Exploiting the Future Reference in Write Buffer Management Design for SSDs

Seongmin Kim and Taeseok Kim

Abstract—**We present some practical issues in designing the write buffer management scheme for SSDs (Solid-State Drives). In SSDs with SATA interface, there exist the pending I/O commands in NCQ (Native Command Queueing), and it can be effectively exploited for write buffer management. Based on this fact, we propose an efficient write buffer management scheme that exploits the future buffer reference pattern from I/O commands queued in NCQ. We also discuss the relationship between the buffer size and the effectiveness of NCQ-awareness in our scheme. Finally, we show that the proposed scheme improves the performance of write buffer for some workloads.**

Index Terms—**NAND flash memory, NCQ, write buffer, SSDs.**

I. Introduction

SSDs are modern storage devices that consist of an array of NAND flash memories. Unlike hard disk drives, SSDs have many advantages such as short data access latency, low power consumption, and resistance to shock. However, the SSD has some shortcomings due to the physical characteristics of NAND flash memory such as poor write performance and limited number of erase operations. For the reason, many commercial SSDs employ the write buffer to overcome these problems.

The write buffer in SSD reduces the amount of write data to be practically flushed to the NAND flash memories. It also reduces the erase operation for improving lifespan of the SSD. There have been many studies on write buffer management for SSD [1]-[3]. However, such studies did not fully exploit the internals of the SSDs. In SSDs with SATA interface, the I/O commands from the host through the SATA interface are queued in NCQ (Native Command Queueing). We specially focus on the NCQ, which has important information for buffer management.

In this paper, we propose an efficient management scheme for write buffer within SSD. In our scheme, I/O commands information is used for exploiting the future reference pattern. NCQ has the pending I/O commands to be serviced later, it gives an important hint for predicting the future buffer reference. Hence, in SATA SSDs with NCQ, future reference pattern as well as past reference pattern can be exploited for write buffer management design.

In section II, related works on write buffer management for SSDs and an internal architecture of SATA SSDs are presented. Section III describes the proposed write buffer management scheme, and discusses an appropriate buffer size in our scheme for obtaining the effectiveness of NCQ-awareness. Finally, Section IV and Section V present the experimental results and concluding remark, respectively.

Manuscript received February 10, 2013; revised March 12, 2013. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0010260).

The authors are with the department of computer engineering, Kwangwoon University, Seoul, 139-701, Republic of Korea (e-mail: {zinjja2, tskim}@kw.ac.kr)

II. Related Works and Background

A. Related works

Many studies on write buffer management scheme for SSDs have been conducted. FAB is a buffer management scheme for NAND flash memory [1]. The scheme manages the data in block unit and selects the largest size of page cluster as a victim to maximize the chance of the switch merge operations. BPLRU employs the page padding scheme for maintaining the consecutive pages in log block and LRU policy for selecting the victim, respectively [2]. Through the page padding, BPLRU reduces the number of full merge operations that are more expensive than switch merge operations. CLC employs dual lists to maintain hot and cold page clusters for considering the characteristics of I/O workloads [3]. When it selects the victims, temporal locality of workload and data size belonging to page clusters are considered. These algorithms use the locality of I/O workloads, the physical characteristics of NAND flash memory based SSDs, or both, but did not fully exploit the internals of the SSDs.

B. The NCQ in SATA SSDs

The NCQ is originally designed to improve the I/O performance in SATA HDDs, but it is still widely employed in SSDs [4]. The SSDs with standardized storage interfaces allow up to 32 I/O commands for queueing, while non-standard SSDs may support up to more than 128 [5]. The I/O commands in NCQ consist of logical address, size, and type information. An I/O command at the head of NCQ is dispatched when SSD is ready to serve. The dispatched I/O command is served by buffer in the SSD or may be served in the NAND flash memory via FTL (Flash Translation Layer).

Based on the I/O command handling procedure in SATA SSDs, we observed that the data belonging to the I/O commands queued in NCQ would be needed in the near future. Therefore, we can conclude that considering the NCQ in designing the write buffer for SATA SSDs may be a good strategy.

III. The Proposed Scheme for Exploiting the Future Reference

In this paper, we discuss some practical issues when

DOI: 10.7763/IJFCC.2013.V2.202

designing the write buffer management scheme for SSDs. In usual buffer management, when the buffer is full, the victim buffer is selected for replacement. Ideally, the buffer that will not be referenced for the longest of time should be selected for victim, but many buffer management schemes including LRU (Least Recently Used) have exploited only the past buffer reference pattern because the future buffer reference information cannot be accessible in real systems. However, in SATA SSDs, since the NCQ has the pending I/O commands to be serviced later, we can obtain the future buffer reference information from NCQ.

