Hybrid Write Buffer Algorithm for Improving Performance and Endurance of Nand Flash Storages

Se Jun Han†‡, Dong Hyun Kang†, Young Ik Eom†
†Sungkyunkwan University, Korea ‡Samsung Electronics, Korea
{sejun.han,kkangsu,yieom}@skku.edu

***Abstract*—This paper proposes a novel buffering algorithm for write buffer, comprised of DRAM and NVM, in NAND flash storages of the home cloud server is proposed. As adopting dirty first migration and clean page caching technique, the algorithm may improve performances and lifetime of storage system of the home cloud server. The experimental results show that the algorithm improved buffer hit ratio by up to 116.51% and reduced the number of erase operations on NAND flash storage by up to 48.67%.**

I. Introduction

Nowadays, most of the home devices are normally employing a small capacity of storage hardware (e.g., eMMC storage) because the home device only needs a small amount of storage space to store their own data. However, as greeting an IoT (Internet-of-Things) era, recent trends predict that some home devices may act a role as a home cloud server [1], [2]. Since the home cloud servers are commonly adopting a virtualization technology for executing multiple systems on a single hardware, the large-scale storage, such as Solid State Drive (SSD), is more suitable for storage devices of the home cloud servers.

SSDs are mainly employed in various systems (e.g., personal computing, enterprise, and data-center environment) as a main storage device because of their advantages: low power consumption, rapid I/O responses, and shock resistance. However, SSDs are needed to mitigate some weaknesses such as impossible in-place-update, limited block erase counts, and asymmetric read/write speed. To alleviate these weaknesses, there have been several studies on the write buffer algorithm for SSD internal write buffer.

One of the most up-to-date write buffer algorithm, CBM [3], is employing not only DRAM but also Non-Volatile memory (NVM). During the last decade, NVM has been studied in several areas for its benefits such as byte-addressability, rapid read latency, and non-volatility. In addition, NVM can be placed side-by-side with DRAM as a write buffer of SSDs for improving performances. As adopting NVM, CBM [3] shows more improved performance than previous write buffer algorithms. However, CBM may suffer from unnecessary write requests caused by the clean page eviction. This paper proposes a novel hybrid write buffer algorithm that can improve the performance and reduce the number of erase operations of SSDs, by dirty first migration and write pattern reshaping. The approach can mitigate the weaknesses of the SSD-based storage systems, and improve the overall performance of the home devices.

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No.R0126-15-1082, Management of Developing ICBMS(IoT, Cloud, Bigdata, Mobile, Security) Core Technologies and Development of Exascale Cloud Storage Technology). Young Ik Eom is the corresponding author of this paper.

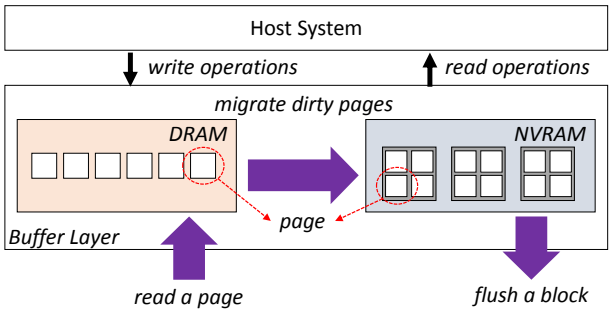


Fig. 1: Overview of the hierarchical write buffer architecture

II. Design of Hierarchical Write Buffer

The goal of the proposed algorithm is improving the performance and extending the endurance of SSDs. To achieve the goal, the algorithm partitions the write buffer into two parts and extend well-known CLOCK algorithm for efficient management of pages in the write buffer of SSD (Figure 1). All pages of DRAM are maintained in a page granularity to maximize the temporal locality. On the other hands, all pages of NVM are maintained in a block granularity to reshape random write patterns to sequential ones.

