EaD: a Collision-free and High Performance Deduplication Scheme for Flash Storage Systems

Suzhen Wu‡, Jindong Zhou‡, Weidong Zhu‡, Hong Jiang*, Zhijie Huang*, Zhirong Shen‡, Bo Mao‡✉
‡School of Informatics, Xiamen University, Xiamen, Fujian, China
*Department of Computer Science & Engineering at the University of Texas-Arlington, Texas, USA
{suzhen, shenzr, maobo}@xmu.edu.cn, zhoujindong@stu.xmu.edu.cn, zwdong1994@gmail.com,
{hong.jiang, zhijie.huang}@uta.edu

***Abstract*—Inline deduplication is a popular technique to effectively reduce the write traffic and improve the space efficiency for flash-based storage. However, it also introduces computing and memory overhead to generate and store the cryptographic hash (fingerprint). Along the advent of 3D XPoint and Z-NAND technologies with vastly improved latency and bandwidth, both the computing and memory overheads are becoming much more pronounced in deduplication-based flash storage with cryptographic hash functions in use. To address these problems, we propose an ECC (Error Correcting Code) assisted deduplication approach, called EaD, which exploits the ECC property and the asymmetric read-write performance characteristics of modern flash-based storage. EaD first identifies data similarity based on the fingerprints of data chunks represented by their ECC values, thus significantly reducing the costly cryptographic hash computing and alleviating the memory space overhead. Based on the identification results, similar data chunks and their ECCs are read from the flash to perform a byte-by-byte comparison in memory to definitively identify and remove redundant data chunks. Our experiments show that the EaD approach significantly reduces the I/O latency by an average of 1.92× and 1.86×, and reduces the memory consumption by an average of 35.0% and 21.9%, compared with the existing SHA- and sampling-based deduplication approaches, respectively.**

***Index Terms*—Data Deduplication, Flash Storage, ECC, Collision Free, Performance Evaluation**

I. Introduction

Flash-based devices have been extensively deployed in modern computer systems to satisfy the increasing demand for storage performance and energy efficiency. Due to the unique characteristics of the flash memory technology, the performance and reliability of flash storage is highly sensitive to the write traffic [25], [21]. Thus, techniques that can reduce the number of writes to flash storage are desirable and have received a lot of attention from both industry and academia [6], [11], [14]. The most popular and effective among these techniques is data deduplication, which has gained increasing traction due to its ability to reduce the storage space requirement by eliminating duplicate data and minimizing the transmission of redundant data in storage systems.

Recent studies [6], [14], [22] have shown that the ability of data deduplication to reduce the write traffic can significantly improve the performance and reliability of the flash storage systems. In fact, inline data deduplication has become a commodity feature in flash-based storage products for many leading companies, such as HPE Nimble Storage [5] and Pure Storage [11], for the purpose of enhancing system performance, reliability and space efficiency. However, despite of data deduplication's great benefits, it has two important drawbacks, namely, high computational and memory overheads on the I/O critical path, which can adversely affect the performance of such systems, and nonzero hash-collision probability, which can cause unrecoverable data corruption.

First, the computational intensity of cryptographic hash functions and memory consumption of fingerprints can lead to serious performance degradation of deduplication-based flash storage. Generally speaking, the deduplication process can be divided into four stages: (1) data chunking that divides data streams/files into roughly equal-sized chunks (often based on content), (2) hash computing for chunk fingerprints that uniquely identify data chunks, (3) index querying that determines whether incoming chunks are duplicates to be removed using fingerprints, and (4) index and metadata updating. More specifically, the MD5/SHA-based Content-Defined Chunking (CDC) algorithms need to compute the hash value of all chunk content, which significantly lengthens the write latency in deduplication-based storage systems, especially in the high performance flash-based devices with the 3D XPoint [1] and Z-NAND [7] technologies. The occupied memory to store fingerprints also implies less cache space to buffer user I/Os. Moreover, the computational and memory overheads can be a critical issue when data deduplication is embedded within flash-based SSDs [6], [14] and Smartphones [22] where the computing and memory resources are limited.

Second, all hash functions have potential collisions in which two different data chunks share the same hash value, although the collision probability depends on the specific hash function. Since most cryptographic hash functions produce a fixed size output from an arbitrarily long data chunks, there will always be collisions due to the loss of precision inherent in representing a larger data block with a smaller hash value. It is now well-known that the MD5 and SHA-1 functions have been cracked [12], [23]. For example, a recent collaborative study between the CWI Institute in Amsterdam and Google announced the first practical technique for generating a SHA-1 hash collision in February 2017 [12]. Though using a more secure hash algorithm like SHA-256 can reduce the probability

of hash collision, it also increases the computing overhead and the memory overhead significantly due to lower cryptographic hash speed and bigger hash length compared with SHA-1.

To simultaneously alleviate the performance and memory overheads and completely avoid the hash collision issue in traditional MD5/SHA-based deduplicating flash storage systems, this paper proposes an ECC assisted deduplication approach, called EaD, which exploits the ECC property and high read performance characteristics of modern flash-based SSDs to establish a collision-free and high performance deduplication system with low memory overhead. The main idea behind EaD is its use of the ECC information already available in each flash page in modern SSDs as the hash value of the chunk content to definitively filter out non-duplicate data chunks to prevent the unnecessary and costly hash computing and comparison of their content. This is possible because there is no false negative detection of duplicates by using ECC as a hash function. The unfiltered blocks are then considered at least similar, if not duplicate (due to false positives), to a stored data chunk since their ECC values match those of stored data chunks. Based on the preliminary identification results, the similar data chunks are read from the flash to perform a byte-by-byte comparison in memory to definitively identify and remove redundant data chunks. Our experiments results show that collision-free EaD significantly outperforms the existing SHA- and sampling-based deduplication approaches [6], [14] in terms of I/O performance by an average of 1.92× and 1.86×, respectively. Meanwhile, EaD also significantly reduces the memory consumption by an average of 35.0% and 21.9%, respectively.