Based on this idea, we could devise a simple and effective buffer management scheme. The proposed scheme additionally exploits the information of the I/O commands in NCQ, which is not exploited by existing studies. It is important to note that the proposed scheme can be integrated with other existing write buffer management schemes without large modification. In order to focus on the effect of NCQ in designing the write buffer management scheme, we present our scheme based on simple LRU (Least Recently Used). We believe that our scheme can be trivially adjusted with any other existing schemes.

Like LRU, our scheme maintains the buffers in reference order from MRU (Most Recently Used) to LRU. When I/O command at the head of NCQ is dispatched, if data belonging to the I/O command resides in the buffer, the data moves to the MRU location. If data belonging to the I/O command dose note reside in the buffer, data from the LRU position is replaced, and the data belonging to the dispatched I/O command is inserted into the MRU position.

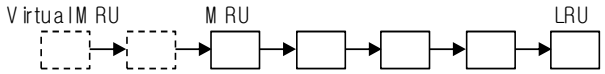


Fig. 1. The virtual MRU concept.

Unlike the existing schemes, our scheme performs an additional operation. Since data belonging to the I/O commands in NCQ will be used in the near future, they should be pinned not to be replaced. When new I/O command arrives at NCQ, our scheme moves data belonging to the I/O command to *virtual MRU* position if the data resides in buffer. The virtual MRU means that it is not actually the most recently used but should be maintained in buffer as if it is most recently used (Fig. 1). If the data belonging to the newly arrived I/O command does not reside in buffer, no action is required.

Fig. 2 illustrates the pseudo code of main function in the proposed scheme. The function check_buffer(R) is invoked when new I/O command arrives at NCQ. If data belonging to the I/O command resides in the buffer, this function moves the data to virtual MRU position using only the I/O command's address and size. This algorithm has very low complexity because it performs only the comparison operation and the pointer changes.

In what situation, does our scheme outperform the LRU? In order to analyze the effectiveness of NCQ-awareness, it is required to discuss the relationship between the buffer size and the effectiveness of NCQ-awareness. Suppose that a new I/O command, *R* arrives at NCQ, and data belonging to *R* already reside in buffer. In LRU scheme, in order to service the data from the buffer when *R* is dispatched later, the buffer size should be larger than or equal to $\sum_{i=0}^{N-1} R_i.size$, where *N* is

```
Input: R : A new I/O command
Output: NULL

// invoked when a new I/O command arrives at NCQ.
begin: check_buffer (R)
   // N_pages: the number of pages per block.
   // R.lpn : R's logical page number, R.size: R's size.
   Max_lbn = R.lpn / N_pages;
   Min_lbn = (R.lpn + R.size - 1) / N_pages;

   for each block B_i in the LRU list
      if B_i.lbn >= Min_lbn and B_i.lbn <= Max_lbn then
         // moves the block Bi to virtual MRU position
         move_to_virtualMRU(B_i);
      end
   end
end
```
Fig. 2. Algorithm of the NCQ-aware LRU scheme.

The number of commands that arrive at NCQ prior to *R*. Note that the maximum of *N* is practically 32, 64, or 128 as mentioned in Section II. The size of each request is also generally limited, which is generally hundreds of KB in Linux [6].

On the other hand, in our scheme, even though the buffer size is smaller than $\sum_{i=0}^{N-1} R_i.size$, the data belonging to *R* can be serviced from the buffer when *R* is dispatched later. It means that our scheme would show better performance than the LRU when the buffer size is limited. In the worst case that all I/O commands arrive at NCQ prior to *R* are serviced from the buffer, our scheme will show the similar performance to the LRU.

IV. Performance Evaluations

A. Experimental Environments

To show the effectiveness of the proposed scheme, we implemented a simulator based on Flashsim [7] that emulates the I/O handling in SSDs with SATA interface. We assume that the hybrid FTL such as BAST is used for address mapping and the buffer is replaced in block units [8]. In order to focus on the effect of NCQ in designing the buffer management scheme, we compared the proposed scheme with simple LRU. The workloads used in our experiments were generated by using IOmeter, which is an I/O subsystem measurement and characterization tool [9]. Table I shows the summary of workloads used in our experiments.

TABLE I: The workloads used in experiments.