A. Page mangement on Write Buffer

Since flash memory has the weaknesses for the random page write requests, most of write buffer algorithms store dirty pages on the write buffer for reshaping write patterns and reducing write requests issued to the flash memory. However, the algorithm stores both clean and dirty pages on DRAM of the write buffer because maintaining clean pages on the write buffer helps to support a short response time in case of the mixed read-write workloads. In order to maximize the space utilization of DRAM, the algorithm maintains all pages on DRAM at a page-level granularity and follows the basic rules of the CLOCK algorithm with some modifications. Unlike traditional CLOCK algorithm, if a page on DRAM is re-accessed with a read operation, the algorithm sets the reference bit of the page. On the other hand, if the page is re-accessed with a write operation, the algorithm does not set the reference bit of the page to maintain clean pages on DRAM as long as possible – called dirty first migration.

When DRAM is full, the algorithm selects a victim page on DRAM by scanning pages until a page with reference bit of zero is found. If the selected victim page is clean, the page is discarded since read operation on flash memory is much faster than write operation. Otherwise, the page is migrated to NVM to transform write requests to flash memory into sequential write patterns.

All pages of NVM also follow the basic rules of the CLOCK algorithm with some modifications. In order to issue write

978-1-4673-8364-6/16/$31.00 ©2016 IEEE

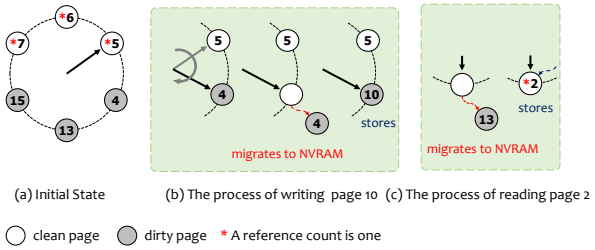


Fig. 2: An example of CLOCK on DRAM operations, with maximum 6 pages in DRAM

requests sequentially, the algorithm groups pages that belong to the same erase block of flash memory into logical blocks and moves the hand of CLOCK at block-level granularity instead of page-level one (i.e., each logical block has a reference bit).

B. Example

Figure 2 illustrates an example of page replacements on DRAM which show how CLOCK handles events (a) - (c) from the initial status of Figure 2a. In initial status of Figure 2a, pages 5, 6, and 7 are clean pages and page 4, 13, and 15 are dirty pages. The hand of CLOCK on DRAM moves in the clock-wise direction and is pointing to page 5 now.

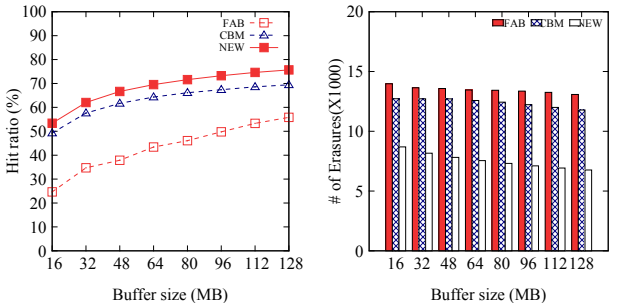
When page 10 is accessed for writing in Figure 2a, a cache miss occurs. In addition, since the DRAM is full, CLOCK on DRAM starts to scan to find a victim page by using its hand. In this example, page 4 is selected as a victim page and the page is migrated to NVM (Figure 2b) for incoming page. For *dirty first migration*, the algorithm does not set its reference bit because the page 10 is a dirty. After then, the hand of CLOCK on DRAM is moved to the next page. Now, when page 2 is accessed for reading, another cache miss occurs (Figure 2c). The hand of CLOCK on DRAM is also traversed to find a victim page, and then page 13 is migrated to NVM before replacing the page with the incoming page. Unlike the previous cache miss, the algorithm sets its reference bit since the page 2 is a clean page (Figure 2c).

