Offline Deduplication-Aware Block Separation for Solid State Disk*

Jeongcheol An
Sungkyunkwan University, Korea

Dongkun Shin
Sungkyunkwan University, Korea

1 Introduction

NAND flash-based solid-state disks (SSDs) are increasingly being deployed in storage systems due to their several advantages over magnetic hard disks: lower access latencies, lower power consumption, and higher robustness. However, their shorter lifetime and garbage collection overhead are critical concerns. As recent SSDs adopt higher density multi-bit-cell flash memory chips in order to provide higher capacity, lifetime is becoming worsen. Deduplication can be an effective solution to increase lifespan as well as space-efficiency in flash memory SSDs. By eliminating the write operation for duplicated data, the deduplication techniques can save the limited endurance of flash memory blocks.

Deduplication can be performed offline or inline. The inline deduplication examines the incoming data and prevents duplicate writes. Therefore, the inline deduplication may increase the write latency due to the duplication checking overhead if most of data are not duplicated or many burst write requests are sent from host. The duplication checking overhead includes the hash key generation time and the key lookup time. The offline deduplication removes the duplicated data to increase space-efficiency during idle time [2]. Although the offline scheme does not increase the write latency, it cannot prevent duplicate writes since the deduplication is done after data has been written on the storage. Therefore, many inline techniques are proposed targeting to increase the lifetime of SSDs [3].

However, regular file systems have relatively low duplication rate and require short write latency compared with backup systems. For such systems, the offline technique will be more efficient. Although the offline deduplication cannot reduce the data amount written at storage, it can reduce the number of page copy operations during garbage collection if the offline deduplication is performed before garbage collection. Since the offline deduplication invalidates duplicated pages, the number of valid page copies will decrease and thus reduces the write amplification ratio of SSDs.

In this paper, we propose a block separation technique for offline deduplication. The goal of block separation is to minimize the garbage collection cost for the flash memory blocks by separating duplicate data and unique data into different flash memory blocks. Since the duplicate data will be invalidated by offline deduplication, the flash memory block assigned for the duplicate data will have many invalid pages and thus the garbage collection will require small number of page copy operations. In experiments with SSD simulator, the proposed block separation technique reduces the page copy cost during garbage collection by 20$\sim$61% compared to the normal offline deduplication.

2 Block Separation

The block separation technique determines the possibility of duplication for the incoming data before the data is written at flash memory blocks. To minimize the overhead on write operation, a lightweight but non-collision-free hash function such as CRC32 is used to filter out the non-duplicated data inline. With the non-collision-free hash function, we cannot identify duplicate data but it can find unique data. Therefore, the incoming data can be divided into two types, unique data and non-determined data.

Figure 1 shows the block separation and deduplication modules within SSD. The incoming data are first split into fixed size of chunks. The chunk size is same to the page size in flash memory (4KB or 8KB). The unique chunk filter generates a 4-bytes CRC32 hash key for a chunk and lookups the key in the CRC32 hash key table which has all hash keys of the written chunks. If the key is not found, the data is definitely unique within the stor-

*This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MEST) (No. 2010-0026511) and (No. 2012-0002117).

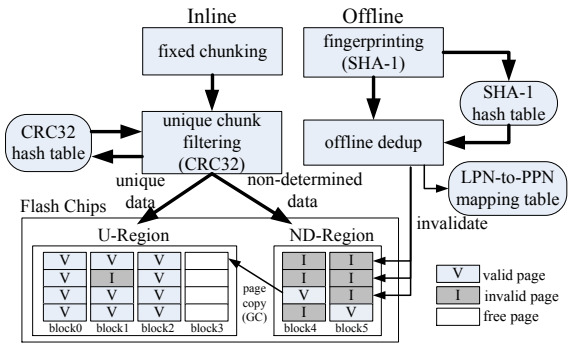


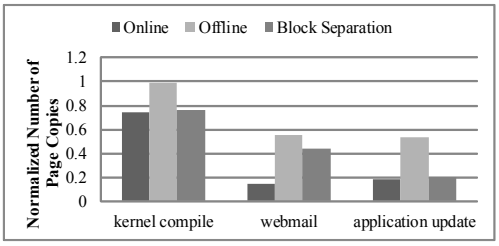
Figure 1: The proposed SSD Architecture.

