EaD: ECC-Assisted Deduplication With High Performance and Low Memory Overhead for Ultra-Low Latency Flash Storage

Suzhen Wu, *Member, IEEE*, Chunfeng Du, *Member, IEEE*, Weidong Zhu, *Member, IEEE*, Jindong Zhou, Hong Jiang, *Fellow, IEEE*, Bo Mao, *Member, IEEE*, and Lingfang Zeng, *Member, IEEE*

Abstract—Data deduplication has become a commodity feature in flash storage products to effectively reduce redundant write data and improve space efficiency. However, it also introduces computing and memory overhead to generate and store the cryptographic hash (fingerprint) in face of the moderate data redundancy in primary storage. With the advent of 3D XPoint and Z-NAND technologies, and the stronger cryptographic hash functions in use, such as SHA-256, both the computing and memory overheads are increasingly serious performance bottlenecks for inline data deduplication in these ultra-low latency flash storage. To address these problems, we propose an ECC-assisted Deduplication approach, called EaD, which exploits the ECC property and the asymmetric read-write performance characteristics of modern flash storage. EaD first identifies data similarity by leveraging the device-generated ECC values of data chunks as their fingerprints, significantly reducing the costly MD5/SHA-based 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 increases I/O performance by up to 4.2×, with an average of 2.5×, compared with the existing MD5/SHA- and sampling-based deduplication approaches.

Index Terms—Ultra-low latency flash, ECC-assisted deduplication, I/O deduplication, high-performance

1 Introduction

Due to the slow mechanical nature of hard disk drivers, flash-based devices have been extensively deployed in modern storage systems to satisfy the increasing demand for storage performance. However, the performance and reliability of flash-based devices is highly sensitive to the write traffic [1]. 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 [2], [3]. 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 write data and minimizing the transmission of redundant data in storage systems.

Previous studies [2], [3], [4] have indicated that the ability of data deduplication to reduce write traffic can help 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 from many leading companies, such as HPE Nimble Storage [5] and Pure Storage [6], for the purpose of enhancing the 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.

(1) High Computational & Memory Overheads: The computational intensity of cryptographic hash functions and memory consumption of fingerprints can lead to serious performance degradation of deduplication-enabled flash-based primary storage where the data redundancy is moderate [7]. 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 for fingerprint verification that determines whether incoming chunks are duplicates to be

- *Suzhen Wu is with the School of Informatics of Xiamen University, Xiamen, Fujian 361005, China, and also with the Wuhan National Laboratory for Optoelectronics, Wuhan, Hubei 430079, China. E-mail: suzhen@xmu.edu.cn.*
- *Chunfeng Du, Weidong Zhu, Jindong Zhou, and Bo Mao are with the School of Informatics of Xiamen University, Xiamen, Fujian 361005, China. E-mail: dcf_wy@163.com, zwdong625@qq.com, 24320181153619@stu.xmu.edu.cn, maobo@xmu.edu.cn.*
- *Hong Jiang is with the Computer Science and Engineering Department, University of Texas at Arlington, Arlington, TX 76019 USA. E-mail: hong.jiang@uta.edu.*
- *Lingfang Zeng is with the ZJ Lab-Enflame Joint Innovation Research Center, Zhejiang Lab, Hangzhou 311121, China. E-mail: zenglf@zhejianglab.com.*

Manuscript received 8 June 2021; revised 31 Jan. 2022; accepted 6 Feb. 2022. Date of publication 18 Feb. 2022; date of current version 13 Dec. 2022.
This work was supported in part by the National Natural Science Foundation of China under Grants 61872305, 61972325, and U1705261, in part by U.S. NSF under Grant CCF-1704504 and CCF-1629625 in part by the Open Project Program of Wuhan National Laboratory for Optoelectronics under Grant 2021WNLOKF011, in part by the Zhejiang provincial "Ten Thousand Talents Program" under Grant 2021R52007 and Center-Initiated Research Project of Zhejiang Lab under Grant 2021DA0AM01, and in part by CCF-Alibaba Innovative Research Fund For Young Scholars.
(Corresponding author: Bo Mao.)
Recommended for acceptance by A. Karanth.
Digital Object Identifier no. 10.1109/TC.2022.3152665

removed, and (4) index and metadata updating. Among these four stages, stages (1) with Content-Defined Chunking (CDC) algorithms and (2) both consume significant CPU resources while stages (3) and (4) occupy a great deal of memory space. More specifically, the MD5/SHA-based hash algorithms [8] all need to compute the hash value of chunk content, which significantly lengthens the write latency in deduplication-enabled storage systems, especially in the ultra-low latency flash-based devices with the 3D XPoint [9], Z-NAND [10] and NVMe technologies. The memory occupied by fingerprints also implies less cache space to buffer user I/Os. Moreover, the computational and memory overhead can be a critical issue when the data deduplication is embedded within flash-based SSDs [2], [3] and Smartphones [4] where the computing and memory resources are limited.

(2) Nonzero Hash-collision Probability: 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 [11], [12]. 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 [11]. 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 its lower cryptographic hash speed and longer hash length than SHA-1.

To simultaneously address the above two problems 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 device-generated ECC information within each flash page 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 content comparison. 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 MD5/SHA- and sampling-based deduplication approaches [2], [3] in terms of I/O performance by up to 4.2×, with an average of 2.5×.

In particular, this paper makes the following contributions:

- To the best of our knowledge, EaD is the first deduplication study to leverage the ECC function already widely embedded in the storage hierarchy to identify the similar data chunks. This significantly reduces the costly computing overhead of MD5/SHA-based cryptographic hash functions within deduplication-based systems.
- By using byte-by-byte comparison, EaD provides a collision-free deduplication method to completely eliminate the hash collision problem from the existing MD5/SHA-based deduplication for flash storage.
- The experimental results show that EaD can significantly reduce the processing and memory overhead from the SHA- and sampling-based deduplication approaches.

The rest of this paper is organized as follows. Background and motivation are presented in Section 2. We describe the design details of EaD in Section 3. The performance evaluation is presented in Section 4 and the related work is presented in Section 5. We conclude this paper in Section 6.

2 Background and Motivation

In this section, we first analyze the performance characteristics of the hash calculations and flash, and then motivate our research by providing a primer on ECC within flash and its comparison with the existing hash algorithms.

2.1 A New Performance Landscape

With the rapid development and application of new storage technologies, such as Intel 3D XPoint [9], Samsung's Z-NAND [10] and NVMe, the performance of flash-based devices has been significantly improved [13]. For example, Intel has integrated the 3D XPoint memory into Intel Optane series SSDs and Samsung has also released Z-NAND flash as the core enabling technology for Samsung 983 ZET flagship datacenter SSDs. Moreover, other manufactures are also in progress of releasing ultra-low latency flash products, such as Everspin's nvNITRO Technology and Toshiba's XL-Flash [13].

On the other hand, the cryptographic hash functions used in deduplication-enabled 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 compares the latencies of reading and writing 4KB data pages from and to Intel Optane 3D XPoint [9] and Samsung Z-NAND flash devices [10], 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 bar in the figure represent the lowest values derived from the fastest hash implementations [14], considering that the ARM Cortex R5 processor is used with the maximum clock frequency of 1.4 GHz [15]. The minimal clock frequency is set to 300MHz. In real SSD products, the SHA-based hash computing latency could be much higher [2], [3], [16], [17]. For example, the latest released Marvell NVMe SSD Controllers (88SS1092 and 88SS1093) use triple ARM Cortex R5 up to 500MHz [16], which implies that the SHA-1 latency of a 4KB page is 40us [18]. Samsung 840 series SSDs use the 3-core ARM Cortex R4 with 300 MHz in

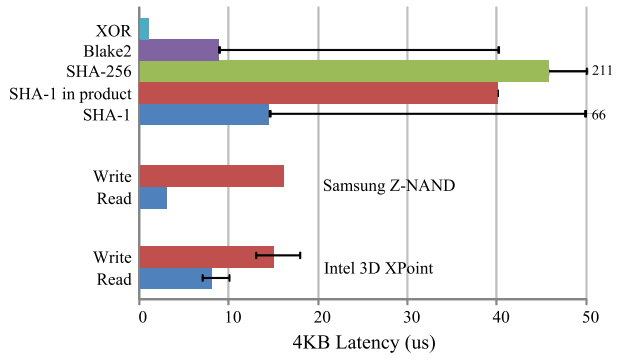


Fig. 1. A comparison of latencies of reading and writing 4KB data pages from Intel Optane 3D XPoint and Samsung Z-NAND flash devices, and performing various computing functions.