III. Performance Evaluation

To verify a proposed algorithm, a write buffer simulator based on DiskSim [4] with SSD model patch [5] was implemented. In order to compare with the state-of-the-art write buffer algorithms, the simulator was configured with the same parameters of CBM [3] except DRAM/NVM ratio in the write buffer because CBM never limits the amount of DRAM. During the evaluation, size of the NVM was configured to 10% of the total buffer size. On the other hand, FAB [6] has been configured with a DRAM-only write buffer because it was designed for DRAM-based write buffer. In order to study the impact on the size of write buffer, the buffer simulator was ran by varying the size of write buffer from 16MB to 128MB with OTLP trace [7].

Figure 3 shows that the algorithm outperforms CBM and FAB in terms of hit ratio and the number of erase operations under the completely read-dominant environment.

The algorithm improves buffer hit ratio by up to 116.51% and 8.97% compared to FAB and CBM, respectively (Figure 3a). Unlike other algorithms, FAB shows the lowest hit ratio by two reasons. First, financial trace is comprised of small portion of



(a) Write buffer hit ratio (b) The number of erase operation

Fig. 3: Financial Trace

hot pages and large portion of cold pages as it is an OLTP workload. Second, FAB, which uses a block-level algorithm, may not keep hot pages as long as possible because of *early eviction problem* [8] – the phenomenon that blocks, which has small number of hot pages, are evicted and re-accessed in the near future repeatedly.

Figure 3b shows the number of erase operations. The experimental results clearly show that the algorithm may reduce the number of erase operations of NAND flash storage device by up to 48.67% and 42.53% compared to FAB and CBM, respectively.

IV. Conclusion

This paper proposes a *hybrid write buffer architecture* that consists of DRAM and NVM for improving storage performance and extending the lifetime of NAND-based storage devices. In order to achieve the goal, NVM is placed side-by-side with DRAM as a write buffer of SSDs, and effectively migrate dirty pages from DRAM to NVM to reshape random write patterns with dirty first migration. For performance evaluation, the write buffer simulator was implemented based on Disksim. Also, experiments were performed with OLTP trace. The experimental results clearly show that proposed algorithm efficiently improves the write buffer hit ratio by up to 116.51% and reduces the number of erase operations by up to 48.67%. As a result, proposed algorithm can enhance the performance and the lifetime of SSDs with small size NVM.

References

[1] H. Park, I. Lee, T. Hwang, and N. Kim, "Architecture of home gateway for device collaboration in extended home space," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 4, pp. 1692–1697, 2008.

[2] M. R. Cabrer, R. P. D. Redondo, F. Vilas, J. J. P. Arias, and J. G. Duque, "Controlling the smart home from tv," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 2, pp. 421–429, 2006.

[3] Q. Wei, C. Chen, and J. Yang, "CBM: A cooperative buffer management for ssd," in *IEEE 30th Symposium on Mass Storage Systems and Technologies (MSST)(2014)*.

[4] J. S. Bucy, J. Schindler, S. W. Schlosser, and G. R. Ganger, "The disksim simulation environment version 4.0 reference manual (cmu-pdl-08-101)," *Parallel Data Laboratory*, p. 26, 2008.

[5] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. S. Manasse, and R. Panigrahy, "Design tradeoffs for ssd performance." in *USENIX Annual Technical Conference*, 2008, pp. 57–70.

[6] H. Jo, J.-U. Kang, S.-Y. Park, J.-S. Kim, and J. Lee, "FAB: Flash-aware buffer management policy for portable media players," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 2, pp. 485–493, 2006.

[7] "OLTP Trace from UMass Trace Repository," http://traces.cs.umass.edu/index.php/Storage/Storage.

[8] G. Wu, X. He, and B. Eckart, "An adaptive write buffer management scheme for flash-based ssds," *ACM Transactions on Storage (TOS)*, vol. 8, no. 1, p. 1, 2012.

978-1-4673-8364-6/16/$31.00 ©2016 IEEE