	Record size	Access pattern	Avg. inter-arrival time
Workload 1	256KB	random	0.83ms
Workload 2	1MB	random	5.02ms

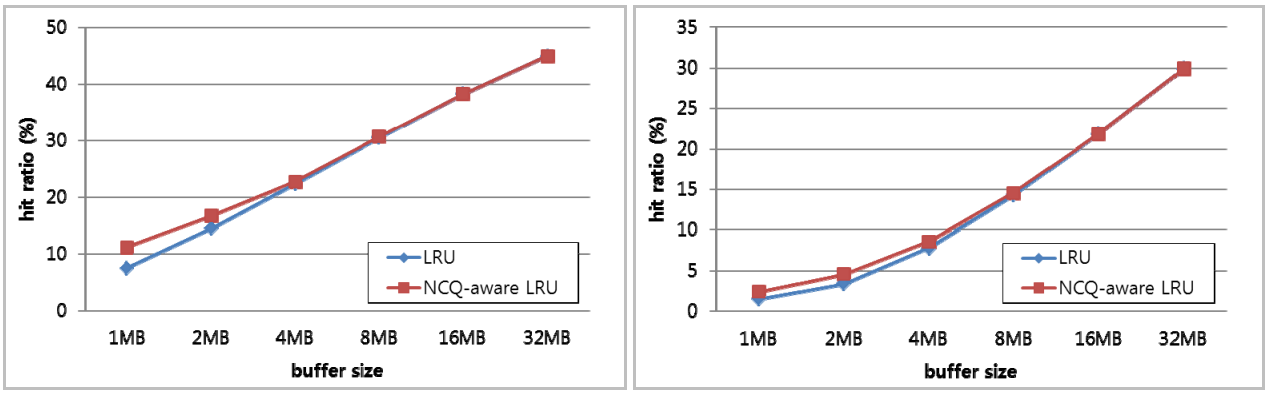
B. Results

As can be seen in Fig. 3, the proposed scheme outperforms LRU in terms of hit ratio. Especially, when the buffer size is 1MB, the proposed scheme performs better than LRU up to 4% and 2% for workload 1 and 2, respectively. As the buffer size is smaller, it seems that exploiting only the temporal locality like in LRU is not sufficient to effectively choose the victim buffer for replacement, and thus exploiting more future reference pattern by using the I/O command information in NCQ is more helpful. When buffer size is large, the performance gain that obtained by considering the NCQ is low because the queue length is short and thus there is little opportunity to exploit the future reference.

Fig. 4 shows the write bandwidth as a function of buffer size. In these results, we can see that our scheme performs better than LRU up to 5% and 2% for workload 1 and 2, respectively. Finally, Fig. 5 shows the IOPS. As can be seen, our scheme outperforms LRU up to 4% and 1%, respectively.

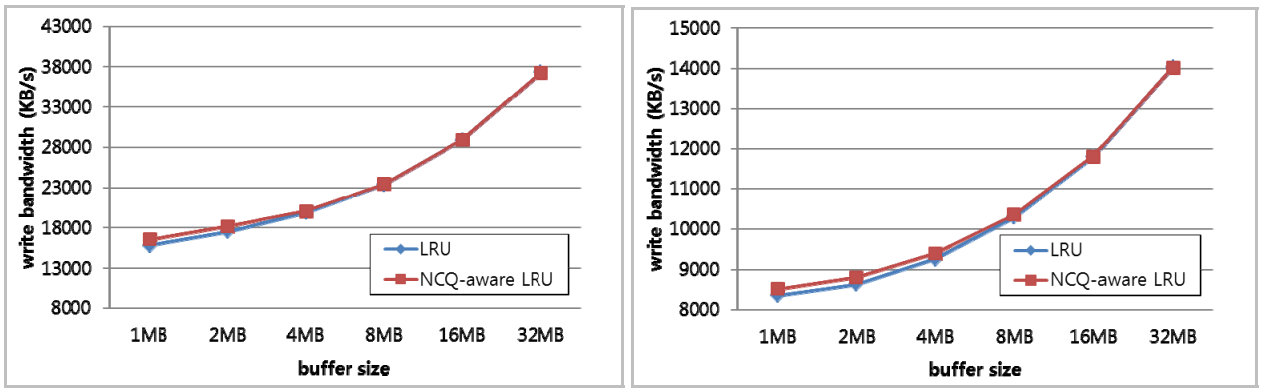
V. Conclusion

In this paper, we presented a novel write buffer management scheme, which exploits the future reference pattern. Our work has started from a careful observation of the SSD's internals. The SSDs with SATA interface have a queue called NCQ, which has many I/O commands to be