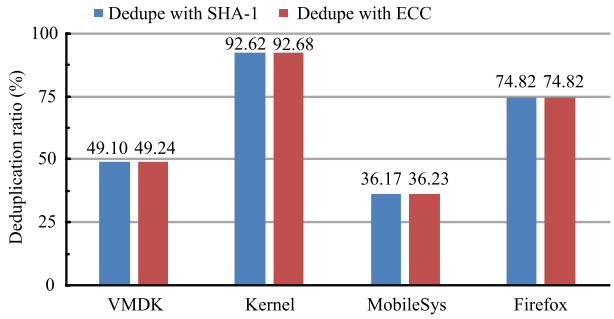


Fig. 2. Comparisons of the deduplication ratios when using SHA-1 and BCH-based ECC respectively as the hash function, driven by four real datasets.

the MDX controller and 400 MHz in the MEX controller [17], which implies that the SHA-1 latency is more than 50us for a 4KB page.

The key takeaway from Fig. 1 is 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 [14], the hash computing latency is still about half of the write latency. Moreover, for non-redundant data chunks, the hash computing process will significantly increase the write latency without any benefit since it is on the critical I/O path. In deduplication-enabled storage systems, all the 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.

In primary storage, the data redundancy is moderate [7], which implies that the deduplication-induced cryptographic hash computing latency is critically significant to the overall system performance. This is because all the unique data blocks, which don't benefit from deduplication but can dominate in low-redundancy data streams in primary storage, will add extra cryptographic hash computing latency during inline data deduplication. Furthermore, Fig. 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 for flash-based storage systems.

2.2 An ECC Primer

Due to the unique and inherently unreliable nature of flash memory, some of the stored data in flash memory 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 flash memory, 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. 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 [19]. Recently, Low Density Parity Check (LDPC) codes are increasingly being used in TLC and QLC (Quad Level Cell) flash devices [20].

It must be noted that different data chunks may generate the same ECC, i.e., BCH value or LDPC value, a problem often referred to as *hash collision* when ECCs are utilized as hash functions. To assess the extent of hash collisions of BCH-based ECC, we experimentally examine the amount of identified by hash functions of SHA-1 and BCH-based ECC respectively and compare their deduplication ratios, driven by real datasets. The difference in deduplication ratios between the two hash functions is a direct indication of their difference in hash collision, i.e., BCH has a slightly higher collision ratio than SHA-a by an extremely small amount. Here we assume the BCH (4224, 4120, 8) with an 8-bit correction capability is used where 4120 bits data (512 bytes valid data plus 24 bits OOB data) need 104 bits (13 bytes) ECC [20]. That is, 104 bytes ECC is generated and stored for a 4KB data page within flash device. In most SSDs, actually much more powerful BCH ECC configurations are used, even with the LDPC method. Thus the hash collision rate with ECC could be much lower than the configuration we choose.

Experimental results shown in Fig. 2 help us draw two key observations. First, the deduplication ratios of BCH-based ECC, when used as a hash function, are slightly 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, the difference in deduplication ratio between BCH-based ECC and SHA-1 is less than 0.2%, suggesting a hash-collision rate of less than 0.2% relative to SHA-1. In other words,

while ECC cannot be used as a hash function for duplicate detection, it may be used for the detection of data similarity in the data deduplication process provided that data integrity is ensured in a rigorous manner.

2.3 Motivation

Data deduplication has been well studied and widely used to reduce the write traffic and improve the space efficiency in flash storage, including embedded systems [2], [3], [4] and enterprise environments [5], [21]. The rapidly increasing performance of flash-based storage due to emerging technologies (see Section 2.1) is shifting the performance bottleneck in deduplication from the storage layer to the compute layer. The reduction in I/O access time is making the performance overhead of hash computing, required for data chunking and fingerprinting, an increasingly large component of the read and write latency and, consequently, a severe performance bottleneck in deduplication-enabled flash storage. This performance bottleneck is especially intrusive and counterproductive for non-redundant data chunks that stand to offer neither write traffic nor space reduction, particularly for ultra-low latency flash devices that use the 3D XPoint, Z-NAND, and NVMe technologies.

In the meantime, the traditional MD5/SHA-1 based cryptographic hash functions have nonzero probabilities of hash-collision, which in theory can cause unique/non-redundant data chunks to be falsely detected and removed as redundant ones and thus lead to unrecoverable data loss [8]. Although a more secure hash algorithm like SHA-256 can reduce the probability of hash collision, it decreases the performance of the storage system and increases the memory overhead significantly. Thus, how to address performance/memory overheads and the hash collision issues associated with data deduplication is of a timely and potentially high impact research challenge.

On the other hand, primary storage usually has moderate data redundancy [7], which implies that the inline deduplication-induced cryptographic hash computing latency is critically significant to the overall system performance because all the unique data blocks will add extra cryptographic hash computing latency. Moreover, from our preliminary evaluations and analysis, we find that the existing device-generated ECC values could be used for similar data identification. Based on the observation and to address the challenge, this paper proposes an ECC-assisted Deduplication (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 or latency, the read/write interference is also alleviated which results in an improved read performance of deduplication-enabled flash storage.

3 EaD Design

In this section, we first present an architectural overview of our proposed EaD method, which is followed by a detailed description of the data structures and the ECC-based redundancy detection to identify similar data chunks. Then we describe the workflow of the deduplication module and the performance optimization with the design of a prefetch cache.

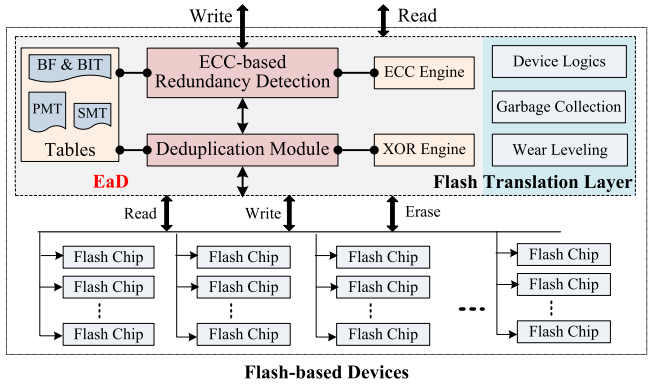


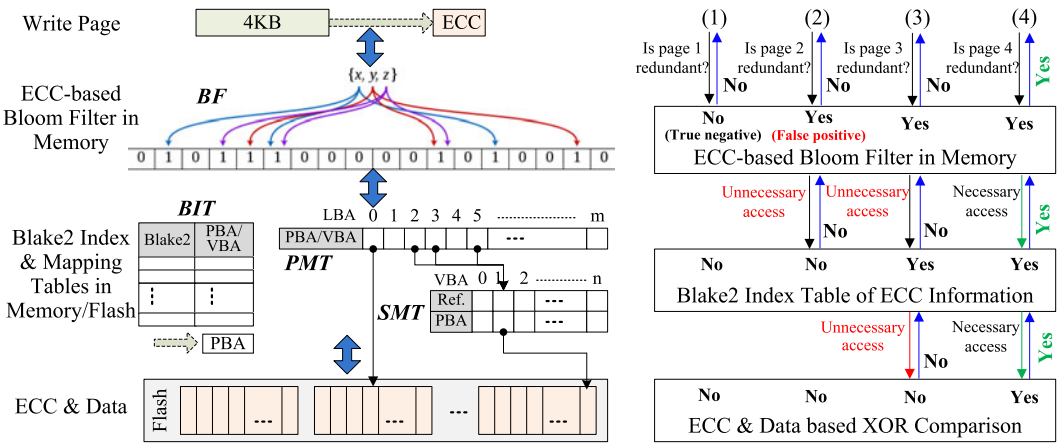
Fig. 3. The system architectural overview of EaD.

