MAST: Multi-Level Associated Sector Translation for NAND Flash Memory-Based Storage System

Jeehong Kim, Dong Hyun Kang, Byungmin Ha, Hyunjin Cho, and Young Ik Eom

Sungkyunkwan University, Republic of Korea
{jjilong,kkangsu,chaosken,hjcho,yieom}@skku.edu

Abstract. NAND flash memories have been widely adopted as storage on computing devices, since it has strong effectiveness on performance. However, asymmetric operation performance of NAND flash memories still impedes high performance. To mitigate this performance hurdle, most storage devices of NAND flash memories have used Flash Translation Layer (FTL). In this paper, we propose a novel workload-aware FTL mapping policy, called Multi-level Associated Sector Translation (MAST). MAST divides log block depending on characteristics of data, and dynamically detects hot pages to avoid unnecessary merge operations of FTL. In experimental results, we show that our approach outperforms FAST method by 10% and LAST by 3%, respectively.

Keywords: NAND flash memory, Flash translation layer, Address translation, Associative mapping.

1 Introduction

Recently, NAND flash memories have been widely used as storage on computer environments, such as desktop, server, and smart device. Compared to HDDs with mechanical part, they provide low access latencies, shock resistance, and uniform random access speed [1]. However, NAND flash memories have asymmetric read and write performance feature whose read operations are faster than write operations [2]. The major reason of asymmetric property is that NAND flash memory blocks have to be erased before they are written again because they do not allow in-place update. In addition, erase operations take about 100 times longer than write operations. To mitigate this performance hurdle, most storage devices, which consist of NAND flash memories, have used Flash Translation Layer (FTL). The FTL is located in between host and NAND flash memories and maps logical addresses to physical flash addresses. To improve performance of NAND flash memory, most researches have focused on FTL mapping policies which can be classified into three categories: block-level mapping, page-level mapping, and hybrid mapping [3]. First, in the block-level mapping, logical page address must be first translated into a physical block and then it is stored into fixed page based on offset value. Although it keeps small mapping table on DRAM, it causes high garbage collection overhead for copying valid pages. In the page-level mapping, logical page address can be mapped into any page with in

James J. (Jong Hyuk) Park et al. (eds.), *Computer Science and Its Applications*,
Lecture Notes in Electrical Engineering 330, DOI: 10.1007/978-3-662-45402-2_116

NAND flash memory. Therefore, page-level FTL improves performance because it reduces garbage collection overhead. However, it maintains large size mapping table on DRAM [4-6]. In the hybrid mapping, it partition flash blocks into data blocks and log blocks to adopt advantages of previous mapping policies such as small mapping table and low garbage cost. The data blocks are managed at the granularity of block (i.e., block-level mapping). On the other hands, the log blocks are managed at the granularity of page (i.e., page-level mapping). However, most hybrid mapping FTLs ignore the temporal locality [7-14]. Lee et al. proposed a LAST, which adopts hybrid mapping policy and considers the temporal locality by distinguishing pages into hot page and cold page [10]. Im et al. introduce ComboFTL, which manages a small SLC region for hot data and a large MLC region for cold data by exploiting the SLC mode of MLC flash memory [11]. However, the LAST can significantly induce the memory overheads because it must save recency of each page and maintain them to classify pages.

In this paper, we propose a workload-aware FTL mapping policy, called MAST, to mitigate asymmetric read and write performance. The MAST also adopts the hybrid mapping approach to improve performance of NAND flash memory and it efficiently classifies pages into hot page and cold page without any memory overheads. The key idea of MAST dynamically detects hot pages to avoid unnecessary merge operations of FTL. The main contributions of this paper are follows:

- We introduce a workload-aware FTL mapping policy, which adopts the hybrid mapping approach.
- Depending on data update frequency, MAST classifies pages into hot page and cold page without any external information and efficiently manages pages on attributed area in log block.
- To reduce merge cost of FTL, MAST first find a victim block, which has the maximum number of valid pages as well as the minimum associated data blocks, in separated area for cold pages and then performs merge operations in the idle time of NAND flash memory.
- In evaluation, we show that our approach outperforms existing FTL mapping techniques

2 MAST: Multi-Level Associated Sector Translator for NAND Flash Memory

Fig.1 shows the architecture of the MAST. The MAST classifies a log block into three log areas based on data properties: *Locality detection Area (LA), Hot Area (HA), and Cold Area* (*CA*). The LA is composed of small log blocks and used to detect *hot pages*. The detected *hot pages* are stored in the HA. The *cold pages* are stored in CA. Finally, LA and HA manage their pages based on Least Recently Used (LRU) algorithm. More details of MAST are described in the remainder of this section.

