JOURNAL OF SEMICONDUCTOR TECHNOLOGY AND SCIENCE, VOL.15, NO.5, OCTOBER, 2015
http://dx.doi.org/10.5573/JSTS.2015.15.5.539

ISSN(Print) 1598-1657
ISSN(Online) 2233-4866

Offline Deduplication for Solid State Disk Using a Lightweight Hash Algorithm

Eunsoo Park and Dongkun Shin*

Abstract—**Deduplication technique can expand the lifespan and capacity of flash memory-based storage devices by eliminating duplicated write operations. The deduplication techniques can be classified into two approaches, i.e., online and offline approaches. We propose an offline deduplication technique that uses a lightweight hash algorithm, whereas the previous offline technique uses a high-cost hash algorithm. Therefore, the memory space for caching hash values can be reduced, and more pages can be examined for deduplication during short idle intervals. As a result, it can provide shorter write latencies compared to the online approach, and can show low garbage collection costs compared to the previous offline deduplication technique.**

Index Terms—Flash memory, SSD, deduplication, lifespan, reliability

I. INTRODUCTION

As flash memory technology advances, memory density is increasing, but the reliability and endurance are declining [1]. The limited number of program/erase (P/E) cycles of flash memory sharply drops as technology scaling continues to high-density flash devices. For example, 2-bit multi-level cell (MLC) flash memory chip permits ten times less P/E cycles than single level cell (SLC) flash memory chip [4]. Therefore, it is important to manage the lifespan of flash memory efficiently.

Flash memory-based storage is composed of multiple flash memory chips. Each chip consists of thousands of erase blocks. A block is normally made up of 64~256 numbers of 4KiB-pages. Flash memory has unique characteristics as follows: (1) Flash memory supports three major operations. The read or write (a.k.a. program) operation is performed in the unit of page. However, the erase operation is performed in the unit of block. (2) The whole block must be erased before writing any page in the block. That is, flash memory does not support in-place overwrite. (3) The pages in an erase block must be written sequentially. (4) The erase block permits a limited number of program/erase cycles. These characteristics of flash memory require a special firmware, called FTL (Flash Translation Layer), to emulate the block storage interface. The FTL provides the following functionalities :

- **Logical-to-physical (L2P) address mapping**: FTL manages the mapping table to convert logical addresses from the file system to physical addresses in flash memory.
- **Garbage collection:** FTL copies the valid pages in a block to other free block and erases the block to reuse it.
- **Power-off recovery:** FTL prevents data in flash storage from losing in sudden power-off situation.
- **Wear-leveling:** FTL provides the functionality for all blocks in flash storage to be worn out evenly.

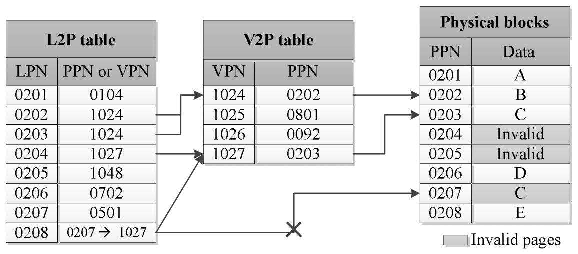
Recent flash memory devices such as solid state disks (SSDs) have smart controllers which can maximize the lifetime of flash memory. There are two approaches to

Manuscript received Apr. 21, 2015; accepted Sep. 24, 2015
College of Information & Communication Engineering,
Sungkyunkwan University, 2066, Seobu-ro, Jangan-gu, Suwon-si, Gyeonggi-do, 16419, Korea
Corresponding Author: Dongkun Shin
E-mail : dongkun@skku.edu