3.1 Overview of EaD

The design objectives of EaD are to improve both performance and memory efficiency while providing collision-free deduplication guarantee for deduplication-enabled flash storage. EaD trades a tiny fraction of duplicate data not being removed for vastly improved performance and memory efficiency and, importantly, guarantees that write data are safely and correctly stored in flash. Moreover, EaD merely intercepts existing device-generated ECC values for data-similarity identification on the write path and does not affect the workflow of ECC on the read path and its recovery capability.

Fig. 3 shows an overview of the system architecture of EaD. EaD is located in and works with the Flash Translation Layer (FTL) in flash-based devices. Different from the traditional deduplication workflow, EaD is new and unique in that there are no SHA-based hash computing procedures conducted on the data chunks. Instead, for each data chunk EaD generates a fingerprint in the form of the Blake2 value directly derived from the ECC information that is automatically generated by the ECC engine of FTL. The size of the data chunk is fixed and usually determined by the default page size of the flash. Note that by combining the ECC values of a group of data pages to generate the Blake2 fingerprints, the data chunk size is adjustable.

Fig. 3 shows that EaD consists of two main functional modules: the ECC-based Redundancy Detection module and Deduplication module. *ECC-based Redundancy Detection* is responsible for detecting possible redundant data chunks by checking with an ECC-based Bloom filter and comparing the Blake2 fingerprints of ECC values associated with data chunks. Based on the results, the non-deduplicate data chunks can be definitively filtered out because there are no false negatives for ECC-based hash functions. *Deduplication* 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 ECC to perform a byte-by-byte comparison. Based on the two modules and four data structures, EaD can eliminate the most duplicate data chunks with minimal computational and memory space overhead.



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

Fig. 4. The data structures and access conditions within EaD.

3.2 Data Structures

Four main data structures are used in EaD 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 Fig. 4.

The BF is build and stored in main memory to check whether the Blake2 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 hash functions h_1, h_2, ..., h_k and each hash function $h_i(e)$ returns an address value pointing to one of the m bit positions of the bit array, *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 k returned values of the k hash 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 the 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 by the k hash functions to "1").

The 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 by the Blake2 [14] algorithm on the corresponding ECC value and its length can be configured, 10 bytes by default in EaD. Blake2 is a faster cryptographic hash function than MD5 and SHA-based hash functions 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 [14]. Due to the much shorter length of ECC codes (tens to hundreds of bytes) than that of data chunks (4KB or more), the Blake2 generation latency on ECC codes is much shorter than the SHA-based hash generation latency. The 32-bit *Address* indicates where the data can be read, either the PBA (Physical Block Address) of a physical flash page or the VBA (Virtual Block Address) of a Secondary Mapping Table (SMT) entry.

Moreover, EaD uses a two-level indirect mapping mechanism consisting of PMT and SMT in deduplication-enabled flash devices [2]. The PMT maps a LBA (Logical Block Address) to either a PBA or a VBA 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*, *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 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 problem of reverse update during garbage collection is simplified to only update the corresponding entry in SMT [2]. 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, it makes EaD flexible because SSDs with EaD can easily switch to a conventional FTL by mapping LBAs to PBAs directly in PMT.

3.3 ECC-Based Redundancy Detection

The core mechanism that makes EaD different from the existing SHA-based or Sampling-based deduplication approaches is the ECC-based redundancy detection. It further relies on ECC-based Bloom filter and Blake2 Index Table to filter out the unique data chunks. Fig. 4 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 [22]. 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 Fig. 4. On the contrary, if the ECC-based bloom filter returns "Yes", a query into the Blake2 Index Table is performed to check whether the Blake2 of ECC value is in the Blake2 Index Table because there are possible false positives in Bloom filters.

Fig. 4 shows three access conditions in EaD, labeled (2), (3) and (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 and thus the corresponding ECC is unique. This further confirms that the data chunk is definitely unique since there are no *false negatives* in ECC.

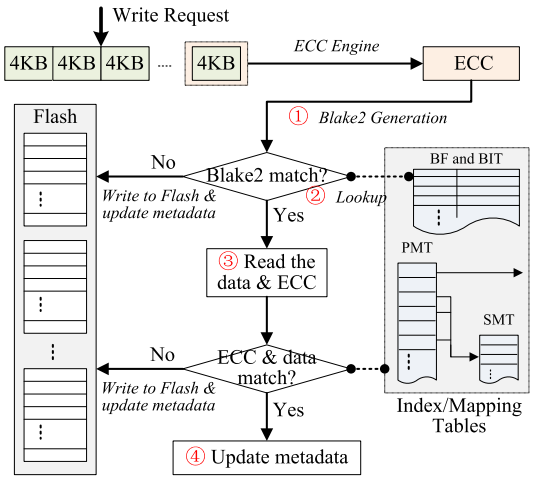


Fig. 5. An illustration of the process of handling a write request in EaD.

Second, for (3) and (4), if the Blake2 Index Table returns "Yes", it indicates that the Blake2 of ECC indeed exists in the Blake2 Index Table. However, it still cannot definitively confirm that the data chunk is redundant because of ECC's non-zero hash collisions. In order to provide 100% certainty, the Deduplication module will initiate a read process to fetch data chunks and ECC values for byte-by-byte comparison, as illustrated in Section 3.4.

The number of unnecessary read requests is extremely low, as that shown in Fig. 4. The reasons are twofold. First, for (2), due to the low probability of Bloom-filter's false positives, the performance overhead induced by the unnecessary accesses is acceptable because they are performed in memory. Moreover, the probability of Bloom-filter's false positives can be further reduced by some optimizations [23]. Second, for (3), due to the non-zero hash collision rate of ECC as suggested by Fig. 2, these unnecessary accesses to memory/flash are negligible. This is confirmed by our evaluation results (Section 4).

3.4 Deduplication Module

The Deduplication module is designed to confirm and eliminate the redundant data chunks. ECC-based checking mechanism has nonzero collision probability, thus the probability of different data chunks having the same ECC value is nonzero. 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 module, 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 arguably highly desirable.

The workflow of handling a write request in EaD is shown in Fig. 5. When a write data chunk arrives, its ECC value is generated and checked in both the ECC-based Bloom filter and the Blake2 Index Table. If it hits in the ECC-based Bloom filter, then its Blake2 value will be generated and 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 module. 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 match, 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 with matched ECC but unmatched content, EaD directly writes them to the flash without updating the Blake2 Index Table and the ECC-based Bloom filter.

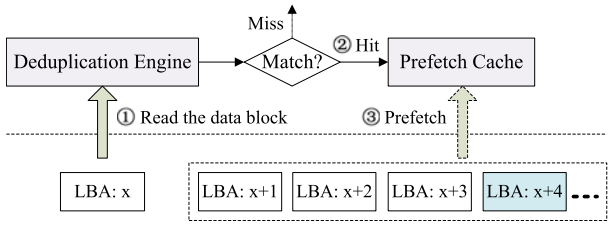


Fig. 6. The workflow of the prefetch cache optimization.

EaD does not update the overlapped ECC values which may decrease the deduplication ratio. However, 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 in 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-enabled 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 very small deduplication ratio, EaD can achieve high deduplication performance and low memory overhead. It is also validated by the detailed experimental results in Section 4.

3.5 Prefetch Cache

To provide 100% duplicate detection accuracy by byte-by-byte comparisons, EaD performs extra read operations for fetching the previously stored data chunks. Although EaD eliminates the overhead of hash computing on the critical write I/O path, the increased read operations must be carefully considered to mitigate the negative impact on system performance. The Prefetch Cache module is designed to reduce the read overhead in EaD by exploiting the content locality of workloads [24].

Fig. 6 illustrates the workflow of the Prefetch Cache module. EaD reads a previously stored data chunk from the storage device and compares it with the incoming data chunk to check whether the latter is redundant. Because of the content locality of the data streams, the subsequent incoming data chunks have higher probability of being redundant. Upon a match of the content of the two comparing data chunks, a new read request will be initiated to fetch the adjacent data chunks from the storage device, potentially during idle periods. By prefetching these data chunks, the subsequent byte-by-byte comparisons can directly be

TABLE 1
The Evaluation Platform

Component	Description
CPU	Intel Xeon E5-2407 2.20GHz
Memory	16GB DDR SDRAM
HDD	Seagate ST9750420AS 7200RPM 750GB
SSD	Intel 750 Series PCIe-based 400GB NVMe SSD
OS	Ubuntu server 14.04 with Linux 4.2.0

TABLE 2
The Characteristics of the Four Datasets [27] Used in the Real Platform

Applications	*Size (GB)*	*Dedup. Ratio*	*Comments*
VMDK	332	49.1%	VM disk image files
Kernel	190	92.6%	Linux kernel source code
MobileSys	256	36.2%	System files for Smartphones
Firefox	287	74.8%	Firefox installation files

performed in memory without any further read operations to the storage devices.