The rest of this paper is organized as follows. Background and motivation are presented in Section II. We describe the design details of EaD in Section III. The performance evaluation is presented in Section IV and the related work is presented in Section V. We conclude this paper in Section VI.

II. Background and Motivation

A. A new performance landscape

The development of emerging storage technologies, such as Intel 3D XPoint [1] and Z-NAND [7], has significantly improved the performance of flash-based storage [30]. For example, Intel has integrated the 3D XPoint memory into Intel Optane series SSDs and Samsung has also deployed Z-NAND flash into Samsung 983 ZET flagship datacenter SSDs. Moreover, other manufacturers are also planning to release low-latency flash products, such as Everspin's nvNITRO Technology and Toshiba's XL-Flash [30]. On the other hand, the cryptographic hash functions used in deduplication-based storage systems for the purpose of data chunking and fingerprinting, such as SHA-1 and SHA-256, have remained unchanged. This has brought about a noticeable change in the performance landscape of flash-based deduplication storage systems in that the performance bottleneck is observed to shift from the I/O stack to the compute layer due to the costly computing of cryptographic hash functions.

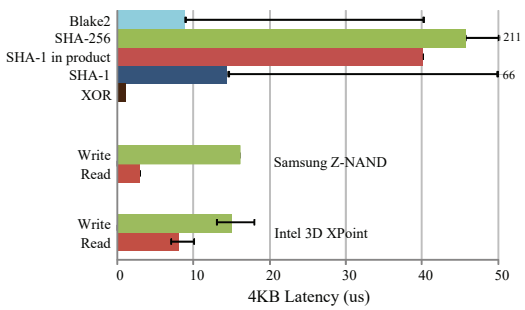


Fig. 1. A comparison of 4KB latencies of reading and writing from Intel Optane 3D XPoint and Samsung Z-NAND flash devices, and performing various computing functions including 4 different hash functions and the XOR operation.

Figure 1 compares the 4KB latencies of reading and writing from Intel Optane 3D XPoint [1] and Samsung Z-NAND flash devices [7], and latencies of performing various computing functions on a 4KB data page. It must be noted that the SHA-based hash computing latencies shown in the bars of Figure 1 are the lowest values derived from the fastest hashing implementations [8], [3], considering the ARM Cortex R5 processor is used with the maximum clock frequency of 1.4 GHz, in contrast to the minimum adjustable frequency of 300MHz [27]. The minimal clock frequency is set to 300MHz. In real SSD products, the SHA-based hash computing latency could be much higher [6], [14], [15], [24]. For example, the latest released Marvell NVMe SSD Controllers (88SS1092 and 88SS1093) use triple ARM Cortex R5 up to 500MHz [15], which implies that the SHA-1 latency of a 4KB page will be 40us instead of the 15us shown in Figure 1 [8]. Samsung 840 series SSDs use the 3-core ARM Cortex R4 with 300 MHz in the MDX controller and 400 MHz in the MEX controller [24] which implies that the SHA-1 latency will be more than 50us for a 4KB page.

Figure 1 shows that the cryptographic hash computing latency is actually higher than or comparable to the write latency of the modern flash devices, which indicates that the hash computing process can potentially offset the benefit of write traffic reduction brought by data deduplication to some extent. Even with the fastest secure hashing algorithms, such as Blake2 [3], the hash computing latency is still about half the write latency. Moreover, for non-redundant data chunks, the hash computing process will significantly increase the write latency since it is on the critical I/O path. In deduplication-based storage systems, all incoming data chunks must be calculated to generate their corresponding cryptographic hash values as fingerprints. Thus, the cryptographic hash computing latency becomes an integral part of the overall write latency. When the hash computing throughput is comparable to the write throughput, the deduplication-induced performance overhead will become a serious performance bottleneck. Furthermore, Figure 1 indicates that the read latency is noticeably lower than the write latency. It offers an opportunity for optimizing the write performance by leveraging the high read performance characteristics of flash-based storage systems.

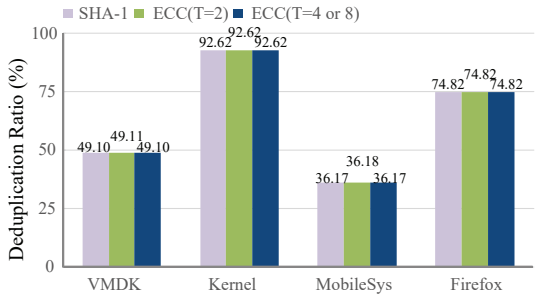


Fig. 2. Comparisons of the deduplication ratios when using SHA-1 and BCH-based ECC with T bits correction capability, driven by four real datasets.

B. An ECC primer

Because of the inherently unreliable nature of NAND memory, some of the stored data in NAND flash may differ in value from the original one due to individual bit errors. Flash controllers usually utilize the Error Correcting Code (ECC) to accomplish the required dependability and reliability. When data is written to NAND flash, an ECC of the data is generated by the ECC engine and stored together with the data, normally in the Out Of Band (OOB) region of each page. When the data is read back, the ECC of the data is recomputed and compared against the one already stored on flash for error detection and correction. In SLC (Single Level Cell), MLC (Multi Level Cell), and TLC (Triple Level Cell) flash devices, Bose, Chaudhuri and Hocquenghem (BCH) codes are regularly used for multi-bit error correction. Recently, Low Density Parity Check (LDPC) codes are increasingly being used in TLC and QLC (Quad Level Cell) NAND flash devices [19].

