SES-Dedup: a Case for Low-Cost ECC-based SSD Deduplication

Zhichao Yan*†✉, Hong Jiang*, Song Jiang*, Yujuan Tan‡✉, Hao Luo§
*Department of Computer Science & Engineering University of Texas-Arlington, Arlington, USA
†Hewlett Packard Enterprise, USA
‡School of Computer Science Chongqing University, Chongqing, China
§Twitter, USA
✉Corresponding Author: zhichao.yan@mavs.uta.edu, tanyujuan@gmail.com

Abstract

Integrating the data deduplication function into Solid State Drives (SSDs) helps avoid writing duplicate contents to NAND flash chips, which will not only effectively reduce the number of Program/Erase (P/E) operations to extend the device's lifespan but also proportionally enlarge the logical capacity of SSD to improve the performance of its behind-the-scenes maintenance jobs such as wear-leveling (WL) and garbage-collection (GC). However, these benefits of deduplication come at a non-trivial computational cost incurred by the embedded SSD controller to compute cryptographic hashes. To address this overhead problem, some researchers have suggested replacing cryptographic hashes with error correction codes (ECCs) already embedded in the SSD chips to detect the duplicate contents. However, all existing attempts have ignored the impact of the data randomization (scrambler) module that is widely used in modern SSDs, thus making it impractical to directly integrate ECC-based deduplication into commercial SSDs. In this work, we revisit SSD's internal structure and propose the first deduplicatable SSD that can bypass the data scrambler module to enable the low-cost ECC-based data deduplication. Specifically, we propose two design solutions, one on the host side and the other on the device side, to enable ECC-based deduplication. Based on our approach, we can effectively exploit SSD's built-in ECC module to calculate the hash values of stored data for data deduplication. We have evaluated our SES-Dedup approach by feeding data traces to the SSD simulator and found that it can remove up to 30.8% redundant data with up to 17.0% write performance improvement over the baseline SSD.

I. Introduction

How to manage the explosive growth of data is a top-priority problem in the big data era. There exists a significant amount of redundant data in the fast expanding digital universe [1], which makes data deduplication (dedup), a space and compute efficient solution for storage capacity optimization, a standard feature for a lot of storage products and installations such as backup systems, file systems, all-flash arrays, cloud storage systems and so on.

Several studies have proposed integrating data deduplication into SSDs to leverage their internal processing capability to achieve such single-instance storage within the SSD. These deduplication-enabled SSDs can not only exploit the benefits of reducing the Program/Erase (P/E) operations to increase SSD's lifespan but also proportionally enlarge its logical capacity to improve the performance of its behind-the-scenes tasks such as wear-leveling (WL) and garbage-collection (GC) [2, 3]. Moreover, deduplication will further improve SSD's reliability because the raw bit error rate will increase sharply with the number of P/E operations [4, 5] increasing. However, deduplication will incur notable computation and space overheads [6]. As a result, designers must balance the potential overheads and benefits of deduplication to make the right design choice.

Different from traditional data deduplication systems running on general-purpose computer systems or servers, deduplication within SSDs is usually constrained by the very limited resources within the SSDs (such as embedded processor and DRAM). In traditional deduplication systems, data streams will be dynamically divided into either variable-length or fixed-size chunks, where a cryptographic hash (such as SHA-256) is calculated per chunk as the fingerprint to uniquely represent the chunk's content, and chunks detected to share the same fingerprint with an already stored unique chunk are considered duplicates and eliminated by replacing them with a pointer each to the unique chunk. Unfortunately, dynamically chunking and computing cryptographic hashes will incur a great deal of computational overhead to SSD's embedded controller, which will affect its frontend I/O performance, thereby offsetting deduplication's advantages. For example, even for some mid-range all-flash array products with the server-level processors, their performance is CPU-bound, which is mainly caused by the data reduction tasks of deduplication and compression [7, 8]. Therefore, we believe that removing substantial compute overhead of fingerprint generation is able to significantly improve the performance of these systems, especially for the smart SSD devices with the built-in deduplication function running on the embedded SSD controller.

