Deduplication in SSD for Reducing Write Amplification Factor*

Jonghwa Kim(student)[1], Ikjoon Son(student)[1], Jongmoo Choi[1], Sungroh Yoon[2], Sooyong Kang[3], Youjip Won[3], Jaehyuk Cha[3]

[1]{zcbm4321, ikjoon, choijm}@dankook.ac.kr, Dankook University, Korea

[2]sryoon@korea.ac.kr, Korea University, Korea

[3]{sykang, yjwon, chajh}@hanyang.ac.kr, Hanyang University, Korea

One of the recent trends of SSDs (Solid State Drives) is applying a multichip/multibus architecture to make use of chip/bus interleaving for archiving high performance. Another trend is employing flash chips based on MLC (Multi Level Cell) technology that can store two or more bits per cell for providing high capacity. However, MLC flash has a notorious drawback, that is a limit on the number of erasures, typically rated at 5K to 10K cycles per block, which is less than a tenth compared to that of SLC (Single Level Cell) flash. As flash memory manufacturers continue to increase bits per cell, the endurance limit is expected to deteriorate as well, becoming a critical problem about the reliability of SSDs [1, 2].

To mitigate the problem, several techniques are investigated in SSDs such as wear-leveling and *writeless* techniques. Wear-leveling is a technique that erases blocks evenly to reduce premature wear for prolonging the lifetime of flash memory. Writeless is a set of techniques including write buffer, compression, and deduplication for reducing WAF (Write Amplification Factor) in flash memory.

In this study, we are focusing on the issues of the deduplication technique in SSDs. Deduplication is a popularly used technique in archival storage [3], virtualization environments [4] and inline backup servers [5] for improving storage utilization and I/O throughputs. The previous studies are all based on servers with disks while our study is based on SSDs with flash memory. Compared with the server environments, deduplication in SSDs raises different research issues in terms of workloads and resource limitations. Besides, how to integrate deduplication with FTL (Flash Translation Layer) is another interesting issue that gives an opportunity to improve not only performance but also reliability of SSDs.

To explore the issues about deduplication in SSDs, we have developed an experimental system as shown in Fig. 1 that can emulate the operations of SSDs. The three main components of the system are an SSD controller, an SATA interface, and an array of flash chips. The controller consists of 400MHz ARM9 CPU, 64MB SDRAM and embedded peripherals. The SATA interface is a software module that issues a sequence of SATA commands based on the traces gathered from a host during the execution of workloads considered in this paper. Finally, the array of flash chips is a software module that emulates the multichip/multibus architecture in SSDs.

On the SSD controller, we have implemented deduplication software, consisting of three software modules: 1) *dedup_engine*, 2) *FP_manager*, and 3) *dedup_map*. The dedup_engine generates a fingerprint using SHA-1 hash function for each SATA command that a host issues to store. The FP_manager maintains a set of fingerprints and detects duplication by comparing the current generated fingerprint with the maintained ones. The dedup_map is a mapping table that translates the logical address of the deduplicated data into the physical address. In this experiments, we set the size of deduplication unit as 4KB, the same size of a page in flash memory. Therefore, a SATA command whose requested data size is larger or smaller than 4KB, the engine divides it or fills zero to adjust 4KB unit. Also, we set the maximum number of fingerprints in the FP_manager as a control parameter and use the LRU policy to choose a victim fingerprint for replacement.

The first issue we want to explore is about workloads. Many previous researches have shown that workloads for servers and archival storage have plenty of redundancies that can be exploited usefully in deduplication. To analyze the redundancies of SSD workloads, we have executed six workloads, namely MS windows install, MS office install, MS windows update, httracks, outlook sync, and wayback machine, as shown in Fig. 2, with the different numbers of fingerprints. The figure shows that the SSD workloads considered in this study provide the deduplication ratio ranging from 6% to 33%. Note that when we take into consideration about the garbage collection overheads per each write in SSDs, the WAF can be reduced larger than the reported ratio.

One interesting observation in Fig. 2 is that the SSD workloads have a strong temporal locality. When we set the number of fingerprints as 100, we can obtain the 50%~90% deduplication ratio of those obtained with the number of fingerprints as 4000. It implies that we can capture the large portion of redundancies with the small size of SDRAM, considering the memory limitation of the SSD controller.

* This work was supported by the IT R&D program of MKE/KEIT No. KI10035202, Development of Core Technologies for Next Generation Hyper MLC NAND Based SSD.

The second issue is about the deduplication overheads in SSDs. Fig. 3 shows the average processing overhead for deduplication measured on ARM 9, the most popular CPU used in SSDs. The results show that the average processing overhead is around 120 μs per deduplication unit, where roughly 100 μs is consumed by the dedup_engine while others by the FP_manager and dedup_map. The figure also presents that as the number of deduplicated pages increases, the overhead of the dedup_map increases as well. For the comparison purpose, we have executed the same deduplication software on Intel i7 CPU where the average processing overhead is measured as 11.5 μs.

