A Novel Memory Hierarchy for Flash Memory Based Storage Systems

Keun Soo Yim

Abstract—**Semiconductor scientists and engineers ideally desire the faster but the cheaper non-volatile memory devices. In practice, no single device satisfies this desire because a faster device is expensive and a cheaper is slow. Therefore, in this paper, we use heterogeneous non-volatile memories and construct an efficient hierarchy for them. First, a small RAM device (e.g., MRAM, FRAM, and PRAM) is used as a write buffer of flash memory devices. Since the buffer is faster and does not have an erase operation, write can be done quickly in the buffer, making the write latency short. Also, if a write is requested to a data stored in the buffer, the write is directly processed in the buffer, reducing one write operation to flash storages. Second, we use many types of flash memories (e.g., SLC and MLC flash memories) in order to reduce the overall storage cost. Specifically, write requests are classified into two types, hot and cold, where hot data is vulnerable to be modified in the near future. Only hot data is stored in the faster SLC flash, while the cold is kept in slower MLC flash or NOR flash. The evaluation results show that the proposed hierarchy is effective at improving the access time of flash memory storages in a cost-effective manner thanks to the locality in memory accesses.**

Index Terms—**Flash memory, high-speed I/O, write buffer, non-volatile RAM, and SLC/MLC flash chips.**

Manuscript received October 20, 2005; revised December 3, 2005.
Computing Laboratory, Samsung Advanced Institute of Technology, San 14-1, Nongseo-dong, Giheung-gu, Yongin, Gyeonggi-do, Korea. (Postal Code: 449-712)
E-mail : keunsoo.yim@samsung.com

I. Introduction

Most of the mobile and embedded systems (e.g., digital camera, cell phone, and PDA) use flash memory as a storage medium for storing and managing personal and multimedia data. As the system users want to manipulate high-quality and large-size data in real-time, flash storage shall provide a larger capacity with a faster I/O speed. Fortunately, the flash storage capacity increases enough to satisfy the needs. For example, the NAND-type flash chip capacity is doubled in every year (e.g., 8GB in 2004 and 16GB in 2005 [1]), and a flash storage contains more many numbers of NAND flash chips (e.g., from 2 to 16) than the previous product.

However, the I/O speed is not sufficient for the electronic appliances which handle multimedia data in real-time. Figure 1 shows the write performance of flash memories and hard disks as a function of the write data size [2]. If the write data size is small (e.g., 10-100KB), NAND flash type-II provides the best performance. However, if the write data size is larger than about 300KB, an IDE hard disk provides the better performance than NAND type-II. Similarly, the disk has a faster write speed than NAND type-I and NAND MLC (multi-level cell) if the write data size is larger than about 35KB and about 25KB, respectively. These are because of two main reasons. One is that hard disk provides an outstanding speed for sequential I/O operations (e.g., ~25MB/s). The other is that in flash memory an erase should precede writes. The erase can be performed in a larger unit than the write, and it takes a long time of 2ms in NAND flash and 1s in NOR flash. Thus, the existing techniques logically hide the erase operation and provide an identical interface

with hard disk. However, the slow write still degrades the value of flash memory as compared with hard disks and MEMs-based storages [3].

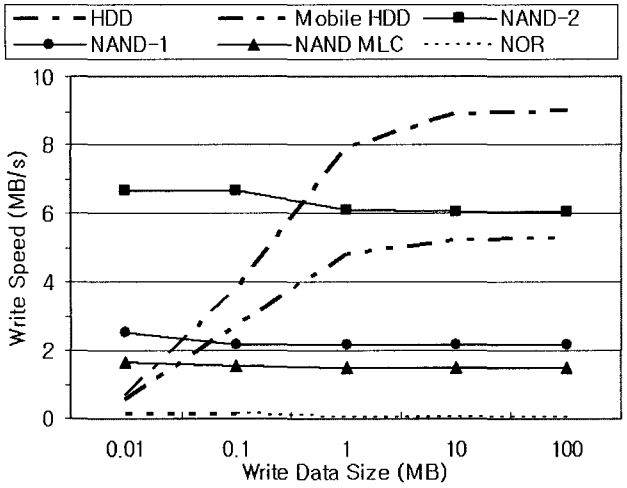


Fig. 1. Write performance of various storage media.
*NAND-1: NAND flash type-I with 512B page size; NAND-2: NAND flash type-II with 2KB page size; NAND MLC: Multi-level cell NAND flash type-I; HDD: IDE hard disk; Mobile HDD: Mobile hard disk.