Recent studies have proposed some low-overhead approaches such as fixed-size chunking (at page size) and using a page's ECC instead of SHA-256 as the fingerprints to help detect the duplicate chunks, which leverages the SSD's internal page mapping mechanism and the page-level ECC

without incurring extra computation cost and at a smaller space cost than that required to maintain the hash table for data deduplication [2, 3]. Unfortunately, on SLC, MLC, TLC or QLC flash chips, different data patterns written have been proven to exhibit different raw bit error rates because of various electronic interference effects [9]. To reduce the raw bit error rates induced by these similar data pattern's disturbances, modern SSDs have integrated a data scrambler module to randomize the incoming data before storing it to the NAND flash chips. As a consequence, ECCs for duplicate data blocks with different LBAs will be rendered completely different, which makes it impossible to leverage this built-in ECC function as a content identification function to detect the duplicate data on flash chips.

In this work, we propose a Scrambler-resistant ECC-based SSD deduplication, called SES-Dedup, which can be implemented in either the host or device side to break through the data scrambler module so as to enable low-cost ECC-based deduplication on modern SSDs. Through extensive evaluations on an simulated SES-Dedup system, SES-Dedup is shown to effectively exploit SSD's built-in ECC module to calculate the hash values of stored data for data deduplication. Specifically, our SES-Dedup approach can remove up to 30.8% redundant data with up to 17.0% performance improvement by feeding our collected data traces to the SSD simulator.

The rest of the paper is organized as follows. Section II describes the background and motivation of this work. Section III elaborates on the design of SES-Dedup. Other implementation issues are described in Section IV. We present the evaluation and analysis in Section V and conclude the paper in Section VI.

II. Motivation

Continued capacity growth and lower unit prices have made SSDs a mainstream storage device. However, future increases in density will result in a significant drop of their performance and reliability. As a result, SSD manufacturers and users will carefully weigh the tradeoff among cost, performance, capacity, and reliability. In this work, we propose to integrate deduplication in SSD to enable an SSD that can run this data reduction technology by itself to avoid writing duplicate contents and enlarge the logical capacity.

Redundant data are prevalent in the digital universe, which makes data deduplication and compression viable and profitable. Given the minimum chunking cost of fixed-size chunking in deduplication and hence its wide use in flash-based consumer electronics, we want to know the most effective fixed-size chunking granularity to help detect the duplicate data in these consumer electronics such as laptops and desktops. As such, we have collected data from two laptops and four desktops that have applied different fixed chunking sizes to the data, and analyze their data redundancy ratios through a data deduplication engine. As illustrated by Figure 1, we have obtained two important observations: one is that there exists a lot of redundant data in these laptops and desktops, which is up to 37.0% on desktop 4; the other is that most redundant data

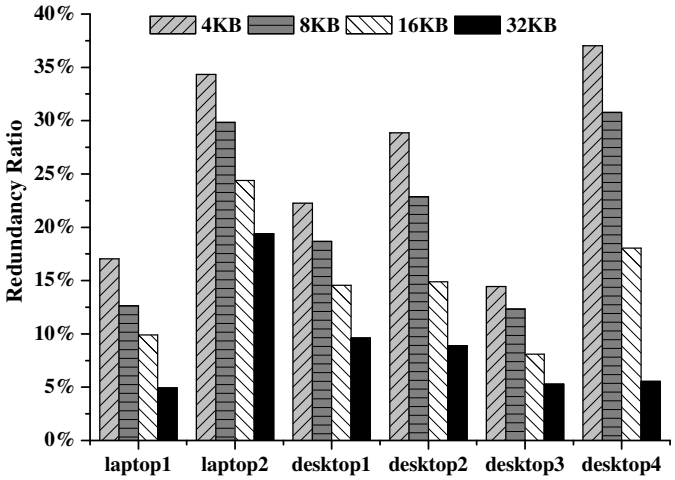

Fig. 1: Data redundancy rates of fixed-size chunking

TABLE I: Page-level operation latencies on different flash chips

NAND Type	Read	Write	SHA-256
SLC	23.4 us	262.6 us	226.5 us
MLC-1	33.5 us	390.0 us	
MLC-2	43.3 us	1084.4 us	

can be found in 8 KB chunks comparing to 4 KB chunking, whose size is close to modern SSD's page size. This study affirms that using SSD's page size as the fixed chunking size can detect most redundant data, which in part motivates this work.