By comparing the deduplication overhead with the elapsed time for writing a 4KB page in flash memory, we can estimate the minimum deduplication ratio to obtain performance gains by applying deduplication. In general, the elapsed time is 500~2000 μs in MLC flash [5]. So, if we assume it as 1200 μs, the minimum deduplication ratio becomes 10%. The figure also shows that when we utilize a hardware accelerator for the dedup_engine, we can drop the deduplication overhead significantly. In addition, we can eliminate the dedup_map overhead, which become serious as the number of deduplicated pages increases, by utilizing FTL's mapping table discussed in the next paragraph.

The final issue is how to integrate deduplication into FTL in SSDs. FTL is a software layer that takes charge of address translation, garbage collection, wear-leveling and bad block handling. When we integrate the deduplication software into FTL, we can exploit several cross-layer optimizations. For example, we can make use of FTL's mapping table as the dedup_map, so that we can remove the dedup_map overhead completely. Another example is that, the duplication information can be utilized to differentiate hot/cold pages for enhancing garbage collection and wear-leveling performance. In other words, incorporating the deduplication software into FTL seamlessly might not only provide the reduction of WAF but also enhance the performance of FTL.

Reference

[1] M. Balakrishnan, A. Kadav, V. Prabhakarani and D. Malkhi, "Differential RAID: Rethinking RAID for SSD Reliability", In Proceeding of the Eurosys 2010 Conference, 2010.

[2] V. Prabhakaran, M. Balakrishnan, J. Davis and T. Wobber, "Depletable Storage Systems", In Hot Topics in Storage (HotStorage'09), 2009.

[3] S. Quinlan and S. Dorward, "Venti: A New Approach to Archival Storage", In Proceeding of the USENIX Conference on file and Storage Technologies (FAST'02), 2002.

[4] A. Clements, I. Ahmad, M. Vilayannur and J. Li, "Decentralized Deduplication in SAN Cluster File Systems", In Proceeding of the USENIX Annual Technical Conference (ATC'09), 2009.

[5] B. Debnath, S. Sengupta and Jin Li. "ChunkStash: Speeding Up Inline Storage Deduplication using Flash Memory", In Proceeding of the USENIX Annual Technical Conference (ATC'10), 2010.

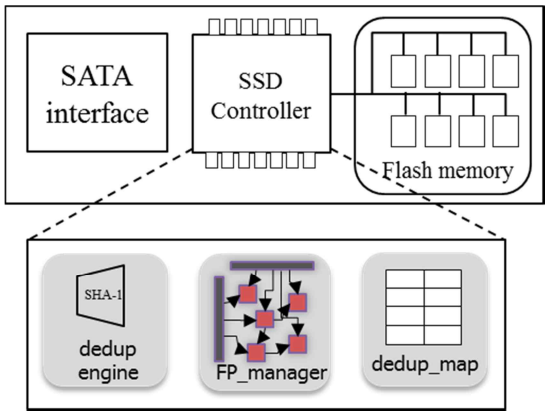

Fig. 1 Deduplication software architecture in SSDs

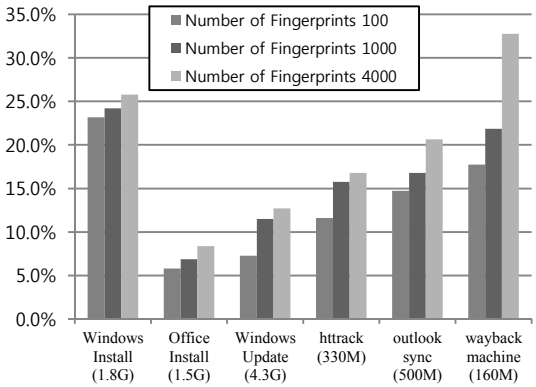

Fig. 2 Deduplication ratio for SSD workloads with various number of fingerprints

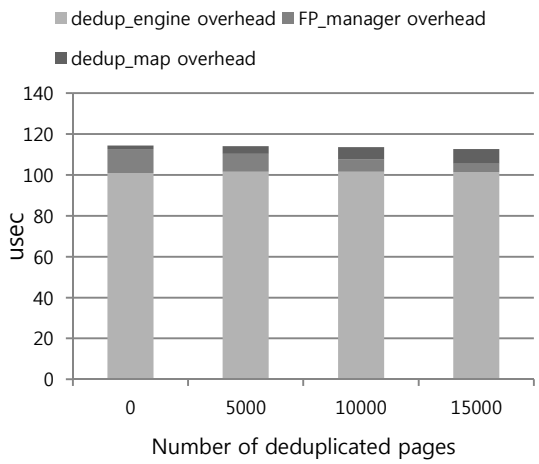

Fig. 3 Deduplication overhead analysis in SSDs