increase the lifetime of flash memory devices. The first one is the wear-leveling technique that monitors the aging level of each flash memory block and makes the blocks to be worn out evenly [2]. The second one is to reduce the total amount of write traffic on flash storage. Many researches have made efforts to reduce the number of write operations on flash storage. For example, an efficient garbage collection (GC) technique can reduce the write amplification factor (WAF) by selecting a low-cost victim block to be reclaimed and thus minimizing the number of page copy operations during the GC. The other example is to reduce the amount of user data with compression or deduplication. While the compression technique can be effective on reducing the data size, it has the size mismatch problem. Since the flash page can be programmed in the unit of page, several compressed pages should be merged before they are programmed. It is difficult to remove the internal fragmentations completely. The deduplication technique detects and removes duplicated pages from the storage device. The compression and deduplication techniques can also reduce the GC cost by minimizing the number of pages to be copied. However, these schemes have considerable runtime overheads.

The deduplication techniques can be classified into the online and offline approaches, depending on when the deduplication is performed. The online scheme checks each incoming page when it arrives, and drops it if a same page exists in the storage. Instead, the FTL firmware changes the address translation table such that the target logical page shares the pre-written physical page with other logical pages. Although the online scheme can reduce the number of write operations, it has the runtime deduplication overhead. Moreover, it has a reliability problem if there is a sudden power-off before the modified address mapping entries are not saved permanently.

The offline deduplication, on the other hand, writes all the data sent from the host at the storage, and removes the duplicated pages during idle time. It can hide the deduplication overhead by exploiting the idle time, and it has no problem related to sudden power-offs. Although the offline technique cannot reduce the number of write operations for handling host requests, it can reduce the GC overhead by eliminating the copy operations on duplicated pages. Both the online and offline techniques need a high complexity of hash algorithm such as SHA-1 and MD5 in order to generate a collision-free fingerprint of each page.


L2P table

LPN	PPN or VPN
0201	0104
0202	1024
0203	1024
0204	1027
0205	1048
0206	0702
0207	0501
0208	0207→ 1027

V2P table

VPN	PPN
1024	0202
1025	0801
1026	0092
1027	0203

Physical blocks

PPN	Data
0201	A
0202	B
0203	C
0204	Invalid
0205	Invalid
0206	D
0207	C
0208	E

▨ Invalid pages

Fig. 1. An example of data deduplication

In this paper, we propose a novel offline deduplication technique, which uses a lightweight hash function such as CRC32 instead of high complexity of hash functions in order to reduce the deduplication overhead. Since our scheme is an offline technique, it can solve the reliability problem of the online technique. In addition, it can reduce the deduplication overhead compared to the previous techniques by using a lightweight hash algorithm.

The rest of this paper is organized as follows. In section II, we discuss on deduplication techniques for flash memory, and address several problems of previous techniques [4, 8]. Section III introduces the design of proposed offline deduplication technique. We present the experimental results in section IV. The last section concludes this paper.

II. RELATED WORKS

Many host-level deduplication techniques have been proposed in order to save the storage space [5], where the file system or storage device controller in the host system performs the deduplication. In the device-level deduplication techniques, on the other hand, the storage controller performs the deduplication work. The device-level deduplication in SSD changes the address mapping table if duplicated data is found. By modifying the mapping table, the logical pages which have identical data can share a same physical page. Since the physical page location can be changed by GC, the data-sharing logical pages point to a shared physical page via the virtual page number as shown in Fig. 1 [3].

Since the logical pages with the LPNs (logical page numbers) of 204 and 208 have the same data of **C**, they are mapped to the VPN (virtual page number) of 1027, which is again mapped to the PPN (physical page number) of 203. The physical page with the PPN of 207 is invalidated after the deduplication.

CAFTL (Content-Aware Flash Translation Layer) [3] proposed both the online and offline deduplication techniques for SSD, although it focuses on the online deduplication as a main deduplication process. In the online scheme of CAFTL, the incoming write request is first temporarily written at the on-device buffer. Then, a collision-free fingerprint such as SHA-1 is computed for each page in the buffer. The generated fingerprint is looked up against a fingerprint store which maintains the fingerprints of all the pages in the storage. If a same fingerprint is found from the fingerprint store, only the address mapping table is updated such that the duplicated pages share a physical page, without writing the new data at flash memory. If no match is found, the write is performed as a regular write. Thus, the average write latency could be reduced in the environment with a high redundancy such as virtualization server. If the redundancy level is low, on the other hand, the online scheme may increase the write latency since it should generate a complex hash value for each page.