Table I lists the page-level read and write operation latencies on several typical NAND flash chips, and the latency of calculating one SHA-256 hash of an 8KB page on ARM Contex processor running at 400 MHz. Obviously, this hashing operation will add non-trivial latency to the write latency if we adopt traditional SHA-256 as the fingerprint for SSD dedup. Moreover, the longer write operation may also increase the read latency, which will degrade both the read and write operations if SHA-256 is used as the fingerprint for SSD deduplication. As shown in Figure 2, the SSD performance drops after enabling SHA-256 based deduplication on different NAND flash chips, because the SHA-256 hashing overhead is incurred on the critical path of every write operation, besides the extra memory overheads on maintaining deduplication metadata. These mixed read-and-write workloads are generated by the FIO tool to process the input data without any deduplicatable pages to learn its deduplication overheads mainly caused by calculating SHA-256. When we feed our collected datasets with up to 30.7% redundant pages at our selected 8 KB page size, this SHA-256 deduplicaiton-enabled SSD can slightly improve the baseline SSD's write performance, by up to 5.8%. By this test, we find that SHA-256 based dedup in SSDs incurs non-trivial overheads (up to 17.3%), that will significantly offset most benefits of deduplication and make it unattractive to be integrated within the SSDs.

Meanwhile, NAND flash chips usually face much more

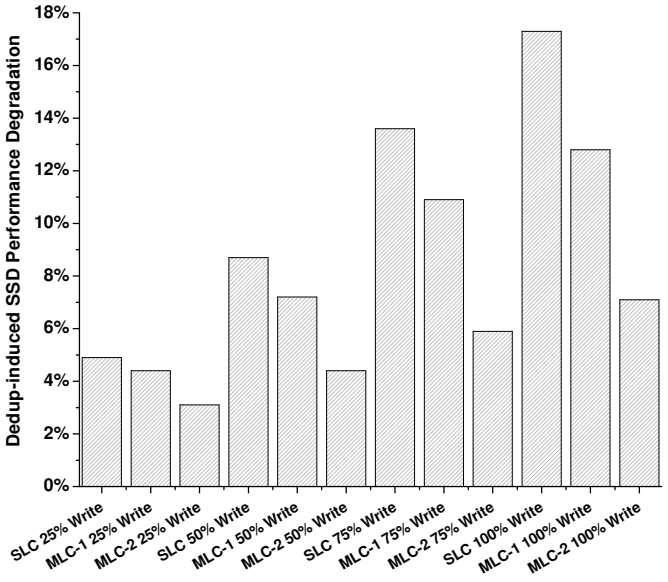

Fig. 2: SSD performance degradation after adding SHA-256 based deduplication on different types of NAND flash chips with different mixed random read-and-write workloads on fixed chunking of size 8 KB

transient failures caused by program disturb, read disturb, over-programming or charge loss due to their inherent device-level characteristics. As a result, ECC has become a mandatory feature integrated into SSDs to detect and correct these transient failures to make SSDs useable. The SSD controller will first divide a page into one or more blocks when a host writes a page of data to SSD. Then, its encoding module will generate a codeword consisting of the block and ECC for each block by its generator matrix. A page's ECC can be treated as the concatenation of all block ECCs within that page.

In this work, we argue that this kind of ECC information sealed within SSD can be leveraged as the content identification of the stored data in lieu of the costly cryptographic hash computation of the stored data for the purpose of redundancy detection. In general, the computation overhead of calculating ECC is much lower than SHA-256, which motivates us to reuse this ECC as the first-round fingerprint for SSD deduplication that is backed up by a byte-to-byte comparison [10] to verify the duplicate contents for possible false positives (due hash collisions). This can help filter out most distinct contents in an efficient way since there is no false negative in ECC value comparisons. For those contents that might have a duplicate counterpart in the SSD, we need to further compare their contents. As shown in Table I, SES-Dedup's byte-to-byte comparison operation will exploit the asymmetric latencies of read and write operations to read the potential duplicate page and compare it with the contents to be written. As a result, besides removing the high computation cost of SHA-256 on the embedded SSD controller, our SES-Dedup can use low-cost read operation to replace the high-cost write operation for those duplicate pages, thus significantly reducing the P/E operations that will increase SSD's lifespan and reliability.