In our current implementation, EaD uses the asynchronous prefetch method to exploit the content locality. When an incoming data chunk is content matched with the fetched data chunk x, EaD initiates a new read request to prefetch n contiguous data chunks besides those already in the cache. In each set of n prefetched data chunks, a trigger data chunk is identified at a trigger distance of m from the end of the prefetched set of data chunks. When $m = 0$, the trigger data chunk is set on the last data chunk of the prefetched set. When a trigger data chunk is hit, a new asynchronous prefetch process is initiated for the next set of n contiguous data chunks. For example, data chunk with LBA = x+4 is set as the trigger data chunk in Fig. 6. When it is matched with the incoming data chunk, a new asynchronous prefetch request will be issued. Thus, asynchronous prefetch ensures that EaD always stays ahead of sequential redundant data chunks and never incurs a read miss after the initial miss for a sequential redundant data stream. Our experimental results also show how the prefetch length affects the system efficiency. For simplicity, EaD utilizes LRU as the cache replacement algorithm to manage the prefetched data chunks.

4 Performance Evaluation

In this section, we first describe the evaluation setup and methodology. Then we evaluate the EaD performance through extensive dataset-driven real platform evaluations and trace-driven simulator experiments.

4.1 Evaluation Setup and Methodology

The EaD prototype is implemented and evaluated on both a real platform and an SSD simulator. In the real platform, we have implemented an EaD prototype in the Linux operating system by adding an extra ECC functionality in the host. We also implemented EaD in the SSDSim [25] simulator where the ECC engine is available at the FTL level. We compare the performance of the EaD prototype with that of the traditional MD5/SHA-based deduplication approaches using different hash algorithms, such as MD5, SHA-1 and SHA-256, and a 4-byte/4KB sampling-based deduplication method proposed in CA-FTL [2]. All the experiments were conducted on a Dell PowerEdge T320 node with an Intel Xeon quad-core processor (2.20GHz) and 16GB memory. In this system, a Seagate ST9750420AS 7200RPM 750GB is used to host the operating system (Ubuntu server 14.04 with Linux 4.2.0) and other software. All the datasets are evaluated on an Intel 750 Series 400GB SSD [26], which is the first NVMe SSD released to the market with a PCIe NVMe 3.0 x4 Interface. For fair comparison, all of the approaches in our experiment adopt the same architecture for running the baseline experiments, such as the Bloom filter and mapping table. The platform setup is summarized in Table 1.

In this paper, we use four datasets generated from real applications that represent different data redundancy characteristics with a data chunk size of 4KB, as summarized in Table 2. We also use the three FIU traces and one Hadoop trace to evaluate different deduplication systems [2], [28] in the simulator-based evaluation, as summarized in Table 3. These traces only contain MD5/SHA-1-based fingerprints [2], [28], we use the first 4-bytes of MD5/SHA-1-based fingerprint as the sample in the sampling-based approach, and the ECC value generated from the MD5/SHA-1-based fingerprint as the ECC in the EaD approach. Since the design objectives of EaD are to avoid the performance bottleneck of the cryptographic hash computing and improve the deduplication efficiency, we measure the write throughput and response time to evaluate the efficacy of EaD.

4.2 Dataset-Driven Real-Platform Evaluations

System Throughput. We first conduct experiments to measure the throughput of MD5/SHA-, sampling-based deduplication method and EaD, driven by the four datasets. Fig. 7 compares EaD against the four traditional approaches in terms of throughput, normalized to that of the MD5-based deduplication approach. First, EaD is shown to achieve the best system throughput among all the approaches and is 2.3× better than the MD5-based deduplication approach on average. For traditional MD5/SHA-based deduplication approaches, cryptographic hash computing is applied to all incoming data chunks to generate fingerprints, no matter they are redundant or unique. While the sampling-based deduplication approach avoids part of the cryptographic hash computing for the non-redundant data chunks, the duplicate-detection accuracy is low and cryptographic hash computing is still needed for the redundant data chunks. In contrast, EaD effectively utilizes the readily available, free ECC information to filter out the unique data chunks, thus avoiding the cryptographic hash computing overhead. Only the data chunks with high probability of being redundant will incur extra read operations, in lieu of hash computing. This has additional benefits because the read throughput is much

TABLE 3
The Characteristics of the Four Traces Used in the Simulator

Trace	*Read requests*	*Write requests*	*Dedup. Ratio*
Homes	150156	10380822	33.3%
Web-vm	3116456	11177701	47.3%
Mail	1948414	20762862	91.0%
Hadoop	5596819	3844964	20.7%

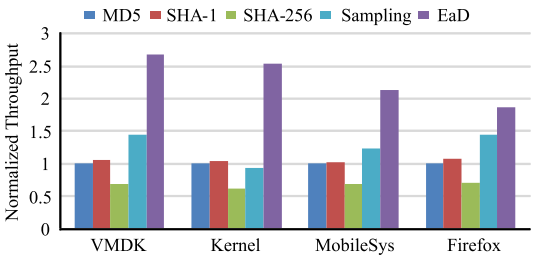


Fig. 7. Throughput results of different approaches driven by the four datasets, normalized to that of the MD5-based deduplication approach.

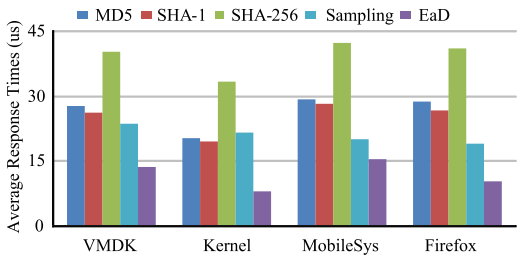


Fig. 8. Average response time results of different approaches driven by the four datasets.

higher than the cryptographic hash computing throughput on modern flash-based storage devices, as shown in Fig. 1 in Section 2.1. Therefore, EaD achieves significant improvement on system throughput over the MD5/SHA-based and sampling-based deduplication approaches.

Second, the results indicate that the specific hash computing algorithm used has a significant impact on system throughput for traditional deduplication-enabled storage systems. MD5- and SHA-1-based deduplication approaches have similar system throughput while SHA-256-based deduplication approach is outperformed by MD5-based approach by an average of 67.1% on Intel 750 Series PCIe-based 400GB SSD. The reason is that the I/O performance of modern flash-based storage devices has been improved significantly, relative to the improvement on CPU performance. This results in the cryptographic hash computing throughput dominating and adversely affecting the throughput of traditional MD5/SHA-based deduplication systems. SHA-256's computing throughput is the lowest among all the three hash algorithms, and thus so is its system throughput. This reinforces the importance of the hash-related computing overhead in the design considerations of deduplication-enabled flash storage systems.

Response Time. Fig. 8 shows comparisons in average response time among traditional MD5/SHA-, sampling-based deduplication approaches and EaD, driven by the four datasets, indicating that EaD has the lowest average response time among the five approaches. Our experimental results show that EaD outperforms the MD5-, SHA-1-, SHA-256-, and sampling-based deduplication approaches by 2.8×, 2.6×, 4.2× and 2.7×, respectively, with an average of 2.5×. The reason is that EaD does not incur any cryptographic hash computing latency on the critical write path. In an EaD-based storage system, the average response time is determined by the write latency when the data chunk is determined to be unique (an unavoidable latency for any deduplication system), 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. Meanwhile, some of the read accesses to the flash storage for the byte-by-byte comparisons in EaD can be eliminated by the Prefetch Cache to further reduce EaD's read latency. Compared with the MD5- and SHA-1-based deduplication approaches, the SHA-256-based deduplication approach has a much longer cryptographic hash computing latency. In our experiments, the cryptographic hash computing latency of SHA-256 is twice as long as that of MD5 and SHA-1, thus the average response time of the SHA-256-based deduplication approach is much higher than that of MD5- and SHA-1-based deduplication approaches.