However, different data chunks may generate the same BCH value or LDPC value, a problem referred to as *hash collision*. To assess the extent of hash collisions of BCH-based ECC, we examine the data redundancy based on the hash functions of SHA-1 and BCH-based ECC respectively and compare their deduplication ratios (i.e., percentage of original data detected as duplicates), experimentally driven by real datasets. Here we assume BCH (4224, 4096, 8) with an 8-bits correction capability is used where 4096 bits (512 bytes) data need 104 bits (13 bytes) ECC [19]. That is, 104 bytes ECC is generated and stored for a 4KB data page within flash device. We use four datasets of real applications that represent different hash functions with a data chunk size of 4KB. Experimental results shown in Figure 2 help us draw the following key observations.

First, the deduplication ratios of BCH-based ECC, when used as a hash function, are higher than those of SHA-1. Given that SHA-1 is sufficiently secure with negligibly low hash collision probability, this suggests that with BCH-based ECC there is a measurable amount of different data chunks that generate the same hash values and thus result in non-negligible false positive duplicate detections (hence the higher deduplication ratios). As a result, ECC cannot be directly used to replace the SHA-based hash algorithms in deduplication storage systems because false positive duplicate detection can lead to unrecoverable data corruption. Second, the hash collision rate is very small as implied by the very small difference between the deduplication ratios in the two cases. In fact, for the four data sets we evaluated, when BCH have 2-bits correction capability (32 bytes ECC per 4KB), the difference in deduplication ratio between BCH-based ECC and SHA-1 is less than 0.01%, suggesting a hash-collision rate of less than 0.01% relative to SHA-1. And there are no measurable hash collisions when BCH has 4-bits (56 bytes ECC per 4KB) and 8-bits (104 bytes ECC per 4KB) correction capability. In other words, while ECC may not be used as a hash function for duplicate detection, it can be used for the detection of data similarity in the data deduplication process provided that data integrity is ensured in a rigorous manner.

In conclusion, from our preliminary evaluations and analysis, we find that the ECC values already available in each flash page can be effectively used for similar data identification. Based on the above observations and to address the challenge, this paper proposes an ECC assisted Deduplication (short for EaD) to construct a collision-free and high-performance deduplication approach with low memory overhead for modern high-performance flash-based devices. EaD exploits the ECC property and leverages the asymmetric read-write performance characteristics to improve the write efficiency. By reducing the write traffic and latency, the read/write interference is also alleviated which results in an improved read performance in deduplication-based flash storage.

III. EaD Design

A. Overview of EaD

EaD locates inside and works with the Flash Translation Layer (FTL) in flash-based devices. The design objectives of EaD are improving both performance and memory efficiency while providing collision free deduplication guarantee for deduplication-based flash storage. EaD does not guarantee that all deduplicate write data can be eliminated but makes sure that write data are safely and correctly stored in flash. Moreover, EaD merely intercepts existing ECC values for data-similarity identification on the write path and is independent of the recovery capability and workflow of ECC on the read path.

Figure 3 shows an overview of the system architecture of EaD. Different from the traditional deduplication workflow, EaD is new and unique in that there are no hash computing procedures conducted on the full data chunks. Considering the high memory overhead of using the ECC code as the fingerprints directly, EaD piggybacks on the ECC information automatically generated by the ECC engine within FTL as the fingerprint for each data chunk. The size of the data chunk is fixed and usually determined by the default page size of the flash. Note that the size of the data chunk can be adjusted by combining the ECC values of a group of data pages to generate the Blake2 fingerprints.

EaD consists of two main functional modules: ECC-based Redundancy Detection and Deduplication Engine, as shown in Figure 3. *ECC-based Redundancy Detection* is responsible for detecting possible redundant data chunks by checking with an ECC-based Bloom filter and comparing the Blake2

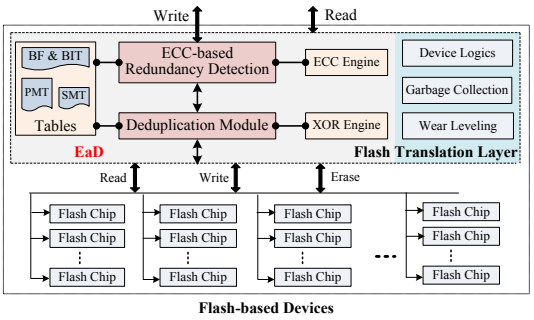
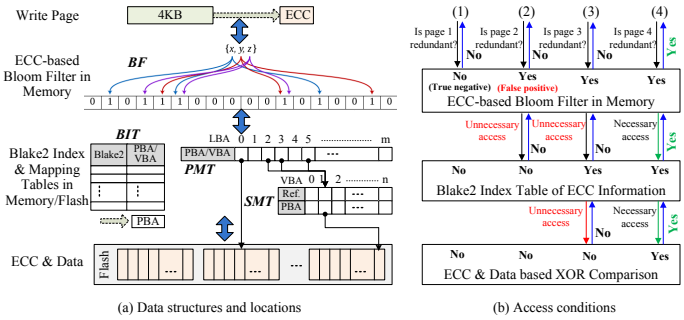


Fig. 3. The system architectural overview of EaD.



