ISCA'09)*, Austin Texas, USA, Jun. 2009.

[69] Y. Zhou, R. Alagappan, A. Memaripour, A. Badam, and D. Wentzlaff, "HNVM: Hybrid NVM Enabled Datacenter Design and Optimization," *Microsoft Research TR*, no. 8, Feb. 2017.

[70] A. Zubair and A. Awad, "Anubis: Ultra-Low Overhead and Recovery Time for Secure Non-Volatile Memories," in *Proceedings of the 46th International Symposium on Computer Architecture (ISCA'19)*, Phoenix, AZ, USA, Jun. 2019.

[71] A. Zubair, S. Gurumurthi, V. Sridharan, and A. Awad, "Soteria: Towards Resilient Integrity-Protected and Encrypted Non-Volatile Memories," in *Proceedings of the 54th International Symposium on Microarchitecture (MICRO'21)*, Virtual Event, Oct. 2021.

[72] P. Zuo, Y. Hua, and Y. Xie, "SuperMem: Enabling Application-transparent Secure Persistent Memory with Low Overheads," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'19)*, Columbus, OH, USA, Oct. 2019.

[73] P. Zuo, Y. Hua, M. Zhao, W. Zhou, and Y. Guo, "Improving the Performance and Endurance of Encrypted Non-volatile Main Memory Through Deduplicating Writes," in *Proceedings of the 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'18)*, Fukuoka Japan, Oct. 2018.

A ARTIFACT APPENDIX

A.1 Abstract

The artifact is the source code of the ESD prototype. It conducts a lightweight ECC-based similarity identification and selective deduplication scheme for NVMM systems, which validates the key ideas in this paper.

A.2 An itemized meta information list

- Program: ESD
- Compilation: Clang++ 11, scons 3.1
- Data set: SPEC CPU 2017 and PARSEC 2.1
- Programming: C, C++ 11, Python 3.8
- Publicly available?: Yes
- Output: Statistics of state information for reads, writes, energy, and latency.
- How much memory space is required (roughly)?: 64 GB
- How much time is needed to complete experiments (roughly)?: 20 hours

A.3 Access to the artifact

The artifact can be obtained from the repository of GitHub: https://github.com/ChunfengDu/ESD.

A.4 System requirements and dependencies

We developed and evaluated our artifact in a Linux server equipped with kernel version 5.4.0 (Ubuntu 18.04 or newer versions), a 24-cores Intel Xeon 5318Y 2.10GHz CPU, and 128GB DRAM. Meanwhile, these traces employed in the artifact are generated jointly by gem5 and real applications (e.g., SPEC CPU 2017 and PARSEC 2.1) that are described as follows:

- ***The SPEC CPU 2017***: cactuBSSN, deepsjeng, gcc, imagick,lbm, leela, mcf, nab, namd, roms, wrf, and xalancbmk.
- ***The PARSEC 2.1***: blackscholes, bodytarck, dedup, facesim, fluidanimate, rtview, swaptions, and x264.

SPEC CPU 2017 is a set of CPU subsystem testing tools, including a total of 43 tests in 4 categories. PARSEC 2.1 (The Princeton Application Repository for Shared-Memory Computers) is a test assembly composed of multi-threaded applications.

Note that the users also can generate other corresponding traces for running in the artifact, which must be kept in the same regulation format illustrated in README.md.

A.5 Steps for evaluation

Please obey the following steps for evaluation if you have a pure Linux server:

- **Step 1:** Update and install necessary dependencies: libssl-dev, openssl, GCC, libffi-dev, zlib1g-dev, libbz2-dev, zlibc, wget.
- **Step 2:** Install the compilation tools: scons (3.1.2) and python (3.8)
- **Step 3:** Run the following commands.

==========

 # git clone git@github.com:ChunfengDu/ESD.git
 # cd ESD
 # scons build-type=fast
 # ./nvmain.fast –ConfigFile=[file path of config] –InputFile=[Trace load] –cycles
 # Latency_Name (Note: Please input File name for storing the latency file)

Next, the users should select a scheme for executing from the following four schemes (e.g., 0: Baseline, 1: Tra_sha1, 2: DeWrite, 3: ESD).

==========

The user can use the shell script (run.sh or run_alone.sh) for automated execution illustrated in the repository of GitHub.

A.6 Expected results

The following main results could be obtained if users follow the above instructions step by step.

(1) The number of writes to NVMM (Figure 11)

(2) The average latency of write/read under different benchmarks (Figure 12 and Figure 13)

(3) The number of IPC under different schemes (Figure 14)

(4) The CDF of write latency (Figure 15)