Fig. 8 indicates that EaD's advantage over the traditional deduplication approaches is the most pronounced under the Kernel dataset. The reason is that the deduplication ratio of the Kernel dataset is higher than those of the other three datasets, as shown in Table 2. The higher the deduplication ratio, the more write requests can be replaced by read requests with EaD, thus enabling EaD to achieve much shorter response time. However, the sampling-based approach performs worse than the MD5-based approach, because there is the least amount of non-redundant data for the former to filter out in the Kernel dataset, which reduces the benefit of sampling. On the other hand, since the cryptographic hash computing latency dominates the response time in the traditional MD5/SHA-based deduplication systems and fingerprints must be generated for all the data chunks regardless of their redundancy status, their average response time is not as sensitive as EaD's to the changes in deduplication ratios of the datasets.

An Analytical Model of Response Time. To help further explain EaD's superiority in response time more intuitively, we use a simplified analytical model to analyze the average response time. Let T, R and W represent the average response time, flash storage read latency and write latency of a 4KB data chunk respectively. And let H and $Ratio$ denote the cryptographic hash computing time and the deduplication ratio respectively. We make a simplifying but reasonable assumption that the average write response time is dominated by the flash storage read (for EaD) and write (for baselines and EaD) latencies and hash compute latencies (for baselines) that lie along the deduplication critical path.

The average response time of writing a 4KB data chunk in traditional MD5/SHA-based deduplication system is:

$$T_{base} = W * (1 - Ratio) + H_{base}.$$

The average response time of writing a 4KB data chunk in EaD-based deduplication system is:

$$T_{EaD} = W * (1 - Ratio) + R * Ratio$$

Fig. 9 shows a comparison of analytical response times between traditional MD5/SHA-based deduplication approaches and EaD as a function of the deduplication ratio. The values of the variables, R, W and H, are derived from experimentally measured data on an Intel 750 Series 400GB PCIe-based SSD, as shown in Fig. 1 in Section 2.1. It must be noted that the estimated results are based on the simplified, first-

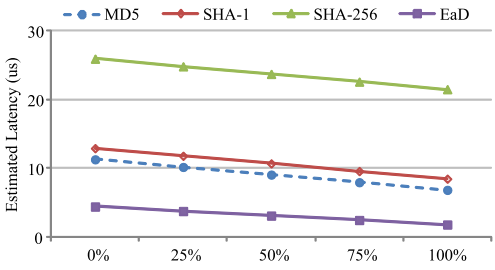


Fig. 9. A comparison of analytical response times between traditional MD5/SHA-based deduplication approaches and EaD as a function of the deduplication ratio.

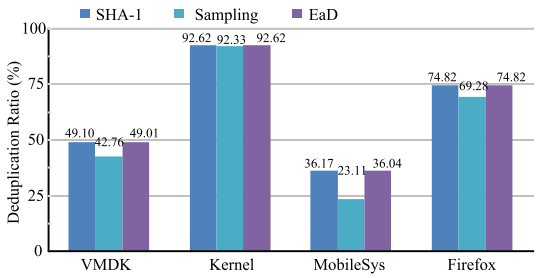


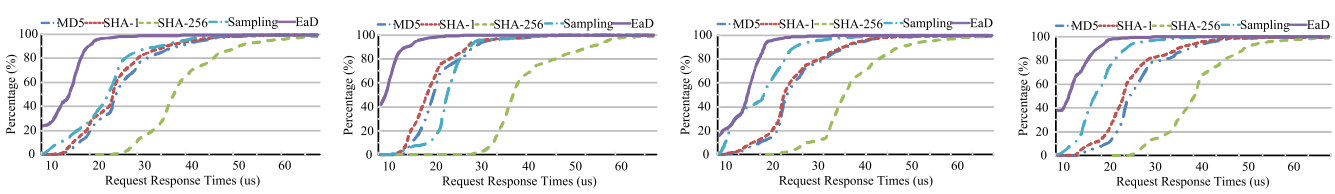
Fig. 11. Deduplication ratios of different approaches.

order analytical model that does not consider second-order system overheads in the real platform. Fig. 9 indicates that the response times decrease as the deduplication ratio increases for all the approaches. The reason is that a larger deduplication ratio means that more data are redundant, thus reducing a larger amount of write traffic to the flash-based storage devices. However, as we explained above, the cryptographic hash computing latency dominates the response time of the traditional MD5/SHA-based deduplication systems and is much longer than the write latency of modern flash-based storage devices. Therefore, the average response time of the traditional MD5/SHA-based deduplication systems is very high. It is also the reason why we propose EaD to eliminate the cryptographic hash computing overhead in flash-based deduplication storage systems.

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 for quality of user experience, particularly for user-facing applications [29]. 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 [30], [31]. One of EaD's objectives is to avoid the cryptographic hash computing latency bottleneck in traditional MD5/SHA- 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 an Intel 750 Series 400GB PCIe-based SSD driven by the four datasets, as shown in Fig. 10. First, the results illustrate that 99% of requests can be completed with a much shorter latency by EaD than any of the other four approaches, across all four datasets. As shown in Fig. 10, for the EaD-based deduplication system, 99% of requests are completed within 36us, 21us, 31us, and 27us under the VMDK, Kernel, MobileSys, and Firefox datasets respectively. However, for the MD5-/SHA-256-based deduplication systems, their 99-percentile latencies are 53us/71us, 38us/56us, 54us/73us, and 52us/71us under the VMDK, Kernel, MobileSys, and Firefox datasets, respectively. The reason is that in the MD5/SHA-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 [2], [3], the unavoidable hash computing overhead in traditional systems will significantly degrade the system performance of modern flash-based storage systems. Increasingly, however, one must balance the benefit of reduced write traffic and the increased cryptographic hash computing latency in face of the steadily improving performance of modern flash-based storage devices. Second, with the SHA-256-based deduplication system, even fewer, if any, requests can be completed within 20us due to the long cryptographic hash computing latency associated with the SHA-256 algorithm. Again, there is a clear trade-off between the much lower probability of hash collisions enabled by SHA-256 and its much longer cryptographic hash computing latency that leads to long tail latency.

Deduplication Ratio. As described in Section 3.4, 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, Fig. 11 shows a comparison in deduplication ratios among the SHA-1, sampling, and EaD approaches and indicates that the deduplication ratio of EaD is almost the same as that of SHA-1, with a maximum reduction of 0.13% under the



(a) VMDK (b) Kernel (c) MobileSys (d) Firefox

Fig. 10. Response time distributions for the different deduplication approaches on an Intel 750 Series 400GB PCIe-based SSD driven by the four datasets, 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.

TABLE 4
The Percentages of Each Case Among the Four Access Conditions

Cases	*VMDK*	*Kernel*	*MobileSys*	*Firefox*
Case 1	47.29%	7.36%	60.82%	24.78%
Case 2	3.62%	0.002%	3.08%	0.40%
Case 3	0.08%	0.018%	0.06%	0
Case 4	49.01%	92.62%	36.04%	74.82%

MobileSys dataset. The reason is that the hash collision rate of BCH-based ECC is no more than 0.2% higher than that of SHA-1 as shown in Fig. 2 in Section 2. 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 MD5/SHA-based deduplication systems can guarantee in theory. However, the sampling-based approach noticeably decreases the deduplication ratio due to its inaccurate detection of redundancy [2]. In summary, EaD provides similar deduplication ratios to those of traditional MD5/SHA-based deduplication approaches, but has 0% false positive redundancy detection and much higher system performance.

Percentages of Different Access Conditions. Fig. 4b shows four access conditions in EaD to identify whether a data chunk is redundant. Table 4 gives the percentages of each case among the four access conditions. First, case 1 and case 4 dominate accesses that determine whether the incoming page is unique (case 1) or redundant (case 4), accounting for 96.6% to 99.9% of all accesses. It indicates that using the ECC information to identify data similarity is an effective method. Second, it is the accesses in case 2 and case 3, which confirm that the incoming page is not redundant, that are affected by either the false positive rate of the Bloom filter or the hash collision issues. These accesses (case 2 & case 3) account for only 0.1% to 3.4% of all accesses, indicating that the extra overhead is negligible.

