A Read-while-write-based Out-of-order Scheduling for High Performance NAND Flash-based Storage Devices

Jin-Young Kim, Sang-Hoon Park, Hyeokjun Seo, Taehee You and Eui-Young Chung
School of Electrical and Electronic Engineering, Yonsei University, Seoul, Republic of Korea
{jy0615.kim, soskhong, jjsky7, xoqhd1212}@dtl.yonsei.ac.kr, eychung@yonsei.ac.kr

***Abstract*—Recently, Phase-change RAMs (PRAMs) are considered to be as a good candidate that can substitute as DRAM cache buffers (CBs) in NAND flash-based storage devices (NFSDs). PRAM is an adequate device to mobile consumer electronics thanks to low static power consumption, high density and non-volatility. However, in spite of many advantages over DRAM, asymmetric write/read speed of PRAM causes performance degradation in NFSD. In this paper, hence, we first propose a novel scheduling method for an NFSD with PRAM CB utilizing read-while-write (RWW) of PRAM. The method schedules NFSD's requests to service read and write simultaneously using RWW. In the experiment result, the proposed method reduces read and write latency on average by 30% and 19%, respectively.**

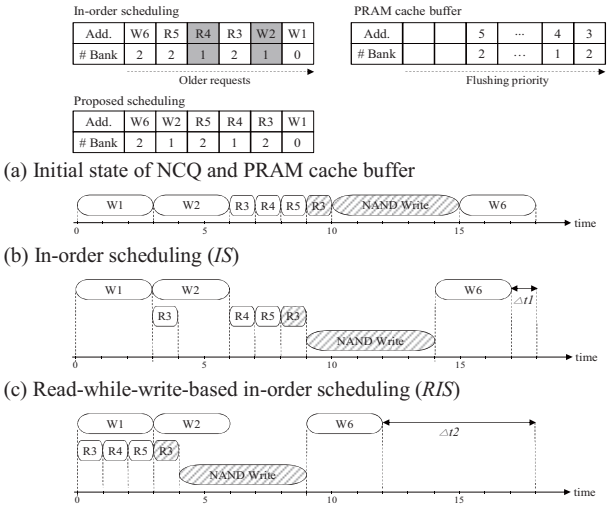
Keywords—NAND flash memory, PRAM cache buffer, read-while-write

I. INTRODUCTION

NAND flash-based storage devices (NFSDs) have been widely employed along with the growth of mobile device market. Modern NFSDs utilize DRAM as cache buffer (CB) to complement various drawbacks of NAND flash memory (NFM) such as long latency and necessity of out-of-place update. However, the use of DRAM has become a burden to NFSDs due to its cost and volatility.

Recently, Phase-change RAMs (PRAMs) [1] have emerged as a good candidate that can substitute DRAMs since they are denser than DRAMs and faster than NFMs. Additionally, non-volatility property of PRAMs clearly motivates researchers to utilize PRAMs as CBs in NFSDs for better reliability [2].

However, in spite of many advantages over DRAM, asymmetric write/read speed of PRAM is still an issue dealt by many works [3], because slow write affects not only write itself but also read. In this paper, hence, we propose a novel scheduling method for NFSDs with a PRAM CB utilizing read-while-write [4] (RWW) of PRAM to minimize performance degradation incurred by long write latency. For the best of our knowledge, this is the first work utilizing RWW for PRAM CB. More specifically, the proposed method schedules NFSD's requests to service read and write simultaneously with multiple banks of PRAM using RWW. As a result, both read and write are improved because long write latency is hidden by many read requests serviced in parallel.



(a) Initial state of NCQ and PRAM cache buffer

(b) In-order scheduling (*IS*)

(c) Read-while-write-based in-order scheduling (*RIS*)

(d) Read-while-write-based out-of-order scheduling (*ROS*)

Fig. 1 Comparison of total time according to scheduling methods

II. PROPOSED METHOD

In NFSD with Serial-ATA interface, a request given from a host is first queued in the native command queuing (NCQ) and is transferred to CB after scheduling. In-order scheduling (*IS*) and out-of-order scheduling (*OS*) are to service incoming requests considering their arrival times and their data characteristics, respectively [5]. We investigate this part to minimize the performance degradation by long write latency of PRAM.

RWW is a unique feature of PRAM which improves throughput of PRAM by handling read and write in parallel using independent banks within a PRAM. We propose a novel scheduling method that utilizes RWW of PRAM, which is called a read-while-write-based out-of-order scheduling (*ROS*).

First, *ROS* reorders requests in the NCQ to handle as many read as possible while ongoing write is serviced in parallel. For this purpose, read which is the latest inserted to the NCQ can be serviced first if the bank for read is not occupied for write at the same time. Such scheduling is activated only when read can be completely hidden by RWW. Accordingly, *ROS* can improve read performance without write performance degradation.

Second, *ROS* also improves write latency by considering data flushing of CB which eventually occurs due to its limited

capacity. The proposed method handles read for data flushing and write for host requests simultaneously using RWW based on the information of CB status. As a result, data flushing, which is an overhead for write latency, can be partially hidden by write operations of PRAM, thus write performance of NFSDs can be improved.