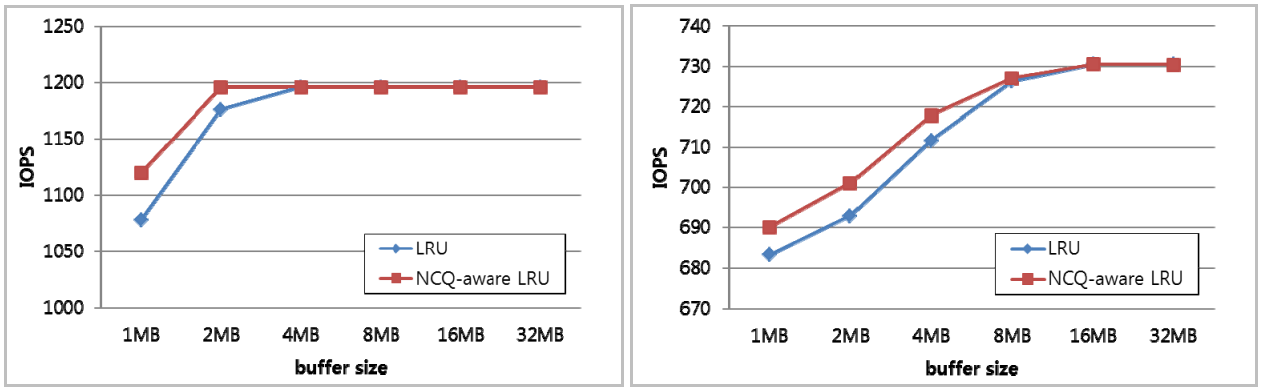
(a) workload 1 (b) workload 2

Fig. 3. Hit ratio as a function of buffer size.



(a) workload 1 (b) workload 2

Fig. 4. Write bandwidth as a function of buffer size.



(a) workload 1 (b) workload 2

Fig. 5. IOPS as a function of buffer size.

served later. Based on this observation, we could devise simple and efficient buffer management scheme that exploits the future reference pattern as well as the past reference pattern. Through the trace-driven simulations, we showed that considering the NCQ for buffer management is significantly effective in terms of hit ratio, write bandwidth, and IOPS.

REFERENCES

[1] 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 Trans. on Consumer Electronics*, vol. 52, no. 2, pp. 485-493, 2006.

[2] H. Kim and S. Ahn, "BPLRU: A Buffer Management Scheme for Improving Random Writes in Flash Storage," in *Proc. 6th USENIX Conf. on File and Storage Technologies*, San Francisco, 2008, pp. 239-252.

[3] S. Kang, S. Park, H. Jung, H. Shim, and J. Cha, "Performance trade-offs in using nvram write buffer for flash memory-based storage devices," *IEEE Trans. on Computers*, vol. 58, no. 6, pp. 744–758, 2009.

[4] G. Gasior, "Intel's X25-E Extreme solid-state drive," The technical report, 2008.

[5] E. Seppanen, M. T. O'Keefe, and D. J. Lilja, "High performance solid state storage under Linux," in *Proc. 26th IEEE Symposium on Massive Storage Systems and Technologies*, Nevada, 2010, pp. 1-12.

[6] B. Ko, Y. Kim, and T. Kim, "Performance improvement of I/O subsystems exploiting the characteristics of solid state drives," in *Proc. ICCSA'11*, Santander, Spain, 2011, pp.528-539.

[7] A. Gupta, Y. J. Kim, B. Tauras and B. Urgaonkar, "FlashSim: A Simulator For NAND Flash-based Solid-State-Drives," in *Proc. International Conference on Advances in System Simulation*, 2009, pp. 125-131.

[8] J. Kim, J. M. Kim, S. H. Hoh, S. L. Min, and Y. Cho, "A Space-Efficient Flash Translation Layer for CompactFlash Systems," *IEEE Trans. on Consumer Electronics*, vol. 48, no. 2, pp. 366-375, 2002.

[9] D. D. Levine, "Iometer User's Guide," Intel Server Architecture Lab. 2003.

Seongmin Kim received the BS degrees in computer engineering from Kwangwoon University, Korea, in 2011. He is currently working toward the MS degree at the School of Computer Engineering, Kwangwoon University. His research interests include operating systems, flash memories and next-generation nonvolatile memories, and embedded system.

Taeseok Kim received the BS, MS and PhD degrees in computer science from Seoul National University, Korea, in 2000, 2002, and 2007 respectively. He is currently an assistant professor in the department of computer engineering, Kwangwoon University, Seoul, Korea. His research interests include multimedia systems, operating systems, storage systems, and embedded system.