Memory Overhead. Memory overhead for storing fingerprints is unavoidable in deduplication systems. By default, EaD uses the 8-byte BCH-based ECC code word that is widely used in the flash-based storage devices and the Linux operating system [32]. Compared with the traditional MD5/SHA- and sampling-based deduplication approaches, EaD introduces the extra memory overhead for the prefetch cache. Fig. 12 shows a comparison in memory consumptions among the different deduplication approaches under the four datasets, indicating that EaD consumes significantly less memory than the traditional MD5/SHA- and sampling-based deduplication approaches. The reason is that EaD uses the Blake2 of the BCH-based ECC code word as the index in the mapping tables, it uses fewer bytes per entry than the MD5/SHA- and sampling-based deduplication approaches.

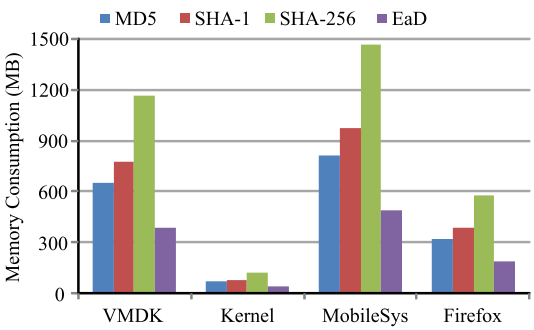


Fig. 12. A comparison among the different deduplication approaches in their memory consumptions under the four datasets.

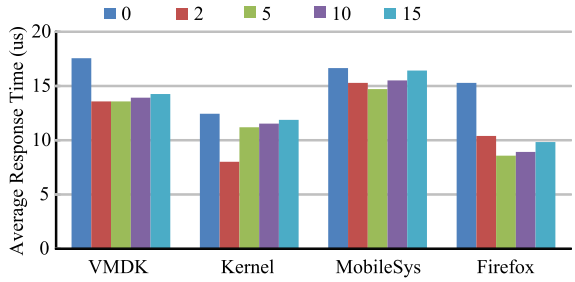


Fig. 13. Average response time as a function of prefetch distance in EaD under the four datasets.

4.3 Sensitivity Study

Reading previously written data from the flash-based storage devices is the main performance overhead of EaD. The Prefetch Cache is designed to address this performance overhead. The number of data chunks prefetched each time, also called *prefetch distance*, clearly affects the performance overhead because whenever a prefetched data chunk is hit in the cache, the read latency from flash storage is avoided in EaD. To assess the impact of prefetch distance on performance, we conduct experiments to evaluate the average response time as a function of prefetch distance under the four datasets, as shown in Fig. 13.

In the experiments, the triggered data chunk is set on the last data chunk of the prefetch set. Fig. 13 clearly indicates that the Prefetch Cache indeed improves the system performance over the system without prefetch (i.e., prefetch distance = 0), which also implies that content locality exists in the data streams [33]. Since different datasets have different locality characteristics, the optimal prefetch distance varies among the four datasets. For example, while for the Kernel dataset, the optimal prefetch distance is 2; it is 5 for the other three datasets. Since the Kernel dataset has many small files, a small prefetch distance is sufficient to exploit its content locality. Moreover, the longer the prefetch distance, the higher the prefetching read overhead will be. In summary, the prefetch cache, combined with the high read performance of modern flash-based storage devices, helps significantly mitigate the read overhead of EaD.

4.4 Trace-Driven Simulator Evaluations

The SSD organization of SSDsim simulator is configured by default. For the performance parameters, we use the Z-NAND latency to configure the SSDSim simulator, shown in Table 5. For the Samsung Z-NAND technique, the read latency of Z-NAND is 3us, which is nearly 20 times faster than conventional NAND [10]. Our preliminary result shows that hashing a 4KB page is 20 times slower than that of hashing a 104B block. Considering that the Blake2 is faster than SHA-1, this implies that the Blake2 latency is less than 1us for

TABLE 5
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 [34]	1

a 104B BCH code (4224, 4120, 8). It must be noted that the latency of SHA-based hashing is configured to be the lowest based on Fig. 1, which implies that the following improvements of EaD are the lower bound. With a larger SHA-based hashing latency, the improvements achieved by EaD will be much more significant.

Fig. 14a shows comparisons in terms of the average write response time among traditional SHA-, sampling-based deduplication approaches and EaD, driven by the four traces. The results show that EaD has the lowest average write response time among the four approaches and EaD outperforms the SHA-1-, SHA-256-, and sampling-based deduplication approaches by an average of 3.4×, 6.7×, 4527.7× and 4.1×, respectively. The reason is similar to that in the real platform evaluations. By utilizing the existing device-generated ECC values, EaD does not incur any cryptographic hash computing latency on the critical write path. Because of the moderate data redundancy, the saved cryptographic hash computing latency for the unique data chunks is significant, which directly improves its system performance over the SHA-based deduplication approaches. The SHA-based deduplication approaches will add the cryptographic hash computing latency for all data chunks to be written, no matter they are redundant or unique. Because of the moderate data redundancy in primary storage, the deduplicaiton-induced overhead is significant for the unique data chunks because of the added extra cryptographic hash computing latency that EaD aims to eliminate. On the other hand, we also see that EaD achieves larger improvements than that in the real platform evaluations because some overheads are not included in the simulator by default. It's also the reason that we choose both the real platform and the simulator evaluations in our study.

Fig. 14b shows comparisons in terms of the average read response time. Because read requests have higher priority to be serviced in the SSDSim simulator, all the schemes have similar read latency. However, due to the limitation of request queue length of SSDs, the read latency of EaD can be much lower than SHA-1-, SHA-256-, and sampling-based deduplication approaches for the IO intensive traces.

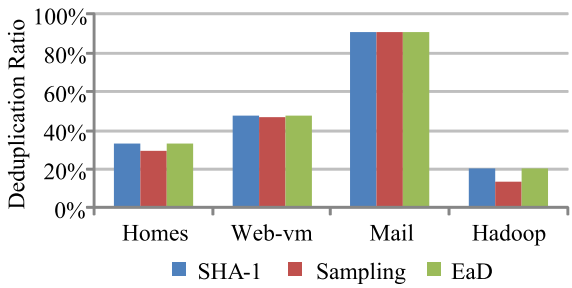


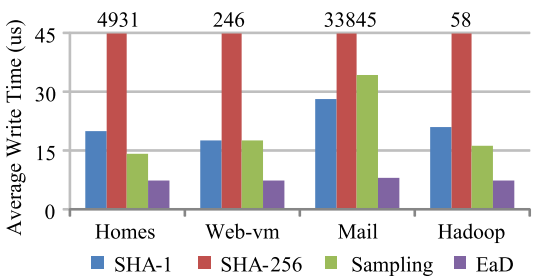
Fig. 15. A comparison among the SHA, Sampling-, and EaD-based approaches in deduplication ratios.

Fig. 15 shows a comparison in deduplication ratios among the SHA-1/256-, sampling-, and EaD-based approaches and indicates that the deduplication ratio of EaD is the same as that of SHA-1 under the four traces. The reason is that EaD can identify all the similar data chunks with the ECC-based similarity detection approach. Moreover, by byte-by-byte comparisons, EaD guarantees that the surmised redundant data chunks are definitively redundant. It implies that there are no false positive detections of redundant data chunks and thus no unrecoverable false data removals. By contrast, MD5/SHA-based deduplication systems cannot provide such a guarantee in theory. In summary, EaD not only significantly improves the deduplication efficiency, but also can completely eliminate the hash collision problem from the existing MD5/SHA-based deduplication for flash-based storage.

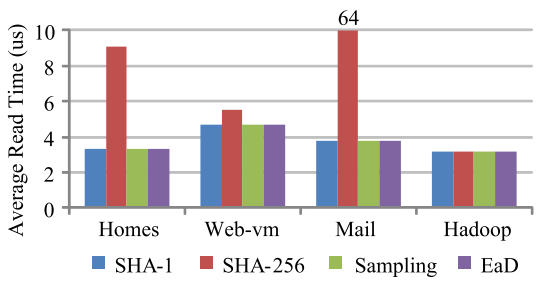
5 Related Work

In face of the explosive growth of digital data, data deduplication has gained increasing attention and popularity in large-scale storage systems from both academia and industry [8], [41]. 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 [36], [42], [43] or hardware-assisted optimizations that leverage the data-parallel processing capabilities of the GPU technology [37], [38], [39].