Examples of Fig. 1 demonstrate the effectiveness of the proposed method. Fig. 1 (a) shows the initial states of the NCQ and PRAM CB, and Fig. 1 (b), (c) and (d) are scheduled by in-order scheduling (*IS*), read-while-write-based in-order and out-of-order scheduling (*RIS* and *ROS*), respectively. To evaluate the performance in a simple way, write/read latency of PRAM and program latency of NFM are assumed to be 3, 1 and 5, respectively. As a baseline, *IS* of Fig. 1 (b) processes requests W1, W2, R3, R4, R5 in order and W6 is processed after flushing data of address 3 from PRAM CB, whose total time is 18. Data flushing of CB is managed by the least recently used (LRU) policy. *RIS* of Fig. 1 (c) is similar to *IS* but shows reduced latency by the factor of $\Delta t1$ due to parallel operation of W2 and R3 by RWW. However, *IS* is not able to improve the latency further since R4 after R3 requires the same bank with W2, thus the total time is 17. On the other hand, *ROS* of Fig. 1 (d) services R3/R4/R5 earlier than W2 and flushes data from CB before it is requested. As a result, *ROS* reduces the total time of NFSD to 12 thanks to the effective utilization of RWW.

III. EXPERIMENTS

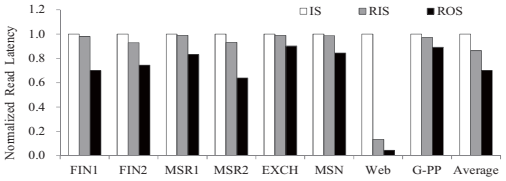
To evaluate the effectiveness of the proposed method, we implemented a trace-driven simulator which consists of an FTL, a specification of PRAM [6] and NFM [7] and the configurations of storage system. Various workloads are collected from [8], [9] and by DiskMon [10] during daily PC use. We assume that the capacity of PRAM and NFM is 16MB and 16GB, respectively, and the size of the NCQ is set to 32.

Fig. 2 shows performance of the proposed method and latency is normalized by that of *IS*. In the perspective of both read and write latency, proposed *ROS* outperforms *IS* and *RIS*. As shown by *RIS*, the effect of RWW is negligible except Web consisting of 98% read requests. However, in Fig. 2 (a) and (b), *ROS* which utilizes RWW for request reordering and data flushing of CB improves read and write latency 30% and 19%, on average.

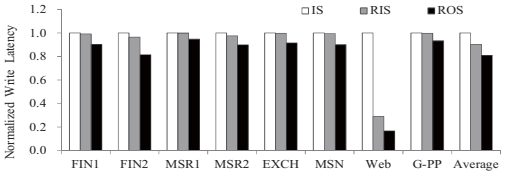
Fig. 2 (c) shows Hidden Read Ratio (HRR), which is newly defined to analyze total PRAM read latency hidden by RWW. The larger HRR means that the more PRAM read operations are hidden by PRAM write operations. On average, HRR of *ROS* is 95% except for Web, which means that most of read from host and read for data flushing of PRAM CB are hidden by write requests.

IV. CONCLUSION

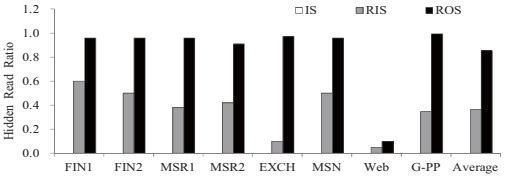
In this paper, we proposed a novel scheduling method for NFSDs with PRAM CB that utilizes RWW of PRAM. The proposed method schedules NFSD's requests to service read and write simultaneously using RWW. By utilizing RWW for requests reordering and data flushing from CB, the proposed method hides most of read time while write is in progress. In the experiment result, the proposed method reduces read and write latency by 30% and 19%, on average.



(a) Normalized read latency



(b) Normalized write latency



(c) Hidden read ratio

Fig. 2 Performance of proposed method

ACKNOWLEDGMENT

This work (Grant No. C0146555) was supported by Business for Cooperative R&D between Industry, Academy, and Research Institute funded Korea Small and Medium Business Administration in 2013, by IDEC (IC Design Education Center) and by Samsung Electronics.

REFERENCES

[1] ITRS, "International technology roadmap for semiconductors," Semiconductor Industry Association, Tech. Rep, 2009.

[2] J. K. Kim, H. G. Lee, S. Choi, and K. I. Bahng, "A pram and nand flash hybrid architecture for high-performance embedded storage subsystems," in *Proceedings of the 8th ACM international conference on Embedded software*, pp. 31–40, 2008.

[3] M. K. Qureshi, M. M. Franceschini, and L. A. Lastras-Monta˜no, "Improving read performance of phase change memories via write cancellation and write pausing," in *High Performance Computer Architecture (HPCA)*, 2010.

[4] K.-J. Lee, B.-H. Cho, W.-Y. Cho, S. Kang, B.-G. Choi, H.-R. Oh, et al., "A 90 nm 1.8v 512 Mb diode-switch pram with 266 MB/s read throughput," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 150–162, 2008.

[5] S. S. Hahn, S. Lee, and J. Kim, "SOS: Software-based out-of-order scheduling for high-performance nand flash-based ssds," in *Mass Storage Systems and Technologies (MSST)*, pp. 1–5, 2013.

[6] Y. Choi, I. Song, M.-H. Park, H. Chung, S. Chang, B. Cho, et al., "A 20nm 1.8 v 8Gb pram with 40MB/s program bandwidth," in *IEEE Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 46–48, 2012.

[7] Micron Technology Inc., NAND Flash Memory Datasheet, MT29FXXG08AXXXX, 2009.

[8] UMass Trace Repository, http://traces.cs.umass.edu/, June 2007.

[9] Storage Networking Industry Association, http://iotta.snia.org/, 2011.

[10] DiskMon, http://technet.microsoft.com/enus/sysinternals/bb896646, 2010.