In this paper, in order to further improve the write speed, we present a novel memory hierarchy for flash memory storages by using emerging non-volatile RAMs (e.g., MRAM, FRAM, and PRAM [4]) and many types of flash memories (e.g., SLC and MLC flash chips). First, the proposed storage system analyzes the I/O history to predict the locality pattern and recognizes the upper-layer file system layout. Then, it classifies each write request into two types: hot and cold, where hot page is vulnerable to be modified in the near future and vice versa. Second, only the hot pages are stored in the non-volatile RAM buffer, making the host write latency short, because these RAMs have a faster write speed than the flash memories and do not have an erase operation. If a page stored in the buffer is modified, the page is directly updated on the buffer, reducing a write operation to flash memory. Third, if a hot page stored in the buffer is evicted, the page is stored in the SLC flash memory. On the other hand, all cold pages are stored in MLC flash memory. Thanks to the locality in memory accesses, the overall write performance is comparable to the SLC flash while the overall cost is similar to the MLC flash. Fourth, the non-volatile RAM buffer is used to provide a byte-based I/O interface, enabling an execution-in-place (XIP) feature. The experimental results show that the proposed memory hierarchy is effective at improving the I/O speed of flash storage systems in a cost-efficient manner.

The rest of this paper is organized as follows. Section 2 briefly reviews the related works on flash storage systems. Section 3 presents both the organization and mechanism of the proposed storage system. Section 4 provides both the experimental methods and results. Section 5 concludes this paper with a summary.

II. Related Work

The majority of the previous works in flash memory was focused on the write performance. Initially, the flash translation layer (FTL) has been developed as an intermediate software layer between a host system and flash memories [5]. FTL logically hides the erase operation of the flash. Specifically, it redirects a write request to a flash page, which has been erased in advance and then maintains a mapping table for the redirected pages. Based on this, FTL provides a common block I/O interface to the host which is similar to the hard disks. However, FTL has no impact on the speed of write operation itself.

On the other hand, the flash compression layer (FCL) can improve the write speed by transmitting data in a compressed form [6]. A fast hardware (de)compressor is used to hide the (de)compression latency. Also FCL is able to expand the flash memory capacity by storing the data in a compressed form. However, its performance heavily relies on the compression ratio of the stored data, and thus its application area is often limited to USB memory sticks, which usually store common data files.

In I/O subsystems, there are two fundamental techniques for improving the I/O speed. One is I/O parallelism. For example, there are many types of parallel I/O architectures in various memory layers (e.g., multi-port cache, interleaved memory, and RAID). Similarly, in flash storage, multiple flash chips are already in use in order to partially parallelize the I/O operations. The other is an input caching and output buffering technique [7]. The input caching is for read and is mainly used in NAND flash. Specifically, only page-based I/O is available in

NAND flash, and thus it is not suitable for the direct execution of a program code. To address this, OneNAND uses a SRAM or DRAM cache in conjunction with NAND flash and provides not only an XIP capability but also a fast read speed [1].

However, so far it was hard to use the output buffering technique in flash storages due to the following two reasons. First, a storage medium shall guarantee the consistency of write data. Thus, the write buffer should be non-volatile [7, 8]. A volatile write buffer (e.g., SRAM and DRAM) can not memorize its data if the flash storage is detached from the host or the host power goes down unexpectedly. Second, the write buffer shall have a faster write speed than the flash. Until now, only battery-backed RAMs satisfied these two conditions. However as flash storages (e.g., CompactFlash, SmartMedia, MultiMedia, SectureDigital cards, and USB sticks) are a tiny removable device which can not hold a battery, battery-backed RAMs are not appropriate.

Fortunately, recently new non-volatile RAMs were produced in a large-scale [4]. Table 1 summarizes the characteristics of these non-volatile RAMs as compared with the volatile RAMs and flash memories. These non-volatile RAMs support byte-based I/O with a faster read and write speed than flash memories, and they do not have an erase operation. In Section 3, these RAMs are utilized as a write buffer of flash memory storages.

Table 1 shows that in flash memories there are tradeoffs between performance and cost. For example, NAND flash type-II provides the best I/O performance, while MLC NAND flash has the lowest cost per byte ratio. Thus, the proposed storage system embeds these heterogeneous flash memory chips in order to maximize the I/O performance while reducing the cost per byte ratio as described in Section 3.

III. Proposed Flash Storage Architecture

This section describes the overall organization and specific mechanisms of the proposed storage system.

1. Heterogeneous Storage Architecture

We basically classify the existing memory devices into four types as shown in Table 1. We select one or set of devices from each type to construct a memory hierarchy. Figure 2 depicts the overall memory hierarchy. In the figure, type S1 and S2 mean the main storage devices where S1 is faster but more expensive than S2. Type W is used as a buffer for both read and write operations since it has the fastest access speed and supports byte-based I/O operations. Optionally, type R can be used as a read buffer in order to extend the lifetime of type W device that has a physical limit in its executable I/O operation count. The

Table. 1. Characteristics of Volatile RAMs, Non-Volatile RAMs, and Flash Memories.