(a) Data structures and locations (b) Access conditions

Fig. 4. The data structures and access conditions within EaD. PBA and VBA stand for Physical Block Address and Virtual Block Address, respectively.

fingerprint of ECC values associated with data chunks. Based on the results, the non-deduplicate data chunks can be definitively filtered out. *Deduplication Engine* is responsible for definitively verifying whether a data chunk with a matched Blake2 fingerprint from the ECC-based redundancy detection is redundant or unique by fetching the data and their ECC to perform a byte-by-byte comparison. Leveraging these two modules and four data structures elaborated next, EaD can eliminate most duplicate data chunks with minimal compute and memory overhead.

B. Data Structures

EaD relies on four key data structures to identify data similarity and eliminate the redundant data chunks, namely, ECC-based Bloom Filter (BF), Blake2 Index Table (BIT), Primary Mapping Table (PMT) and Secondary Mapping Table (SMT), as shown in Figure 4(a).

The ECC-based Bloom filter is constructed and stored in main memory to check whether the Balke2 value of a data chunk's ECC exists in the Blake2 Index Table. Initially, the ECC-based Bloom filter is a bit array consisting of m bits that are set to "0". Each element e within the ECC Set E uses k hashing functions h_1, h_2, ..., h_k and each hashing function $h_i(e)$ returns an address value in the bit array that ranges from *0* to *m-1*. Subsequently, the addressed bit is set to "1". Upon inserting a new element (i.e., a new ECC value) into the Bloom filter, if none of the bits addressed by the returned values from the k hashing functions on the ECC value is "0", the ECC value is considered to have already existed in the ECC-based Bloom filter that represents the membership of Blake2 values of ECC information in the Blake2 Index Table. Otherwise, the ECC value is considered to be a new one and the ECC-based Bloom filter will be updated accordantly (by setting all addressed "0" bits to "1").

Blake2 Index Table (BIT) is an in-memory hash structure to store the fingerprints of ECC codes in EaD. Each entry is a key-value pair, *fingerprint*, *Address*. The indexed *fingerprint* is generated with Blake2 [3] algorithm on ECC value and its length can be configured, 8 bytes by default in EaD. Blake2 is a cryptographic hash function faster than MD5 and SHA-based hashes and is at least as secure as the latest standard SHA-3. It has been adopted by many projects and can produce digests of any size between 1 and 64 bytes at a speed of 3.08 cycles per byte [3]. Due to the much shorter length of ECC codes (tens to hundreds of bytes) than the length of data chunks (4KB or above), the hash generation latency on ECC codes is much shorter than that on data chunks. The 32-bit *Address* indicates where we can read the data, either the PBA of a physical flash page or the VBA of a Secondary Mapping Table entry.

Besides the above two data structures to detect possible redundant data chunks, EaD uses a two-level indirect mapping mechanism consists of Primary Mapping Table (PMT) and Secondary Mapping Table (SMT) in deduplication-based flash devices [6]. PMT maps an LBA (Logical Block Address) to either a PBA address or a *Virtual Block Address* in SMT which is differentiated by the highest bit in the 32-bit page address. Each entry in SMT is indexed by the *VBA* and has two variables, *PBA* and *reference*. The 32-bit *PBA* indicates the physical flash page and the 32-bit *reference* records the exact reference count, i.e., the number of different of logical pages mapped to this physical flash page. The relationship between entries in PMT and entries in SMT is essentially N-to-1 mapping. By using a two-level indirect mapping mechanism, the reverse update issue during garbage collection is simplified to only update the corresponding entry in SMT [6]. By doing so, all the logical pages linked to this physical flash page are updated automatically without exhaustively searching for all the referencing LBAs in PMT. Moreover, SSD with EaD can easily switch to a conventional FTL by mapping LBAs to PBAs directly in PMT, which shows the EaD's flexibility.

C. ECC-based Redundancy Detection

ECC-based redundancy detection is the core mechanism in EaD that is different from the previous SHA-based or Sampling-based deduplication approaches. It relies on ECC-based Bloom filter and Blake2 Index Table to filter out the unique data chunks.

Figure 4(b) shows the access conditions among ECC-based Bloom filter, Blake2 Index Table, ECC and data store in flash. If the ECC-based Bloom filter returns "No", meaning that the ECC value does not hit the ECC-based Bloom filter, then the ECC value is definitely not in the current Bloom filter as there are no *false negatives* in Bloom filters [4]. It further indicates that the data chunk associated with the ECC value is unique. For such unique data chunks, no extra read requests will be

issued in EaD. This access condition is labeled (1) in Figure 4. On the countrary, if the ECC-based bloom filter returns "Yes", a query into the Blake2 Index Table is performed to check whether the Balke2 of ECC value is in the Blake2 Index Table because there are possible false positives in Bloom filters.

There are three access conditions, labeled (2), (3) and (4) in Figure 4. First, for (2), if the Blake2 Index Table returns "No", meaning that the Blake2 of ECC value does not hit the Blake2 Index Table which indicates that the corresponding ECC is unique. This further confirms that the data chunk is definitely unique since there are no *false negatives* in ECC. Second, for (3) and (4), if the Blake2 Index Table returns "Yes", it indicates that the Blake2 of ECC value indeed exists in the Blake2 Index Table. However, it still cannot definitively confirm that the data chunk is redundant because of ECC's non-negligible hash collisions. In order to provide 100% certainty, the Deduplication Engine will initiate a read process to fetch data chunks and ECC values, as elaborated in Section III-D.

It must be noted that the unnecessary read requests, as a result of verifying similar chunks as non-duplicate ones, are extremely rare in Figure 4. First, for (2), due to the low probability of Bloom-filter's false positives, the performance overhead induced by the unnecessary accesses is acceptable since these accesses are performed in memory. Second, for (3), due to the almost 0 hash collision rate of ECC as shown in Figure 2, the probability of these unnecessary accesses to memory/flash will be negligibly low. Our evaluation results also confirm and validate these low probabilities.