Moreover, some deduplicated pages can be lost at sudden power-offs. Generally, FTL does not flush the updated mapping table into non-volatile flash device immediately. Therefore, the storage device may have the old version of address mapping table at the power-off-recovery. In this situation, FTL can recover the mapping entries of all the written pages with the reverse mapping (i.e., physical-to-logical (P2L) page mapping) in the spare (OOB) area of flash pages. However, the deduplicated pages at online scheme are never written in the flash memory and the reverse mapping is not either. Consequently, the FTL cannot recover the deduplicated pages at sudden power-offs. Therefore, the file system consistency will be crashed.

If SSD flushes the mapping table at each deduplication in order to solve the problem, there is no gain by deduplication since each deduplicated page invokes one page write operation. CAFTL assumes that supercap can be used to solve the problem, but it requires additional hardware cost. Therefore, the online deduplication technique is not practical considering the sudden power-off situation.

The offline deduplication technique hides the deduplication overhead by exploiting the idle time of flash storage. All the incoming data is first written at flash memory. The written data is later examined for deduplication during idle time. The previous offline scheme also generates the high complexity of hash value, and it is maintained at the fingerprint store. If a match found in the fingerprint store, the duplicated physical page is invalidated by modifying the L2P mapping table.

In the offline deduplication, we can indirectly reduce the average write latency by reducing GC overhead. The block separation technique for offline deduplication [6] can further reduce the GC cost by separating deduplicable pages into specific blocks so that the pages to be invalidated by deduplication are collected in the blocks. In order to check whether a page has a possibility of duplication (i.e., deduplicable page), the block separation technique uses a light-weight hashing at online.

III. Offline Deduplication with Lightweight Hash

We adopt the offline approach as a deduplication technique considering the reliability issue of the online deduplication. In order to improve the performance of the offline deduplication, the incoming pages are pre-hashed at online with a light-weight hashing such as CRC32, and the pages with a same pre-hash value are managed as candidates for deduplication. Fig. 2 shows the overall architecture of the proposed offline deduplication scheme. For each page in the page buffer, a CRC32 hash is computed and it is searched from the CRC32 table. The CRC32 hash table is cached in the internal DRAM of flash storage. The CRC32 table has multiple buckets. Each bucket is identified with a unique CRC32 value, and it has the list of physical pages whose data have the CRC32 value of the bucket.

If no match is found from the CRC32 table, the page is identified to be unique and it is inserted into a new bucket. If a match is found, the page is undetermined whether they are duplicated or not since the CRC32 hash function is not collision-free. Then, the PPN is inserted at the corresponding bucket in the CRC32 table. The flash memory blocks are classified into the non-determined (ND) blocks and the unique (U) blocks [6]. While the