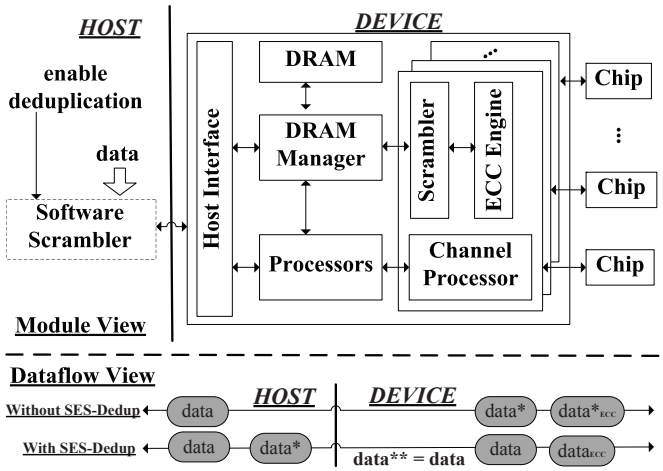

Fig. 3: Architecture of host-side SES-Dedup system

III. SES-Dedup System Design

In this Section, we show the basic design of SES-Dedup, which can be implemented at either host-side or device-side plus some necessary modifications in device's FTL. More implementation issues are discussed in Section IV.

We observe that the scrambler module usually adopts a Linear Feedback Shift Register (LFSR) to generate a scrambling vector by using the LBA of the scrambled data block as the randomization seed [11, 12]. This scrambling vector will then be XORed with the origin data to generate the scrambled data. Upon a read request, the scrambled data will be descrambled by the same logic because the XOR operation is reversible. Therefore, we can break through this scrambler module by generating the scrambled data at the host-side, so as to perform the on-demand deduplication on SSDs by exploiting their built-in ECCs, whose dataflow overview is shown in the lower part of Figure 3.

Figure 3 shows a high-level architectural view of host-side SES-Dedup design. Its right upper part shows the major functional components (i.e., Scrambler next to ECC Engine) integrated within a typical SSD controller. In particular, the incoming data will be scrambled before calculating their ECCs, and then written to the chips. This makes ECC-based deduplication impossible because the ECC values of duplicate data blocks with different LBAs will be completely different after passing through the scrambler module. In this case, a deduplication-enabled SSD must keep extra information, such as the fingerprints of data blocks before scrambling, to help deduplication, which will incur both computation and space overheads. As shown in the left upper part of Figure 3, we add a software scrambler module at the host side to help SSD reverse the randomized data to the original data written by the host and store them on the flash chips.

For a specific SSD controller, we can easily recreate the scrambler in software by writing some predefined data patterns (like all '0') to the SSD. In this case, we know the content of the input data and its LBA, and we can dump the chip's image to obtain the scrambled version of the input data so as to infer

and recover the scrambler function of the embedded scrambler module inside the device and recreate the software scramble module at the host-side. By this approach, we can perform the on-demand deduplication on SSD by enabling the host-side software scramble module to randomize the data before writing to the SSD, then it will be scrambled again in SSD, and finally storing the origin data on NAND flash chips. Thus its ECC is leveraged to detect the duplicate content to perform the fix-sized chunk data deduplication algorithm on SSD.

Host-side SES-Dedup system can selectively bypass the embedded scramble module in SSD, but it will store the data without scrambling on the NAND flash chips, which might increase the raw data error rates, although we can modify FTL to help redirect writing the original data to different physical units to avoid/limit writing similar patterns in the same block. In order to overcome this problem, we propose a device-side SES-Dedup system, which is based on the distributive property of matrix multiplication. In particular, as shown in Equation 1, ECC vector is usually calculated by a generator matrix multiplication operation, whose input is a scrambled data vector obtained by the origin data XORing the scramble vector. Based on Equation 2, we can obtain the original data's ECC as its content identification because the generator matrix and all scramble vectors are known, which can be integrated within the firmware to support deduplication in SSD. Therefore, we can implement the device-side SES-Dedup system without storing the origin data on NAND flash chips. Once the device-side SES-Dedup system reads the ECC from NAND flash chips, the SSD controller can further calculate its original data's ECC and use it as the fingerprint to detect the duplicate data. As a result, this device-side design might incur a little latency for calculating its original data's ECC.

$$([V_{Data}] \oplus [V_{Scrambler}]) \times [M_{Encoding}] = [ECC] \quad (1)$$

$$[V_{Data}] \times [M_{Encoding}] = [ECC] \oplus [V_{Scrambler}] \times [M_{Encoding}] \quad (2)$$

IV. Other Implementation Issues

There are two types of deduplication performed on SES-Dedup, in-line deduplication and post-processing deduplication. While the former will not only reduce the P/E operations but also enlarge an SSD's logical capacity, the latter can only enlarge an SSD's logical capacity because it has written the contents to the flash chips before performing deduplication.