D. Deduplication Engine

The main objective of the Deduplication Engine is to confirm and eliminate the redundant data chunks. Since ECC has non-zero collision probability as SHA-based algorithms, different data chunks can have the same ECC value, which, however small the probability may be, must not be ignored to avoid unrecoverable data corruption. To address this problem, EaD performs byte-by-byte comparisons between the incoming write data chunks and previously stored data chunks that have been filtered by the ECC-based Redundancy Detection, triggering read operations to fetch the previously stored data chunks from flash. However, these read operations only occur for ascertaining data redundancy for chunks already identified by the ECC-based Redundancy Detection module as being highly likely to be redundant ones. Moreover, based on the performance characteristics of modern flash devices with superior read performance and SHA-based hash computing operations with very high compute overheads, trading off some extra read operations for write traffic reduction and reliability enhancement is not only feasible but also arguably highly desirable.

Figure 5 shows an illustration of the process of handling a write request in EaD. Upon receiving a data chunk, its ECC value is generated and checked in both the ECC-based Bloom filter and the Blake2 Index Table. If it hits the ECC-based Bloom filter, its Blake2 value will be generated and

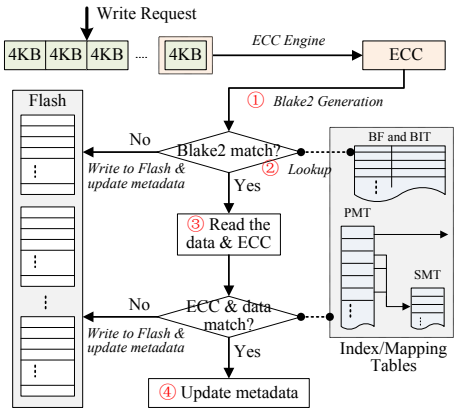


Fig. 5. An illustration of the process of handling a write request in EaD.

checked in the Blake2 Index Table. Only the data chunk whose ECC's Blake2 value exists in the Blake2 Index Table is further processed by the Deduplication Engine. In this case, EaD fetches the previously stored data chunk and its ECC value from the flash and performs a byte-by-byte comparison between these ECC and data chunks in main memory. If both the ECC and data chunks are matched, the incoming data chunk is redundant and, other than only updating the corresponding metadata, its data need not be stored. Otherwise, the incoming data chunk is unique. For the data chunks only with ECC matched but content are not matched, EaD directly writes them to the flash without updating the ECC-based Bloom filter and the Blake2 Index Table.

Although not updating the overlapped ECC values of these unique data chunks will likely decrease the deduplication ratio, our experimental results show that the decreased deduplication ratio is minimal. On the other hand, by only keeping a unique Blake2-Address key-value pair within the Blake2 Index Table, EaD significantly reduces the memory overhead of storing the fingerprints, which is analogous to the fingerprint table in the traditional deduplication-based storage systems. Moreover, EaD also reduces the number of read requests when these overlapped ECC values hit the Blake2 Index Table. In summary, by sacrificing a tiny amount of deduplication ratio, EaD can achieve high deduplication performance and low memory overhead. Section IV also validates this by providing the detailed experimental results.

IV. Performance Evaluation

A. Evaluation setup and methodology

We implemented EaD in SSDSim [16] and the configuration of SSDSim is summarized in Table II. We compare the performance of EaD with that of the SHA-1-based approach (SHA-1), Blake2-based deduplication approach with 16 bytes Blake2 code (Blake2), and a 4-byte/4KB sampling-based deduplication method (Sampling) proposed in CA-FTL [6]. For fairness, we add the Bloom filter to all deduplication approaches. Since the objectives of EaD are to avoid the performance bottleneck of the cryptographic hash computing and improve the deduplication efficiency, we measure the

TABLE I
THE CHARACTERISTICS OF FOUR TRACES.

Trace	*Read request*	*Write request*	*Dedup. Ratio*
Homes	150156	10380822	33.3%
Webs	3116456	11177701	47.3%
Mails	1948414	20762862	91.0%
Hadoop	5596819	3844964	20.7%

TABLE II
THE EXPERIMENTAL PLATFORM.

Parameters	**Value**	**Parameters**	**Latency (us)**
Page Size	4KB	Page Read	3
Pages per Block	64	Page Wrte	100
Blocks per Plane	4096	SHA-1 hashing (4KB)	14.3
Planes per Die	2	Blake2 hashing (4KB)	9.8
Dies per Chip	2	Blake2 hashing (104B)	1
Chips per Channel	2	XOR	1
Channel number	18	Hardware BCH [18]	1

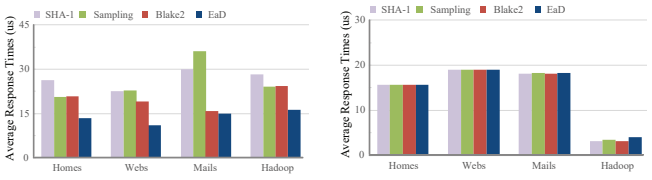
deduplication ratio, memory consumption, the read and write response times to evaluate the efficacy of EaD.

We use three FIU traces [10] and Hadoop trace [6] to evaluate different deduplication methods. The characteristics of these traces are summarized in Table I. However, FIU traces and Hadoop traces only contain MD5- and SHA-1-based fingerprints [10], [6], but not the BCH-ECC fingerprint. To obtain the latter, we cut an existing fingerprint into 8 pieces, replicate each 1/8 hash fingerprint to 512 Bytes and combine them together to reconstruct the original 4KB data. We calculate the BCH (4224, 4096, 8) per 512 Byte as the ECC value in the EaD approach, and we use the first 4 bytes of the reconstructed data as the sample in the sampling-based approach.