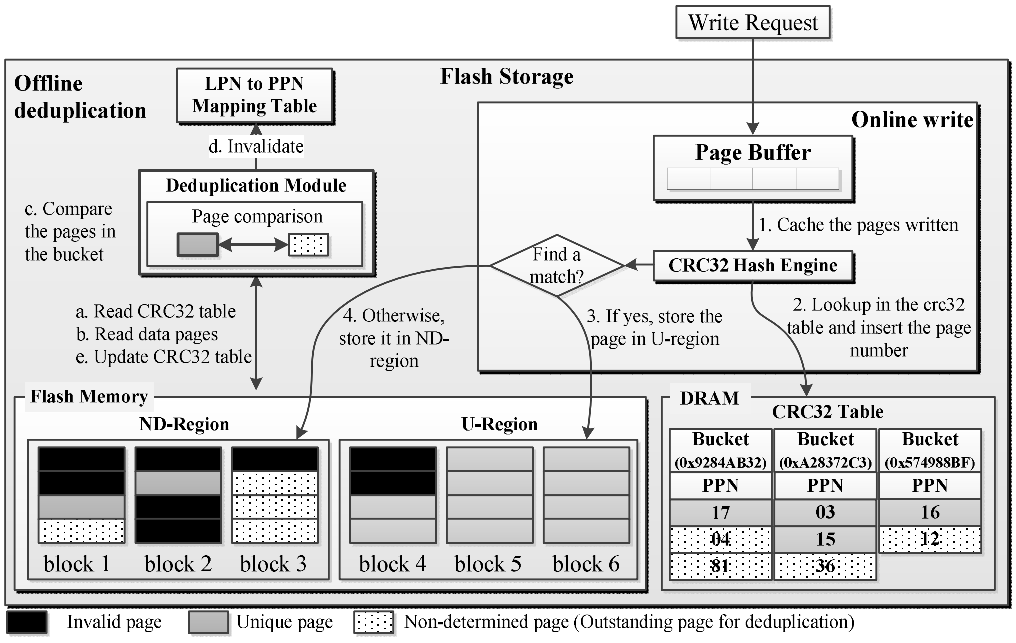


Fig. 2. Architecture of proposed offline deduplication technique

non-determined pages are written at the ND blocks, the unique pages are written at the U blocks. Each entry in the CRC32 buckets has a flag to represent the page type, either ND flag or U flag. The flag can be changed after the offline deduplication.

When the host notices the start of idle time via the SATA interface, legacy SSD normally activates the background GC. In the proposed scheme, however, we set a higher priority on the offline deduplication and allow SSD to activate background GC after the completion of the deduplication process to reduce the page copy overhead. We are solving such an arbitration situation with the novel technique in future work. For example, we can predict deduplication probability of the page by profiling the request and the page with lower probability would be dropped from CRC32 table so as to reduce the deduplication cost. Moreover we also refer to sooner update probability of the victim page, since deduplication process for the page which is about to be updated soon makes useless deduplication. Therefore, in idle time, the candidate pages are examined for duplication with byte-level comparison, so that we can remove the time and space overhead of the collision-free fingerprints. For each physical page with the ND flag in a bucket, it is compared with the physical pages with the U flag in the same bucket. Compared with the previous offline deduplication technique, the proposed technique can reduce the number of comparisons for deduplication by examining only the pages written in the ND region.

If a non-determined page has the same data with a unique page, it is invalidated by updating the L2P mapping table. The corresponding entry of the deduplicated page in the bucket is removed from the CRC32 table. If a non-determined page is identified to be unique, the flag of the entry in the CRC32 table is changed to unique. Each bucket has the reference count, which represents how many duplicated pages were found from the bucket. The bucket with the highest reference count is first examined to find duplicated pages since the bucket has a higher probability of duplication.

Although the byte-level comparison has a high cost, the cost can be reduced in proposed offline deduplication. For the byte-level comparison without hashing, all pages in the flash storage should be read from flash to dram in controller and compared with the target page. Using a light-weight algorithm, however, only few pages have to be read and compared since CRC32 table provides the PPNs to be compared. In addition, the lightweight hash such as CRC32 is still powerful as enough as it does not generate much collision [3], the number of read and comparisons is small. Therefore, the overhead of byte-level comparison is the read overhead for at least two pages and CPU comparison overhead. Concerning CPU comparison overhead, for the single page byte-level

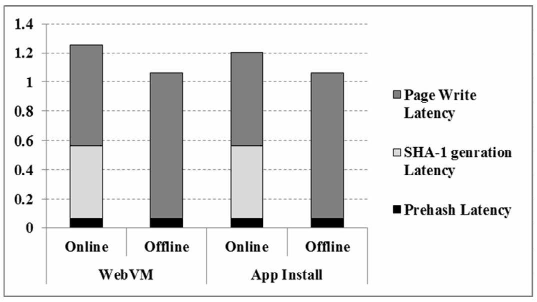


Fig. 3. Response time of write request

comparison CPU spends smaller cycles than lightweight hash key generation according to SimpleScalar. Moreover, it is common that the comparison finished sooner than expected since the different pages are usually not matched in middle.