In the SES-Dedup system, an ECC-based fingerprint is used as the first-level filter to identify a potential duplicate page with a matched ECC value, which will then be processed by a byte-to-byte comparison with the other page's contents to verify the redundancy. Such byte-to-byte operations will help replace high-cost write operations with low-cost read operations to exploit the asymmetric latencies of read and write operations. Because the FTL has a lot of other functions (e.g., logical block mapping, wear leveling, garbage collection, write amplification, bad block management, etc.) to perform, there is very limited computation power and memory space left in the SDD controller that can be used for in-line deduplication. As a result, we believe that post-processing deduplication will be a must-have option for SSDs to deduplicate the redundant contents. That is, periodically scanning the SSD to identify and remove the redundant data in the background or during idle periods by leveraging the ECC information as its fingerprint, which saves a lot of costs associated with computing SHA-256 to support deduplication on SSDs.

The integration of the data deduplication feature in FTL will change SSD's logical-to-physical mapping from 1-to-1 to n-to-1. Multiple LBAs will be mapped to a single Physical Block Address (PBA), which is fine for normal read/write operations. However, this will not work for the garbage-collection (GC) task, because it must notify all associated LBAs that their corresponding PBA's content will be moved to another PBA. As a result, SES-Dedup adds a reverse lookup mechanism to check all LBAs associated with a specific PBA for garbage collection.

Host-side and device-side SES-Dedup systems are designed for different application scenarios. The host-side design is suitable for personal usage that provides a flexible on-demand interface to enable the deduplication feature on SSDs. There are two main reasons for this application scenario. First, different data generated by different applications have shown to exhibit significantly different data redundancy characteristics, i.e., very little redundancy exists among data generated by different applications [13], making it unnecessary to deduplicate data generated by fundamentally different applications. Second, applications or file systems may write the same metadata to multiple logical blocks to avoid potential data loss due to single-block failures, which means that devices should not eliminate such intentional data redundancy by the deduplication feature to reduce the risk of data loss. On the other hand, the device-side design is more suitable for large-scale data center, which contains a lot of different SSDs, whose host may not have sufficient computation power to run the software scramble module to support a lot of SSDs simultaneously. Moreover, it can seal the deduplication function within the device that provides better compatibility.

Some SSD controllers may integrate a data compression engine to compress the incoming data before writing it to the flash chips. If the compression is performed after SSD's data scrambler module, there is no impact on our host-side SES-Dedup design although it will hurt the compression ratio because the data has been randomized. When the compression module is placed before the scrambler module, we should disable the software scrambler module at the host side and write some extra ECC data, leveraging the compression algorithm's ECC as the content's fingerprints rather than the NAND flash's ECC [14], to the Out-of-Band space to help perform data deduplication on SSDs. On the other hand, in our device-side SES-Dedup design, compression will not impact its function because all necessary work can be processed within the device.

A page has several codewords, which can provide a finer

TABLE II: Configurations of SSD simulator

Description	Configuration
Flash Page Size	8 KB
Pages per Block	256
Block per Plane	256
Plane per Package	8
Number of Packages	8
Garbage Collection Threshold	5%
Flash Erase Latency	1.8 ms

deduplication granularity at an ECC codeword rather than a whole page. It will be especially useful for large page size because we find that the most predominate fixed-size chunking granularity that can detect most duplicate data is around 8 KB. We will explore this part as our future work.

V. Evaluation

A. Experimental Environment

We evaluate SES-Dedup on GEM5 full system simulator [15], whose SSD model is ported from the extended FlashSim simulator [16] and integrates with ECC-based deduplication functions. We set the major parameters of the host as a 1.6 GHz X86 CPU plus an eight-bank 8 GB DDR3-1600 DRAM. The SSD configurations are listed in Table II, while the read and write latencies on different flash chips are listed in Table I. We shrink stimulated SSD size to 32 GB with 64 MB DRAM to make our collected data easily saturate its capacity. Each codeword of 1 KB is protected by a code rate of 32/33 LDPC code (i.e., the coding redundancy is 256 B per 8 KB data page). In particular, calculating SHA-256 hash will take 226.5 us, host-side SES-Dedup will not incur extra hashing computation cost on the SSD device and recalculating LDPC ECC will be 14.5 us on a 400 MHz ARM processor for device-side SES-Dedup system.