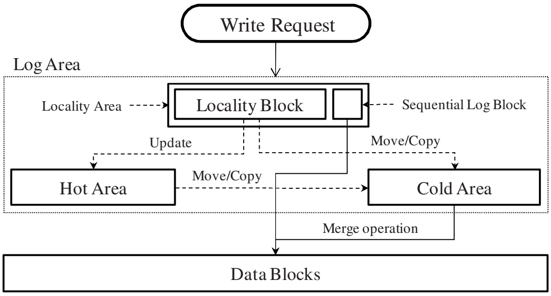

Fig. 1. The architecture of MAST

2.1 Locality Area

The LA is divided into sequential log blocks and random log blocks to separate the sequential request from the random request. Classifying sequential requests are very important to reduce merge cost because sequential requests positively impacts performance and then it can lead to lower cost with switch merge. To effectively achieve this separation, we follow the basic rules of FAST [8], and make separated sequential requests be stored in sequential log blocks. When a new write request is issued from host, we checks whether the requested page is hot page or not by scanning pages in LA. If the requested page is stored, we consider it as hot page and then we send hot page to HA because it will update frequently within a very short period. If there is no free space in LA, to create the space when the stored page in LA is requested during next scan, we consider the page as cold page and then transfer a page from LA to CA. To minimize the copying overhead for valid pages, we also migrate a block in LA to CA, when the number of valid pages in a block of LA is larger than threshold.

2.2 Hot Area and Cold Area

The HA maintains hot pages to reduce unnecessary merge operation. However, hot pages can be dynamically changed cold pages according to workload. Therefore, we check a recency of page based on LRU algorithm when a page is modified by a write request. If there is no free space in HA, we migrate a valid page to CA because it is the most unlikely to be updated in the future.

In normal, the CA keeps almost valid page as the cold page. Therefore, the rate of valid pages in block is higher than other area and it means that the associativity degree of a log block can be increased. To reduce the associativity degree of a log block, we select a block that has the maximum number of valid pages as well as the minimum associated data blocks as a victim block. As a result, MAST significantly reduces merge cost when the victim selection is triggered.

3 Dynamic Area Size Optimization

The performance of the MAST is determined by the size of LA and HA because it suffers from frequently copy and merge overheads due to eviction operation. Large LA and HA size will place cold pages in their area instead of CA because eviction operation of LA and HA does not occur frequently and classifying hot pages are not easy. On the other hand, small LA and HA size will place hot pages in CA due to frequently eviction operations. As a result, unnecessary merge operations are increased according to workloads. Therefore, MAST decides the size of LA and HA needs properly depending on their utilization. If the number of log blocks in LA is small, it refers that hot data frequently moves to the CA. Thus, if the CA's utilization is less than a threshold value or gradually decreasing, it is necessary to diminish the size of CA and increase that of LA. Otherwise, if too many blocks is in the LA, cold data are frequently recorded in the HA, and the utilization of the HA becomes higher. Therefore, if the utilization of the HA is more than a threshold value and gradually growing, it is needed to reduce the size of LA and increase that of CA.

4 Experimental Result

For evaluating MAST, we measured the system call trace to collect properties of workloads in PC environment. In evaluation, we use simulator for NAND flash memory, whose specifications are 4GB capacity, 128KB block size, 2KB page size and 256 log blocks.

4.1 Utilization Analysis

Fig. 2 shows the utilization of each log block area in three micro applications. The utilization of application represents the rate of valid pages among all pages in each area. As shown Fig. 2, the utilization of the HA is lower and that of the CA is higher in application. It represent that hot and cold data are divided well and sent to a right area in accordance with goal of MAST. Utilization of application 2 and 3 are similar with application 1.

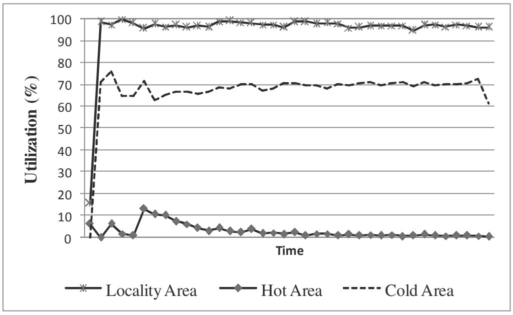

Fig. 2. Utilization by each area

4.2 Merge Operation Analysis

Fig. 3 shows the ratio of each types of merge operation. In Fig. 3, we observed that the FAST [7] method had the largest percentage of the full merge and the lowest of the dead merge. It represents that when a merge operation was occurred, estimation cost was large. LAST [10] technique had higher rates of the switch merge and the dead merge than the FAST. However, the rate of the full merge was still higher and that of the partial merge was lower. In comparison with these, MAST had the lowest rate of full merge operation and the highest rate of dead merge operation. It represents that the independent management focusing on hot data had done well.