Since CRC32 table has the PPN instead of LPN, the offline deduplication can directly access the target physical pages without accessing the L2P mapping table. In addition, when a duplicated page is found, the corresponding L2P mapping entry can be accessed with the P2L mapping written at the spare area.

IV. EXPERIMENTS

Our simulation environment was built by inserting the deduplication module at the DiskSim-based SSD simulator [7]. We used two storage access workloads for evaluations, WebVM [8] and Windows Application Installation. The former workload is based on the traces of web mail proxy and online course management, and the latter is generated with the Open Storage Toolkit while installing multiple versions of applications.

We assumed that the latencies of SHA-1, CRC32 hash functions, and 4KiB byte-level comparison overhead are 100 μs, 13 μs, and 10 μs, respectively, which are measured with the SimpleScalar-ARM cycle accurate simulator. We used the default SSDSim parameters of page read and write latencies, i.e., 25 μs and 200 μs, respectively. In order to generate GC, we prepended the dummy write to the trace files for 90% SSD utilization.

Fig. 3 shows the response time of write request, which is normalized by the response time of normal SSD without deduplication. The noticeable differences are shown in both the SHA-1 generation latency and the page write latency between the online and offline

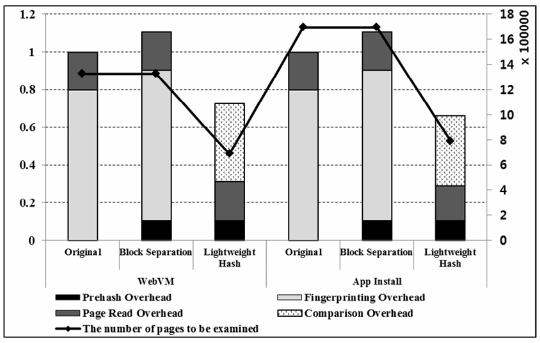


Fig. 4. Offline deduplication overhead

techniques. The online scheme has shorter page write latencies than those of the offline scheme. However, the online scheme has significant fingerprinting overheads. In a high redundancy workload, the reduction in page write latency will be larger than the increased fingerprinting cost. In a low redundancy workload, on the other hand, the fingerprinting cost will be larger than the reduction in page write latency, and thus the offline deduplication will be superior to the online deduplication in the average write latency. The examined two workloads have redundancy rates of 30% and 35%, respectively. Under the low redundancy rates, the offline deduplication shows 19% and 13% of shorter write response times than the online deduplication, respectively.

We also estimated the deduplication overheads under different offline deduplication techniques as shown in Fig. 4. We implemented the original offline deduplication technique of CAFTL [3]. The offline deduplication technique with block separation [6] is also implemented for comparison. Each overhead value is normalized by the overhead of the original offline deduplication technique.

The block separation technique uses a collision-free hash as the original deduplication scheme does, and it also uses the CRC32 hash value in order to separate unique pages and non-determined pages into different blocks. Therefore, the deduplication overhead of block separation technique is the biggest among three techniques. In the proposed technique, even though each page should be read to check duplication, the page read overhead is insignificant since the number of pages written in the ND region is small. As a result, the deduplication overhead is reduced by more than 30% compared to the previous offline deduplication schemes.