Our data sets are collected from two laptops and four desktops, which contain the typical office workloads, such as coding, file editing, Internet surfing, emailing, file sharing, running virtual machine, etc. These data sets have normal redundancy ratios, which vary from 12.3% to 30.8% at 8KB fixed chunking size. We use the FIO tool [17] to create the synthetic data access traces based on these data sets to evaluate the performance of SES-Dedup system.

B. ECC-based Fingerprint Filter

First of all, we must design a Fingerprint Filter to help reduce the size of the in-memory fingerprint table because SSD's embedded DRAM capacity is limited and cannot hold all fingerprints. In SES-Dedup design, we truncate each codeword's ECC from 32 B to 4 B to form a page's ECC fingerprint, which has the total length of 32 B (256-bit). We do not observe any hash false positives by replacing SHA-256 with this 256-bit ECC because an SSD's capacity is small. Specifically, for a 32 GB capacity SSD with 8 KB page size, it only contains 4 M pages, and the 256-bit fingerprint

TABLE III: Skew-distributed duplicated pages

	Hot Fingerprint Ratios	Ratios in Redundant Data
laptop1	17.6%	74.1%
laptop2	13.8%	86.3%
desktop1	15.8%	79.8%
desktop2	14.9%	81.1%
desktop3	18.8%	72.1%
desktop4	12.7%	89.3%

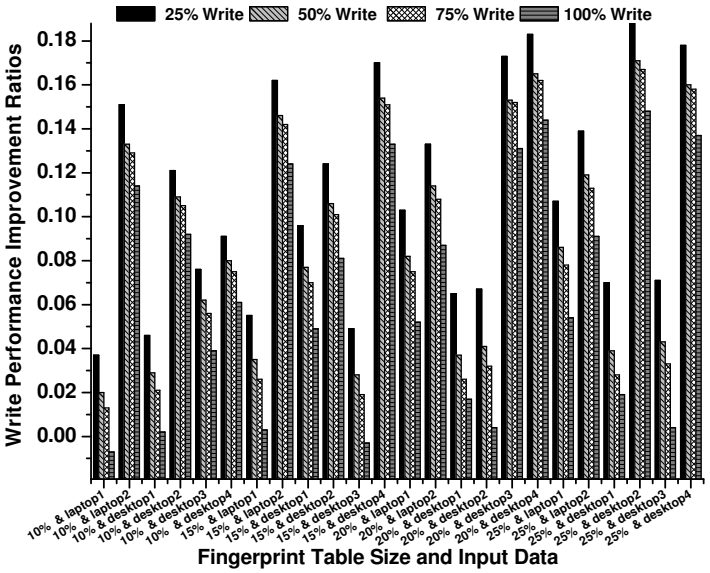

Fig. 4: Study of write performance improvements on different fingerprint table size

length has provided enough hash space to avoid any collisions, which can even fully meet 8 TB SSD's requirement. However, 4 M 256-bit ECC fingerprints will occupy 128 MB memory, which is larger than the total simulated SSD's DRAM capacity (64 MB).

As shown in Table III, the distribution of duplicated pages is highly skewed in these data sets. We have observed that 12.7% to 18.8% hot duplicated pages, whose reference count is larger than 2, have occupied about 72.1% to 89.3% of total redundant data. In other words, it provides an opportunity to optimize the fingerprint table, thus making it small enough while containing most duplicated fingerprints. In order to achieve this goal, we have designed a fingerprint table that can store 15.0% of SSD's total number of fingerprints. By this approach, the fingerprint table shrinks from 128 MB to 19.2 MB, but still occupying a lot of DRAM capacity because most DRAM is used to store FTL's mapping table. We further reduce each fingerprint entry's length to shrink the fingerprint table's size. In this design, we sample a quarter of the ECC-based fingerprints, which reduce fingerprint table size to 4.8 MB that can be fit within the limited DRAM (around 7.5% space overhead).