Generally speaking, the traditional data deduplication process can be divided into four stages: data chunking that can be either Fixed-Size Chunking (FSC) or Content-Defined Chunking (CDC), hash computing, index querying, and index and



(a) Average Write Time



(b) Average Read Time

Fig. 14. Response times of SHA, sampling-, and EaD-based deduplication schemes driven by the FIU and Hadoop traces.

TABLE 6
A High-Level Comparison Between EaD and Prior Art Most Closely Related to EaD

Method	*Device*	*Collision Freedom*	*Chunking*	*Hash*	*Performance Acceleration Method*
Guo [35]	HDDs	No	FSC	SHA-1/MD5	Software-based pipeline and parallelism
P-Dedup [36]			CDC	SHA-1/MD5	
StoreGPU [37]	HDDs	No	CDC	SHA-1/MD5	GPU-accelerated computing offload
Shredder [38]					
GHOST [39]					
CAFTL [2]	Flash	No	FSC	SHA-1	Sampling and light-weight pre-hashing
CA-SSD [3]				MD5	On-board cryptographic co-processor
Kim [40]				SHA-1	Hardware logic with sampling-based filtering
EaD	Flash	**Yes**	FSC	**ECC**	Leveraging the existing device-generated ECC values and high read-performance of ultra-low latency flash storage

metadata updating. Liu *et al.* [42] propose THCAS, a dedicated five-stage storage pipeline design that overlaps the CPU-bound (i.e., chunking and fingerprinting), I/O-bound (i.e., index and store) and network communication tasks in deduplication-enabled storage systems. Inspired by THCAS, P-Dedupe [36] 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.* [35] propose an event-driven and multi-threaded client-server interaction model to pipeline FSC-based deduplication systems. Ma *et al.* [43] 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. GPU has been demonstrated to have stronger processing power than CPU for many compute-intensive applications, especially for the hash and cryptographic computations in high-performance storage systems. Therefore, GPU-based hardware accelerators for the hash computation of the deduplication-enabled storage systems have been proposed. StoreGPU [37] and Shredder [38] accelerate the popular compute-intensive primitives (i.e., chunking and fingerprinting) in deduplication-enabled storage systems by exploiting the massively parallel processing power of GPUs. Similarly, GHOST [39] offloads the deduplication tasks of chunking, fingerprinting and indexing to GPGPU to remove the computing bottleneck in high-performance primary storage systems. In summary, software-based solutions can be easily implemented in deduplication-enabled 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.

Primary storage systems in the cloud have moderate data redundancy, where the deduplication technique can also bring significant cost saving for primary storage systems [7], [44]. For example, the CAFTL [2] and CA-SSD [3] 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.* [40] propose SHA-1 hardware logic with sampling-based filtering to alleviate the SHA-1 processing overhead and SES [45] only addresses scrambler-induced conflict problem which is orthogonal to the EaD approach. Table 6 summarizes studies most closely related to EaD. Different from the traditional MD5/SHA-based deduplication approaches, EaD does not need data chunking and hash computing, but leverages the existing device-generated ECC values to identify the similar data chunks and optimizes the write performance by leveraging the high read performance characteristics for ultra-low latency flash storage.

6 Conclusion

Data deduplication is a popular technology for space efficiency and reliability in flash-based storage systems by reducing write traffic and space capacity requirements. However, it also introduces noticeable processing overhead on the critical I/O path, which significantly degrades the system performance for modern ultra-low latency flash devices. To address the problem, we propose an ECC-assisted Deduplication approach (called EaD) that avoids cryptographic hash computing altogether by leveraging the existing on-device ECC function 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 EaD prototype implementation on a real platform show that EaD significantly outperforms the existing MD5/SHA- and sampling-based approaches in terms of I/O performance by up to 4.2×, with an average of 2.5×.

References

[1] B. Schroeder, R. Lagisetty, and A. Merchant, "Flash reliability in production: The expected and the unexpected," in *Proc. 14th USENIX Conf. File Storage Technol.*, 2016, pp. 67–80.
[2] 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 *Proc. 9th USENIX Conf. File Storage Technol.*, 2011, pp. 77–90.
[3] A. Gupta, R. Pisolkar, B. Urgaonkar, and A. Sivasubramaniam, "Leveraging value locality in optimizing NAND flash-based SSDs," in *Proc. 9th USENIX Conf. File Storage Technol.*, 2011, pp. 91–103.
[4] B. Mao, S. Wu, H. Jiang, X. Chen, and W. Yang, "Content-aware trace collection and I/O deduplication for smartphones," in *Proc. 33rd Int. Conf. Massive Storage Syst. Technol.*, 2017, pp. 1–7.
[5] "With nimble, less is more," 2018. [Online]. Available: https://www.nimblestorage.com/its-all-about-data-reduction/
[6] J. Colgrove *et al.*, "Purity: Building fast, highly-available enterprise flash storage from commodity components," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2015, pp. 1683–1694.
[7] K. Srinivasan, T. Bisson, G. Goodson, and K. Voruganti, "iDedup: Latency-aware, inline data deduplication for primary storage," in *Proc. 10th USENIX Conf. File Storage Technol.*, 2012, pp. 1–14.