We use the Z-NAND latency to configure the SSDSim simulator, as shown in Table II. For the Samsung Z-NAND technology, the read latency is 3us, which is nearly 20 times faster than conventional NAND [7]. Our preliminary result shows that hashing a 4KB page is 20 times slower than hashing a 104B block, considering that Blake2 is faster than SHA-1, which implies that the latency of Blake2 is less than 1us for a 104B BCH code. It must be noted that the latency of SHA-based hashing is configured to be the lowest based on Figure 1, which implies that the improvements of EaD shown in the rest of this section are the lower bound, the real improvements achieved by EaD will likely be much more significant.

B. Performance evaluation

Response Time: Figure 6(a) shows a comparison of average write response times among traditional SHA-1-based, sampling-based, Blake2-based deduplication approaches, and the EaD scheme, driven by the four traces, indicating that EaD has the lowest average response time among the four approaches. Our experimental results show that EaD outperforms the SHA-1-based, sampling-based, and Blake2-based deduplication approaches by an average of 1.92×, 1.86×, and 1.42×, respectively. The reason is that EaD only incurs cryptographic hash computing latency for BCH code, and the hash computing latency of 104B BCH code is far less than that of 4KB data. Actually, in an EaD-based storage system, the average response time is determined by the write latency when



(a) Write (b) Read

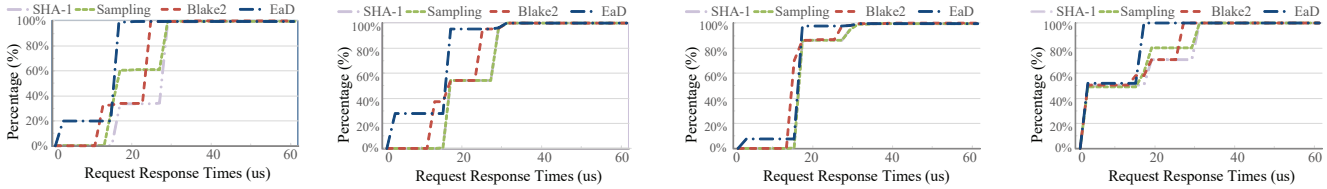
Fig. 6. A comparison of average response times between traditional deduplication approaches and EaD driven by the four traces.

the data chunk is determined to be unique (an unavoidable latency for any deduplication systems), or read latency when determining whether a highly likely redundant data chunk is indeed redundant. Fortunately, for modern flash-based storage devices, the read/write latencies tend to become increasingly shorter than the cryptographic hash computing latency. On the other hand, compared with the SHA-1-based deduplication approaches, the Blake2-based deduplication approach has a shorter cryptographic hash computing latency.

Figure 6(b) shows a comparison in average read response time. Because the original read requests have higher priority, EaD does not increase the read latency notably. The difference of read response time between EaD and SHA-1 is less than 1.7%.

Tail Latency: In modern large-scale storage systems, such as Google, Microsoft Bing, Facebook and Amazon, the long tails of the service latency are of particular concerns [9]. With the wide deployment of flash-based storage devices in large-scale storage systems, the tail latency of flash-based devices ought to be a very important consideration for the design of storage systems [29]. One of EaD's objectives is to avoid the cryptographic hash computing latency bottleneck in traditional SHA-1-based and sampling-based deduplication storage systems, which should have a direct impact on the tail latency.

To estimate this impact of EaD, we evaluate and analyze the response time distributions for the different deduplication approaches on SSDSim driven by the four traces, as shown in Figure 7. The results illustrate that 90% of requests can be completed with a much shorter latency by EaD than any of the other three approaches. As shown in Figure 7, for the EaD-based deduplication system, 90% of requests are completed within 17us, 16us, 16us, and 15us under the Homes, Webs, Mails, and Hadoop traces, respectively. However, for the SHA-1/Sampling/Blake2-based deduplication systems, their corresponding 90-percentile latencies are 28us/26us/24us, 27us/27us/24us, 28us/28us/25us, and 28us/30us/24us, respectively. The reason is that in the SHA-1/Sampling/Blake2-based deduplication systems, the cryptographic hash computing latency occupies a significant portion in the request response time. While data deduplication can reduce the write traffic to the flash-based storage systems and some systems are specifically designed for this purpose [6], [14], the unavoidable hash computing overhead in traditional systems will significantly degrade the system performance of modern flash-based storage systems.



(a) Homes (b) Webs (c) Mails (d) Hadoop

Fig. 7. Response time distributions for the different deduplication approaches driven by the four traces, where the X-axis indicates the request response times while the Y-axis indicates the fraction of requests whose response times are lower than the corresponding values on the X-axis.

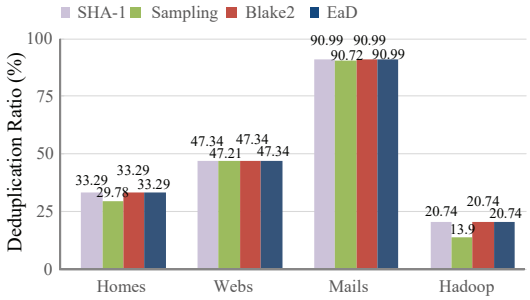


Fig. 8. A comparison of deduplication ratios among different deduplication approaches driven by the four traces.