age. Therefore, the data is written at the block assigned for the unique region (U-Region) and the key value is inserted into the hash table. If the same key is found at the hash table, we cannot determine the duplication of the data since the CRC32 hash function is not collision-free. Therefore, the data should be examined during the offline deduplication with a collision-free hash function such as SHA-1 or MD5. The block separation technique writes the non-determined data at the block assigned for the non-determined region (ND-Region). When there is no free space in each region, a free block is allocated to the region. Therefore, the sizes of U-region and ND-region are variable.

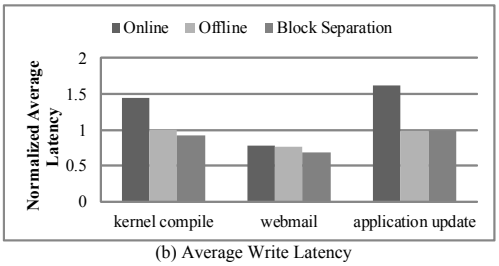
During idle time, the offline deduplication generates the fingerprints (SHA-1 hash keys) of the chunks which are updated after the previous offline deduplication. For only the pages within the ND-Region, the deduplication lookups the generated fingerprints in the SHA-1 hash table to find duplicated data. When the offline deduplication finds duplicated data from ND-region, it invalidates the corresponding physical page, and updates the mapping table, which maintains the address translation between logical page number (LPN) and physical page number (PPN), such that the duplicated logical pages share one physical page. For the pages in U-Region, the fingerprint (SHA-1 hash key) should be generated by the offline deduplication but there is no need to search the key from the SHA-1 hash table to check duplication.

Although the CRC32 hash function is not collision-free, its collision rate is very low. When we examined the collision rate of CRC32 for typical storage traces, the rate was lower than 8×10^{-5}. Therefore, most of pages in ND-region have actually duplicate chunks.

By storing the unique data and the non-determined data into different regions, the flash memory blocks in ND-region may have many invalid pages invalidated by the offline deduplication as shown in Figure 1. The garbage collection will select the block 4 or block 5 for a victim since it has the smallest number of valid pages. In the example, only one page is copied before erasing the block. It can be written at U-Region since it is identified as a unique chunk after offline deduplication.



(a) Number of Page Copies during GC



(b) Average Write Latency

Figure 2: Experimental Results.

3 Performance Evaluation

Our simulation environment was built by inserting the deduplication module at the DiskSim based SSD simulation plug-in [1]. With SimpleScalar-ARM cycle-accurate simulator, we estimated the SHA-1 and CRC32 hash function latencies as 75μs and 6μs, respectively. Three storage access workloads are used for evaluation, `kernel compile`, `webmail`, and `application update`. The `kernel compile` traces are collected during kernel code compiling at Linux machine. The `application update` trace is collected during application version update operations at MS-Windows machine.

Figure 2(a) compares the numbers of page copies during garbage collection under different deduplication techniques. The results are normalized by those under no deduplication scheme. The block separation scheme significantly reduces the page copy operations compared with the normal offline technique. Figure 2(b) compares the average write latencies. The inline dedup shows longer latencies due to the duplication checking overhead. The block separation scheme shows lower latencies than those of the normal offline scheme. This is because the write operations can be delayed by foreground garbage collection. Since the block separation scheme reduces the garbage collection overhead, the write latency delay is also reduced.

References

[1] SSD extension for DiskSim simulation environment. http://research.microsoft.com/enus/downloads/b41019e2-1d2b-44d8-b512-ba35ab814cd4/default.aspx.

[2] ALVAREZ, C. NetApp deduplication for FAS and V-Series deployment and implementation guide. Technical ReportTR-3505.

[3] CHEN, F., LUO, T., AND ZHANG, X. CAFTL: a content-aware flash translation layer enhancing the lifespan of flash memory based solid state drives. In *Proceedings of FAST'11*.