Figure 4 shows various random write performance improvements on simulated SLC SSD with different fingerprint table size ratio, where 10% means this fingerprint table can

TABLE IV: In-line and off-line deduplication processing redundancy data ratios on the host-side SES-Dedup system with 100% random write workload

Data Set	In-line Dedup			Off-line Dedup			Duplicate Ratio
	SLC	MLC-1	MLC-2	SLC	MLC-1	MLC-2	
laptop1	7.1%	6.5%	5.4%	5.5%	6.1%	7.2%	12.6%
laptop2	17.4%	16.1%	12.9%	12.5%	13.8%	17.0%	29.9%
desktop1	11.0%	9.9%	8.1%	7.7%	8.8%	10.6%	18.7%
desktop2	13.7%	12.1%	9.9%	9.2%	10.8%	13.0%	22.9%
desktop3	6.5%	6.1%	5.2%	5.8%	6.2%	7.1%	12.3%
desktop4	18.2%	16.9%	13.6%	12.6%	13.9%	17.2%	30.8%

store 10% of all possible fingerprints (0.4 M in this test), the MLC results with the same trend are omitted due to the space limit. From this test, we find that different data sets exhibit different random write performance improvements due to their different duplicated data distributions. When this table size ratio increases from 15% to 20%, the performance gains are diminishing, thus indicating that 15% of max table size can obtain the best price/performance ratio, which is mainly determined by the skew distribution of duplicated pages. Specifically, SES-Dedup system can improve up to 17.0% random write performance under this setting.

C. Effectiveness of Deduplication

SES-Dedup can in-line deduplicate every redundant page (up to 30.8% duplicate data at 8 KB fixed chunking size) without considering the limited computation power of embedded SSD controller. In order to reduce the performance interferences caused by the in-line deduplication processing, both host-side and device-side SES-Dedup systems try their best to process deduplication in-line while leaving the other pages to be processed off-line. As a result, we want to know how much redundant data can be detected by in-line deduplication because it inform us of several important metrics, such as how many P/E operations are saved, how much space can be reclaimed by post-processing deduplication, and so on.

The host-side SES-Dedup system is controlled by the host-side module, which will limit the contents to be deduplicated on SSDs. It adds negligible overheads to check the fingerprint table for deduplication. We have collected the in-line and off-line deduplication processing ratios in Table IV. For example, 52.9% to 59.8% duplicated data is processed by in-line deduplication under SLC SSD. In other words, it can directly reduce 6.5% to 18.2% of the data written to SLC chips, which will reduce the corresponding P/E operations. Meanwhile, those remaining duplicated pages to be processed by off-line (post-processing) deduplication will further increase about 5.5% to 12.6% storage capacity on SLC SSD. In this case, we will not compare it with SHA-256 based deduplication because the device is not changed by adding extra hardware/software module to calculate SHA-256 except for some necessary deduplication changes, such as adding the fingerprint table and modifying FTL's mapping table. From this test, the MLC SSDs process less in-line deduplication data because its inherent access latency is higher than SLC device, which prolongs its write latency.

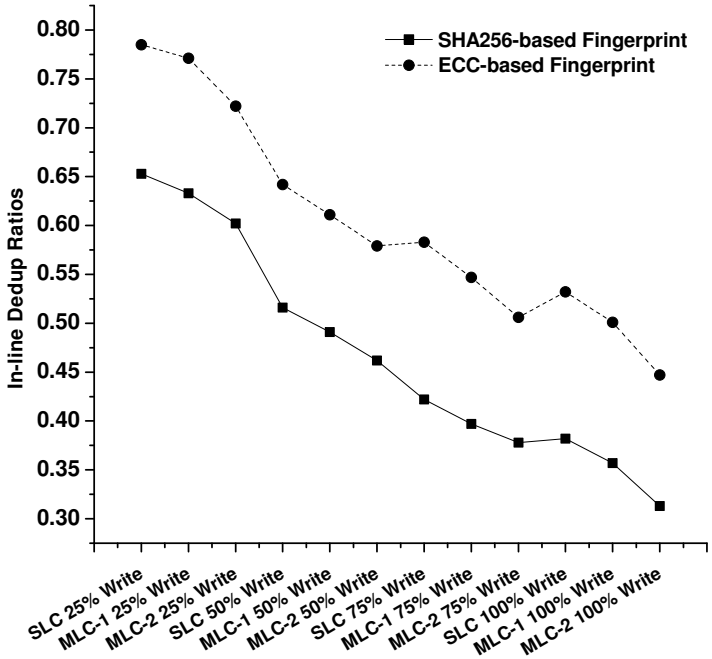

Fig. 5: In-line deduplication ratios of device-side SES-Dedup with different hash fingerprints