[8] W. Xia *et al.*, "A comprehensive study of the past, present, and future of data deduplication," *Proc. IEEE*, vol. 104, no. 9, pp. 1681–1710, Sep. 2016.
[9] Intel Optane SSD 800P Series, 2018. [Online]. Available: https://www.intel.com/content/www/us/en/products/docs/memory-storage/solid-state-drives/consumer-ssds/800p-series-brief.html
[10] W. Cheong *et al.*, "A flash memory controller for 15us ultra-low-latency SSD using high-speed 3D NAND flash with 3us read time," in *Proc. Int. Solid-State Circuits Conf.*, 2018, pp. 338–340.
[11] Google Security Blog on first SHA1 collision, 2017. [Online]. Available: https://shattered.io/
[12] MD5 Collision, 2006. [Online]. Available: http://www.mscs.dal.ca/selinger/md5collision/
[13] J. Zhang *et al.*, "FlashShare: Punching through server storage stack from kernel to firmware for ultra-low latency SSDs," in *Proc. 13th USENIX Symp. Oper. Syst. Des. Implementation*, 2018, pp. 477–492.
[14] BLAKE2–fast secure hashing, 2020. [Online]. Available: https://blake2.net/blake2.pdf
[15] The ARM Cortex R5 processor, 2016. [Online]. Available: https://developer.arm.com/ip-products/processors/cortex-r/cortex-r5
[16] High performance PCIe SSD Controllers, 2020. [Online]. Available: https://www.marvell.com/storage/ssd/88ss1092–93/
[17] New Elements to Samsung SSDs: The MEX Controller, Turbo Write and NVMe, 2013. [Online]. Available: https://www.anandtech.com/show/7152/new-elements-to-samsung-ssds-the-mex-controller-turbo-write-and-nvme
[18] Crypto++ Benchmarks, 2019. [Online]. Available: https://www.cryptopp.com/benchmarks.html
[19] V. Regulapati, "Error correction codes in NAND flash memory," Master's thesis, Master Sci. Eng., Univ. Texas at Austin, Austin, TX, USA, Dec. 2015.
[20] Y. Luo, "Architectural techniques for improving NAND flash memory reliability," Ph.D. dissertation, Doctor Philosophy, Carnegie Mellon Univ., Pittsburgh, PA, USA, Aug. 2018.
[21] C. Li, P. Shilane, F. Douglis, H. Shim, S. Smaldone, and G. Wallace, "Nitro: A capacity-optimized SSD cache for primary storage," in *Proc. USENIX Annu. Tech. Conf.*, 2014, pp. 501–512.
[22] B. Bloom,"Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.
[23] F. Deng and D. Rafiei, "Approximately detecting duplicates for streaming data using stable bloom filters," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2006, pp. 25–36.
[24] F. Ni and S. Jiang, "RapidCDC: Leveraging duplicate locality to accelerate chunking in CDC-based deduplication systems," in *Proc. ACM Symp. Cloud Comput.*, 2019, pp. 220–232.
[25] 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 *Proc. Int. Conf. Supercomputing*, 2011, pp. 96–107.
[26] Intel X25-M Mainstream SATA SSD, 2015. [Online]. Available: https://www.intel.com/content/www/us/en/support/products/79678/memory-and-storage/enthusiast-ssds/intel-ssd-750-series.html
[27] S. Wu, J. Zhou, W. Zhu, H. Jiang, Z. Huang, Z. Shen, and B. Mao, "EAD: A collision-free and high performance ECC assisted deduplication scheme for flash storage," in *Proc. 38th IEEE Int. Conf. Comput. Des.*, 2020, pp. 155–162.
[28] FIU IODedup traces, 2011. [Online]. Available: http://iotta.snia.org/traces/391
[29] J. Dean and L. Barroso,"The tail at scale," *Commun. ACM*, vol. 56, no. 2, pp. 74–80, 2013.
[30] M. Hao, G. Soundararajan, D. Kenchammana-Hosekote, A. Chien, and H. Gunawi, "The tail at store: A revelation from millions of hours of disk and SSD deployments," in *Proc. 14th USENIX Conf. File Storage Technol.*, 2016, pp. 263–276.
[31] 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 *Proc. 15th USENIX Conf. File Storage Technol.*, 2017, pp. 15–28.
[32] Shared BCH ECC library, 2011. [Online]. Available: https://lwn.net/Articles/426856/
[33] W. Xia, H. Jiang, D. Feng, and Y. Hua, "SiLo: A similarity-locality based near-exact deduplication scheme with low RAM overhead and high throughput," in *Proc. USENIX Annu. Tech. Conf.*, 2011, pp. 26–28.
[34] Y. Lee, H. Yoo, I. Yoo, and I.-C. Park, "High-throughput and low-complexity BCH decoding architecture for solid-state drives," *IEEE Trans. Very Large Scale Integration Syst.*, vol. 22, no. 5, pp. 1183–1187, May 2014.
[35] F. Guo and P. Efstathopoulos, "Building a high-performance deduplication system," in *Proc. USENIX Annu. Tech. Conf.*, 2011, pp. 1–14.
[36] W. Xia, H. Jiang, D. Feng, L. Tian, M. Fu, and Z. Wang, "Exploiting parallelism in data deduplication system," in *Proc. 7th IEEE Int. Conf. Netw. Archit. Storage*, 2012, pp. 338–347.
[37] S. Al-Kiswany, A. Gharaibeh, E. Santos-Neto, G. Yuan, and M. Ripeanu, "StoreGPU: Exploiting graphics processing units to accelerate distributed storage systems," in *Proc. ACM Int. Symp. High-Perform. Parallel Distrib. Comput.*, 2008, pp. 165–174.
[38] P. Bhatotia, R. Rodrigues, and A. Verma, "Shredder: GPU-accelerated incremental storage and computation," in *Proc. 10th USENIX Conf. File Storage Technol.*, 2012, Art. no. 14.
[39] C. Kim, K. Park, and K. Park, "GHOST: GPGPU-offloaded high performance storage I/O deduplication for primary storage system," in *Proc. Int. Workshop Program. Models Appl. Multicores Manycores*, 2012, pp. 17–26.
[40] J. Kim *et al.*, "Deduplication in SSDs: Model and quantitative analysis," in *Proc. 28th Int. Conf. Massive Storage Syst. Technol.*, 2012, pp. 1–12.
[41] P. Shilane, R. Chitloor, and U. Jonnala, "99 deduplication problems," in *Proc. 8th USENIX Workshop Hot Top. Storage File Syst.*, 2016, pp. 86–90.
[42] C. Liu, Y. Xue, D. Ju, and D. Wang, "A novel optimization method to improve de-duplication storage system performance," in *Proc. 15th Int. Conf. Parallel Distrib. Syst.*, 2009, pp. 228–235.
[43] J. Ma, B. Zhao, G. Wang, and J. Liu, "Adaptive pipeline for deduplication," in *Proc. 28th IEEE Conf. Massive Storage Syst. Technol.*, 2012, pp. 1–6.
[44] A. El-Shimi, R. Kalach, A. Kumar, A. Oltean, J. Li, and S. Sengupta, "Primary data deduplication - large scale study and system design," in *Proc. USENIX Annu. Tech. Conf.*, 2012, Art. no. 26.
[45] Z. Yan, H. Jiang, S. Jiang, Y. Tan, and H. Luo, "SES-Dedup: A case for low-cost ECC-based SSD deduplication," in *Proc. 35th Symp. Mass Storage Syst. Technol.*, 2019, pp. 292–298.

Suzhen Wu (Member, IEEE) received the BE and PhD degrees in computer science and technology and computer architecture from the Huazhong University of Science and Technology, Wuhan, China, in 2005 and 2010, respectively. Since August 2014, she has been an associate professor with Computer Science Department, Xiamen University. She has more than 50 publications in journal and international conferences, including *IEEE Transactions on Computers*, IEEE-TCAD, IEEE-TPDS, ACM-TOS, USENIX FAST, USENIX LISA, ICS, ICDCS, ICCD, MSST, DATE, SRDS, and IPDPS. Her research interests include computer architecture and storage system. She is a member of ACM.

Chunfeng Du (Member, IEEE) is currently working toward the PhD degree with Computer Science Department, Xiamen University. He has two papers accepted or published in IEEE-TC and IPDPS 2021 on deduplication-enabled flash storage. His research interests include flash and NVM storage systems.

Weidong Zhu (Member, IEEE) received the BE degree in computer science and technology from the Huazhong University of Science and Technology, and the master's degree from Computer Science Department, Xiamen University. His papers have been published in IEEE-TCAD, IPDPS, and ICCD. His research interests include flash-based storage systems, SSD-based disk arrays, Key-Value store, and data deduplication.

Jindong Zhou received the BE degree in software engineering from Xiamen University, Fujian, China, where he is currently working toward the master's degree in software engineering. His papers have been published in IEEE-TPDS, IEEE-TCAD, MSST, ICCD, and SRDS. His research interests include flash storage systems and data deduplication.

Hong Jiang (Fellow, IEEE) is currently the chair and Wendell H. Nedderman endowed professor with the Computer Science and Engineering Department, University of Texas at Arlington (UTA). Prior to joining UTA, he was the program director with National Science Foundation from January 2013 to August 2015 and he was with the University of Nebraska-Lincoln in 1991, where he was the willa cather professor of computer science and engineering. He has more than 300 publications in main journals and international conferences in these areas, including IEEE-TPDS, IEEE-TC, *Proceedings of IEEE*, ACM-TACO, ACM-TOS, JPDC, ISCA, MICRO, USENIX ATC, FAST, EUROSYS, SOCC, LISA, SIGMETRICS, ICDCS, IPDPS, MIDDLEWARE, OOPLAS, ECOOP, SC, ICS, HPDC, INFOCOM, and ICPP. His current research interests include computer architecture, computer storage systems and parallel I/O, high-performance computing, big data computing, cloud computing, and performance evaluation. His research has been supported by NSF, DOD, and industry. He is a member of ACM.

Bo Mao (Member, IEEE) received the BE degree in computer science and technology from Northeast University, Shenyang, China, in 2005, and the PhD degree in computer architecture from the Huazhong University of Science and Technology, Wuhan, China, in 2010. He is currently an associate professor with Software School, Xiamen University. He has more than 50 publications in international journals and conferences, including IEEE-TC, IEEE-TCAD, IEEE-TPDS, ACM-TOS, USENIX FAST, USENIX LISA, ICS, ICDCS, ICCD, DATE, MSST, SRDS, and IPDPS. His research interests include storage system, cloud computing, and Big Data. He is a member of ACM and USENIX.

Lingfang Zeng (Member, IEEE) received the BS degree in computer application from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2000, the MS degree in computer application from the China University of Geoscience, China, in 2003, and the PhD degree in computer architecture from HUST in 2006. He was a research fellow with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore, during 2007–2008 and during 2010–2013, and a visiting professor with Johannes Gutenberg University Mainz, Germany during 2016–2018. He is currently a PI with Zhejiang Lab. He has authored or coauthored more than 60 papers in major journals and conferences.