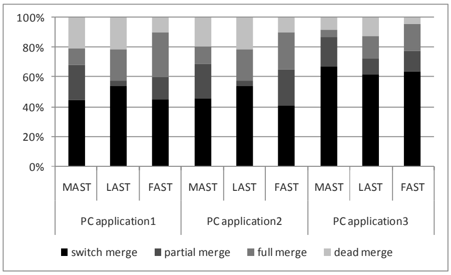

Fig. 3. Merge operation ratio in comparison of existing techniques

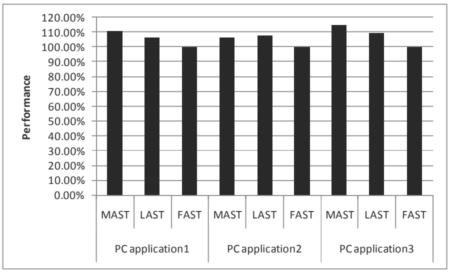

Fig. 4. Efficiency in comparison of existing techniques

4.3 Efficiency Analysis

Fig. 4 shows the efficiency of MAST. In application 1 of Fig. 4, MAST outperformed FAST technique by 7% and LAST by 3%, respectively. We observed that it is due to ratio of the full merge operation and dead merge operation in Fig. 3. In application 2, MAST performed 5% more efficient that FAST, whereas performed 2% lower than LAST. This lower performance is within the margin of error. Lastly, in application 3, MAST outperformed FAST by 12% and LAST by 4%, respectively.

The MAST shows a similar efficiency to the LAST. However, it is strength that MAST requires little external information with even similar capacity to LAST.

5 Conclusion

In order to improve the performance in NAND flash memories which has asymmetric performance properties, mapping policy technique in FTL have been researched. We propose a workload-aware FTL mapping policy, called MAST, to mitigate asymmetric read and write performance. MAST divides log block depending on characteristics of data, then dynamically detects hot pages to avoid unnecessary merge operations and manages hot and cold pages on attributed area in log block. In experimental results, we showed that our approach outperforms FAST method by 7% and LAST by 3%, respectively.

Acknowledgement. This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Planning (2010-0020730)

References

[1] Dinkelman, T.: SSDs A Shift in Data Storage. In: Proc. of Flash Memory Summit (2008)

[2] Samsung Corporation. K9XXG08XXM Flash Memory Specification, http://www.samsung.com

[3] Chung, T.-S., Park, D.-J., Park, S., Lee, D.-H., Lee, S.-W., Song, H.-J.: System software for flash memory: A survey. In: Sha, E., Han, S.-K., Xu, C.-Z., Kim, M.-H., Yang, L.T., Xiao, B. (eds.) EUC 2006. LNCS, vol. 4096, pp. 394–404. Springer, Heidelberg (2006)

[4] Wu, M., Zwaenepoel, W.: ENVy: a non-volatile, main memory storage system. In: Proc. of Architectural Support for Programming Languages and Operating Systems, pp. 86–97 (1994)

[5] Kim, H., Lee, S.: A new flash memory management for flash storage system. In: Proc. of Computer Software and Applications Conference, pp. 284–289 (1999)

[6] Chiang, M.L., Lee, P.C.H., Chang, R.C.: Cleaning policies in mobile computers using flash memory. Journal of Systems and Software, 213–231 (1999)

[7] Kim, J., Kim, J.M., Noh, S.H., Min, S.L., Cho, Y.: A space efficient flash translation layer for compact flash systems. IEEE Transactions on Consumer Electronics, 366–375 (2002)

[8] Lee, S.W., Park, D.J., Chung, T.S., Lee, D.H., Park, S., Song, H.J.: A log buffer-based flash translation layer using fully-associative sector translation. ACM Transactions on Embedded Computing Systems, 6:1–6:27 (2007)

[9] Park, C., Cheon, W., Kang, J., Roh, K., Cho, W., Kim, J.S.: A reconfigurable FTL (Flash Translation Layer) architecture for NAND flash based applications. ACM Transactions on Embedded Computing Systems 38, 1–38 (2008)

[10] Lee, S., Shin, D., Kim, Y.J., Kim, J.: LAST: locality-aware sector translation for NAND flash memory-based storage systems. ACM SIGOPS Operating Systems Reviw, 36–42 (2008)

[11] Im, S., Shin, D.: ComboFTL, Improving performance and lifespan of MLC flash memory using SLC flash buffer. Journal of Systems Architecture, 641–653 (2010)

[12] Wei, Q., Gong, B., Suraj, P., Bharadwa, V., Zeng, L., Okada, K.: WAFTL: A workload adaptive flash translation layer with data partition. In: Proc. of Mass Storage Systems and Technologies (2011)

[13] Kwon, S.J., Cho, H.J., Chung, T.S.: Hybrid Associateve Flash Translation Layer for the Performance Optimization of Chip-Level Parallel Flash Memory. ACM Transactions on Storage, 13:1–13:24 (2013)