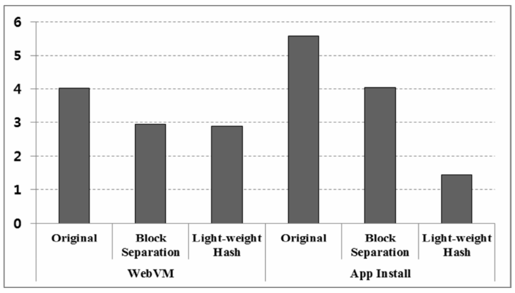


Fig. 5. The number of page copies during GC

Fig. 5 shows the number of page copies during GCs. The numbers are normalized by the value in the online deduplication scheme. The number of page copies depends on the number of deduplicated pages and whether the block separation technique is applied or not. In the proposed offline deduplication scheme, 42% and 23% of pages in the ND region are determined to be unique pages under two workloads, respectively. Since the offline deduplication process is performed only for the ND region, more pages can be deduplicated during short idle intervals. Therefore, more pages can be invalidated before GCs, and the number of page copies during GCs is reduced.

The application installation workload has short idle intervals. The proposed deduplication scheme can examine more pages than other offline deduplication techniques. As a result, the offline deduplication with light hash can reduce the GC overhead compared with the previous offline deduplication techniques.

V. CONCLUSIONS

The reliability and endurance of flash memory is getting worse as memory density is increased. Therefore, it is an important issue to expand the lifespan of flash memory-based storage. In this paper, we proposed an offline deduplication to increase the lifespan of flash memory. We first addressed the reliability and deduplication overhead problems of the previous deduplication techniques. To solve the problems, a novel offline deduplication technique that uses a lightweight hash function is proposed. We evaluated our technique with a SSD simulator. The proposed offline deduplication reduces the GC overhead significantly compared with the previous offline deduplication schemes.

ACKNOWLEDGMENTS

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2013R1A1A2A10013598).

REFERENCES

[1] Y.-H. Chang, J.-W. Hsieh, and T.-W. Kuo, “Endurance enhancement of flash-memory storage systems: An efficient static wear leveling design,” *Design Automation Conference,* pp.212–217, Jun., 2007.

[2] S. Boboila and P. Desnoyers, “Write endurance in flash drives: measurements and analysis ,” *File and storage technologies, the 8th USENIX conference on*, Vol.10, pp.9-9, Feb., 2010.

[3] Chen, Feng, Tian Luo, and Xiaodong Zhang, "CAFTL: A Content-Aware Flash Translation Layer Enhancing the Lifespan of Flash Memory based Solid State Drives," *File and storage technologies, USENIX conference on,* Vol. 11, Feb., 2011.

[4] Andersen, David G., and Steven Swanson, "Rethinking flash in the data center," *IEEE micro 4, pp.*52-54, 2010.

[5] Meyer, Dutch T., and William J. Bolosky, "A study of practical deduplication," *ACM Transactions on Storage, 7.4:14,* 2012.

[6] A. Jeongcheol, S. Dongkun, “Offline Deduplication-Aware Block Separation for Solid State Disk,” *File and storage technologies, USENIX conference on*, 2013.

[7] N. Agrawal et al., “Design Tradeoffs SSD Performance,” *USENIX Annual Technical Conference*, Jun., 2008.

[8] R. Koller et al., “I/O Deduplication: Utilizing Content Similarity to Improve I/O Performance,” *ACM Transactions on Storage,* 6.3, 2010.

Eunsoo Park received the B.S. degree in computer engineering from Sungkyunkwan University, Korea in 2015. He is currently a Master student in the School of Information and Communication Engineering, Sungkyunkwan University. His research interests include embedded software, file systems and flash memory.

Dongkun Shin received the B.S. degree in computer science and statistics, the M.S. degree in computer science, and the Ph.D. degree in computer science and engineering from Seoul National University, Korea, in 1994, 2000 and 2004, respectively. He is currently an Assistant Professor in the School of Information and Communication Engineering, Sungkyunkwan University (SKKU). Before joining SKKU in 2007, he was a senior engineer of Samsung Electronics Co., Korea. His research interests include embedded software, low-power systems, computer architecture, multimedia and real-time systems.