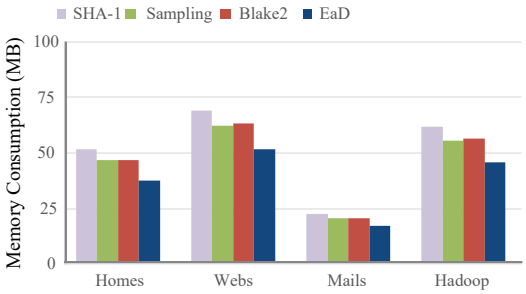


Fig. 9. A comparison among the different deduplication approaches in the memory consumptions under the four traces.

Deduplication Ratio: As described in Section III-D, the deduplication engine of EaD only stores a single data chunk among the different chunks that happen to have the same ECC value (hash collisions) to reduce the memory overhead and the number of read requests. This is a conscious design choice we made in favor of deduplication performance at the expense of possible small deduplication ratio reductions. In other words, in the unlikely event of ECC hash collisions EaD will prevent some redundant data chunks from being detected and eliminated. To quantify this design choice's negative impact on deduplication ratios, Figure 8 shows a comparison in deduplication ratios among the SHA-1, sampling, Blake2, and EaD approaches and indicates that the deduplication ratio of EaD is almost the same as that of SHA-1 and Blake2. The reason is that the hash collision rate of BCH-based ECC is no more than 0.01% higher than that of SHA-1 as shown in Figure 2 in Section II. Moreover, EaD guarantees that the determined redundant data chunks are definitively redundant by byte-by-byte comparisons. In other words, EaD guarantees that there are no false positive detections of redundant data chunks and thus no unrecoverable false data removals, something that none of the SHA-1/Blake2-based deduplication systems can guarantee in theory. However, the sampling-based approach noticeably decreases the deduplication ratio due to its inaccurate detection of redundancy [6]. In summary, EaD provides similar deduplication ratios to those of traditional deduplication approaches, but has 0% false positive redundancy detection and removal and much higher system performance.

Memory Overhead: Memory overhead for storing fingerprints is unavoidable in deduplication systems. Figure 9 shows a comparison in memory consumptions among the different deduplication approaches, including SHA-1-based, Sampling-based, Blake2-based approaches, and the EaD scheme, driven by the four traces. Because directly using the BCH code as the fingerprint consumes significantly more memory space than the traditional deduplication approaches, EaD addresses the challenge by using the Blake2 of the BCH code. The results shown in Figure 9 indicate that EaD consumes notably less memory space than the traditional SHA-1/Blake2- and sampling-based deduplication approaches, by an average of 35.0%, 21.9%, and 23.4%, respectively. The reason is that for the ECC-LBA or SHA-1-LBA/Blake2-LBA mapping table, EaD uses less memory than the traditional deduplication approaches. Given the small memory overhead of EaD, it is more suitable for EaD to store all the information of the mapping table in memory to accelerate the deduplication speed.

V. Related Work

The state-of-the-art approaches to accelerating the compute-intensive data deduplication processes are broadly either software-based optimizations that exploit the parallelism of the deduplication workflow [20], [28] or hardware-assisted optimizations that leverage the data-parallel processing capabilities of the GPU technology [2].

Of the four stages of traditional deduplication identified in Section 1, chunking (either fixed-size FSC or content-defined CDC), fingerprinting, indexing and updating, P-Dedupe [28] further parallelizes the sub-tasks of chunking and fingerprinting, thus achieving higher throughput by effectively exploiting the idle resources of modern computer systems with multi-core or many-core processors. Guo et al. [13] propose an event-driven and multi-threaded client-server interaction model to pipeline FSC-based deduplication systems. Ma et al. [20] propose an adaptive pipelining model to determine the optimal

order of the sub-tasks in the pipeline based on different hardware platforms and data types.

The CDC-based deduplication approaches can improve the deduplication ratio but require extra processing resources. Therefore, GPU-based hardware accelerators for the hash computation of the deduplication-based storage systems have been proposed. For example, Shredder [2] accelerates the popular compute-intensive primitives (*i.e.*, chunking and fingerprinting) in deduplication-based storage systems by exploiting the massively parallel processing power of GPUs. In summary, software-based solutions can be easily implemented in deduplication-based storage systems by pipelining the deduplication tasks or parallelizing the tasks of chunking and fingerprinting, while hardware-based solutions can provide higher throughput but require additional hardware costs.

Recently, primary storage systems are found or expected to have moderate data redundancy, which means that the deduplication technique can also bring significant cost saving for primary storage systems [26]. For example, the CAFTL [6] and CA-SSD [14] schemes utilize the deduplication technique in the internal flash-based SSD to reduce the write traffic to the SSD device. However, due to the limited computing resources within flash-based SSDs, CAFTL and CA-SSD have to design a set of acceleration techniques to reduce the runtime overhead and minimize the performance impact. Kim et al. [17] propose the SHA-1 hardware logic with sampling-based filtering to alleviate the SHA-1 processing overhead. Different from the traditional MD5/SHA-based deduplication approaches, EaD does not need data chunking and hash computing, but leverages the existing ECC function to identify the similar data chunks and optimizes the write performance by leveraging the high read performance characteristics for flash-based storage systems.

VI. Conclusion

This paper proposes an ECC assisted Deduplication approach (called EaD) that avoids cryptographic hash computing altogether by leveraging the existing ECC function embedded in the memory hierarchy to identify the similar data chunks based on their ECC values. EaD reads the identified similar data chunks from flash storage and performs byte-by-byte comparison of the data content to detect and eliminate the redundant data chunks with complete certainty. Experiments conducted on our lightweight prototype implementation of the EaD system show that the EaD approach significantly reduces the I/O latency by an average of 1.92× and 1.86×, and reduces the memory consumption by an average of 35.0% and 21.9%, compared with the existing SHA- and sampling-based deduplication approaches, respectively.