Different from the host-side approach, the device-side SES-Dedup system will add the ECC processing latency to support its deduplication function. As shown in Figure 5, we plot the geometric means of in-line deduplication ratios of different data sets under different mixed read-and-write workloads. We find that the majority of duplicated pages can be detected and removed inline while leaving some pages to be processed off-line in ECC-based SES-Dedup approach, which means that it will reduce the corresponding duplicate writes to the NAND flash chips. Specifically, it can process 19.9% to 42.8% more duplicated data in-line than SHA256-based approach, which means that a lot of P/E operations can be saved by this ECC-based approach.

VI. Conclusion

In this work, we propose the SES-Dedup system to help bypass the data scrambler module within SSD to enable the low-cost ECC-based data deduplication on SSD. Our experimental results show that it can extend the flash space by

a factor of up to 30.8% with up to 17.0% write performance improvement over the baseline SSD with our collected real-world data sets.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for their valuable feedback and constructive suggestions. This research is partially supported by research grant from NetApp, the U.S. National Science Foundation (NSF) under Grant Nos. CCF-1704504 and CCF-1629625, Chongqing High-Tech Research Program (cstc2016jcyjA0274), National Natural Science Foundation of China(61402061), and Fundamental Research Funds for the Central Universities (2018CDXYJSJ0026). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] J. Gantz and D. Reinsel, "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east," https://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf, 2012.

[2] J. Kim, C. Lee, and et.al, "Deduplication in ssds: Model and quantitative analysis," in *Proceedings of the 28th IEEE Symposium on Mass Storage Systems and Technologies*, 2012, pp. 1–12.

[3] 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 *Proceedings of the 9th USENIX Conference on File and Stroage Technologies*, 2011, pp. 6–6.

[4] S. Boboila and P. Desnoyers, "Write endurance in flash drives: Measurements and analysis," in *Proceedings of the 8th USENIX Conference on File and Storage Technologies*, 2010, pp. 9–9.

[5] L. M. Grupp, A. M. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, P. H. Siegel, and J. K. Wolf, "Characterizing flash memory: Anomalies, observations, and applications," in *42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2009, pp. 24–33.

[6] F. Guo and P. Efstathopoulos, "Building a high-performance deduplication system," in *Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference*, 2011, pp. 25–25.

[7] "Performance and tuning considerationsfor sas on pure storage fa-420 flash array," 2014, SAS Institute Inc.

[8] Y. T. Jin, S. Ahn, and S. Lee, "Performance analysis of nvme ssd-based all-flash array systems," in *2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2018, pp. 12–21.

[9] Y. Cai, S. Ghose, and et.al, "Error characterization, mitigation, and recovery in flash-memory-based solid-state drives," *Proceedings of the IEEE*, pp. 1666–1704, 2017.

[10] B. Hong, D. Plantenberg, D. D. E. Long, and M. Sivan-Zimet, "Duplicate data elimination in a san file system," in *Proceedings of the 21st IEEE Symposium on Mass Storage Systems and Technologies (MSST)*, 2004.

[11] J. Cha and S. Kang, "Data randomization scheme for endurance enhancement and interference mitigation of multilevel flash memory devices," *ETRI Journal*, vol. 35, no. 1, pp. 166–169, 2013.

[12] C. Chang, M. Bekerman, I. Swarbrick, and C. Hanson, "Xor-based scrambler/descrambler for ssd communication protocols," 2016, patent: US20160328567A1.

[13] Y. Fu, H. Jiang, and et.al, "Aa-dedupe: An application-aware source deduplication approach for cloud backup services in the personal computing environment," in *2011 IEEE International Conference on Cluster Computing*, 2011, pp. 112–120.

[14] Z. Yan, H. Jiang, and et.al, "Deduplicating compressed contents in cloud storage environment," in *Proceedings of the 8th USENIX Conference on Hot Topics in Storage and File Systems*, 2016.

[15] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011. [Online]. Available: http://doi.acm.org/10.1145/2024716.2024718

[16] M. Bjørling, "Extended flashsim," 2011. [Online]. Available: https://github.com/MatiasBjorling/flashsim

[17] J. Axboe, "Flexible i/o tester." [Online]. Available: https://github.com/axboe/fio