Acknowledgement

This work is supported by the National Natural Science Foundation of China under Grant No. 61972325, No. 61872305, No. U1705261, and No. 61702569, the US National Science Foundation under Grant No. CCF-1704504 and No. CCF-1629625.

References

[1] 3D XPoint: A Breakthrough in Non-Volatile Memory Technology. https://www.intel.com/content/www/us/en/architecture-and-technology/intel-micron-3d-xpoint-webcast.html.

[2] P. Bhatotia, R. Rodrigues, and A. Verma. Shredder: GPU-accelerated Incremental Storage and Computation. In *FAST'12*, Feb. 2012.

[3] BLAKE2–fast secure hashing. https://blake2.net/blake2.pdf.

[4] B. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM*, 13(7):422–426, 1970.

[5] CASL Architecture in Nimble Storage. http://www.nimblestorage.com/products/architecture.php.

[6] F. Chen, T. Luo, and X. Zhang. CAFTL: A Content-Aware Flash Translation Layer Enhancing the Lifespan of Flash Memory based Solid State Drives. In *FAST'11*, pages 77–90, Feb. 2011.

[7] W. Cheong, C. Yoon, S. Woo, and et. al. A Flash Memory Controller for 15us Ultra-Low-Latency SSD Using High-Speed 3D NAND Flash with 3us Read Time. In *ISSCC'18*, Feb. 2018.

[8] Crypto++ Benchmarks. https://www.cryptopp.com/benchmarks.html.

[9] J. Dean and L. Barroso. The Tail at Scale. *Communications of the ACM*, 56(2):74–80, 2013.

[10] FIU traces. http://iotta.snia.org/traces/390.

[11] FlashReduce Data Reduction in Pure Storage. http://www.purestorage.com/flash-array/flashreduce.html.

[12] Google Security Blog: Announcing the first SHA1 collision. https://shattered.io/.

[13] F. Guo and P. Efstathopoulos. Building a High-performance Deduplication System. In *USENIX ATC'11*, Jun. 2011.

[14] A. Gupta, R. Pisolkar, B. Urgaonkar, and A. Sivasubramaniam. Leveraging Value Locality in Optimizing NAND Flash-based SSDs. In *FAST'11*, Feb. 2011.

[15] High performance PCIe SSD Controllers. https://www.marvell.com/storage/ssd/88ss1092-93/.

[16] Y. Hu, H. Jiang, D. Feng, L. Tian, H. Luo, and S. Zhang. Performance impact and interplay of ssd parallelism through advanced commands, allocation strategy and data granularity. In *ICS'11*, Jun. 2011.

[17] J. Kim, C. Lee, S. Lee, I. Son, J. Choi, S. Yoon, H. Lee, S. Kang, Y. Won, and J. Cha. Deduplication in SSDs: Model and Quantitative Analysis. In *MSST'12*, Apr. 2012.

[18] Youngjoo Lee, Hoyoung Yoo, Injae Yoo, and In-Cheol Park. High-throughput and low-complexity bch decoding architecture for solid-state drives. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(5):1183–1187, 2013.

[19] Y. Luo. *Architectural Techniques for Improving NAND Flash Memory Reliability*. PhD thesis, Carnegie Mellon University, Aug. 2018. Doctor of Philosophy.

[20] J. Ma, B. Zhao, G. Wang, and J. Liu. Adaptive Pipeline for Deduplication. In *MSST'12*, Apr. 2012.

[21] S. Maneas, K. Mahdaviani, T. Emami, and B. Schroeder. A Study of SSD Reliability in Large Scale Enterprise Storage Deployments. In *FAST'20*, Feb. 2020.

[22] B. Mao, S. Wu, H. Jiang, X. Chen, and W. Yang. Content-aware Trace Collection and I/O Deduplication for Smartphones. In *MSST'17*, May 2017.

[23] MD5 Collision Demo. http://www.mscs.dal.ca/~selinger/md5collision/.

[24] New Elements to Samsung SSDs: The MEX Controller, Turbo Write and NVMe. https://www.anandtech.com/show/7152/new-elements-to-samsung-ssds-the-mex-controller-turbo-write-and-nvme.

[25] B. Schroeder, R. Lagisetty, and A. Merchant. Flash Reliability in Production: The Expected and the Unexpected. In *FAST'16*, Feb. 2016.

[26] K. Srinivasan, T. Bisson, G. Goodson, and K. Voruganti. iDedup: Latency-aware, Inline Data Deduplication for Primary Storage. In *FAST'12*, Feb. 2012.

[27] The ARM Cortex R5 processor. https://developer.arm.com/ip-products/processors/cortex-r/cortex-r5.

[28] W. Xia, H. Jiang, D. Feng, L. Tian, M. Fu, and Z. Wang. Exploiting Parallelism in Data Deduplication System. In *NAS'12*, Jun. 2012.

[29] S. Yan, H. Li, M. Hao, H. Tong, S. Sundararaman, A. Chien, and H. Gunawi. Tiny-Tail Flash: Near-Perfect Elimination of Garbage Collection Tail Latencies in NAND SSDs. In *FAST'17*, Feb. 2017.

[30] J. Zhang, M. Kwon, D. Gouk, C. Lee, M. Alian, M. Chun, M. Kandemir, N. Kim, J. Kim, and M. Jung. FlashShare: Punching Through Server Storage Stack from Kernel to Firmware for Ultra-Low Latency SSDs. In *OSDI'18*, Oct. 2018.