Type	Device	Volatility	I/O Unit	Cell Size	Endurance	Read Time	Write Time	Erase Time
R	SRAM	Volatile	Byte	50~80F^2	10^{15}	<50ns/B	<50ns/B	N/A
R	DRAM	Volatile	Byte	8~10F^2	10^{15}	<100ns/B	<100ns/B	N/A
W	MRAM	Non-volatile	Byte	>30F^2	10^{12}	<50ns/B	<50ns/B	N/A
W	FRAM	Non-volatile	Byte	15~30F^2	10^{12}	<100ns/B	<100ns/B	N/A
W	PRAM	Non-volatile	Byte	8~10F^2	10^{12}	<100ns/B	<500ns/B	N/A
W	Ploymer	Non-volatile	Byte	5~6F^2	10^{10}	10μs/B	10μs/B	N/A
S1	NAND type-II	Non-volatile	Page	5~10F^2	10^5-10^6	<25μs/page	~300μs/page	~2ms/block
S1	NAND type-I	Non-volatile	Page	5~10F^2	10^5-10^6	<12μs/page	~200μs/page	~2ms/block
S2	NAND MLC	Non-volatile	Page	5~10F^2	10^4-10^5	<20μs/page	~300μs/page	~2ms/block
S2	NOR Flash	Non-volatile	Byte	5~10F^2	10^5-10^6	<150ns/B	~200μs/B	~1s/block

* N/A: Not available; Page: 512B in both NAND flash type-I and NAND flash MLC, and 2KB in NAND flash type-II; Block: 16KB in NAND flash type-I, and 128KB in both NAND flash type-II, NAND MLC, and NOR flash.

FTL part is used to manage this memory hierarchy. FTL consists of an embedded microprocessor with a small memory (e.g., ~15KB). A system on chip (SoC) technology can be used to reduce the hardware cost of FTL. Also depending on the fabrication technology, the non-volatile RAM buffer can be embedded in either the FTL SoC or the flash memory chip.

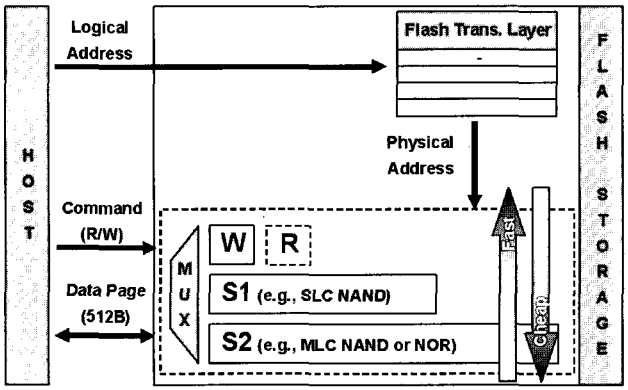


Fig. 2. Overall memory hierarchy of the proposed flash storage architecture.

The host requests I/O operations to the storage system using a typical block I/O interface, where the logical block size is 512 bytes, and an I/O request consists of I/O mode (either read or write), number of logical blocks, and address of start block. The host then waits until it receives an acknowledgement signal from FTL. FTL analyzes the I/O request and consequently controls the non-volatile write buffer, flash memories, and internal buses.

Specifically, since the buffer provides the fastest I/O speed than the flash memories, and the S2 devices (e.g., MLC flash) has the lowest cost per byte ratio than S1 device (e.g, SLC flash), both the buffer and flash memory parts logically construct a hierarchy of memory. The proposed system utilizes this hierarchy in order to support a fast I/O speed (as fast as the buffer) with a low cost per byte ratio (as low as the MLC flash). For example, FTL can handle a write request by using the buffer to send an acknowledgement quickly. This reduces the host delay short. Then, when the flash storage idles, FTL can copy the data stored in the buffer to flash memories. Similarly, when the buffer is full, FTL can utilize the fact that the SLC flash has a faster write speed than the MLC flash. Thus, a write data can be first stored in the SLC flash and then it can be copied into the MLC flash.

After FTL copied the data from the buffer to the flash memories, FTL makes a mark for the data in the buffer. If an exception occurs (e.g., power failure) during the write-back operation, the mark will not exist in the buffer. At boot-up time, FTL checks this mark to guarantee the consistency of the data in the buffer. Optionally, for this purpose, an error checking code (ECC) can be used.

In addition, the endurance of the non-volatile RAMs is about six orders of magnitude longer than that of flash memories. This implies that if the flash storage capacity (e.g., <1GB) is smaller than the six orders of magnitude of a non-volatile RAM capacity (e.g., 1KB), which is used as a write buffer, the non-volatile RAM will work safely as long as the lifetime of the flash storage, even though the non-volatile RAM buffers all write data of the flash storage. The exact lifetime of the write buffer is partially influenced by the management policy and the I/O pattern.

2. Data Locality Aware Management

The proposed system divides the write buffer into three parts. These three parts are used for different purposes. One part is for processing a small write request (e.g., ~4KB) faster than the flash memory. Specifically, the host generates a write operation to a set of contiguous flash pages. If the host requests a 4KB write operation (e.g., for file system metadata), FTL stores the requested data to the buffer and quickly sends an acknowledgement to the host. As the buffer is fast, this makes the host delay short. Later when the storage idles, FTL copies the data stored in the buffer to the requested address of the flash memory.

Another part of the buffer is used for reducing the number of write operations performed by the flash memory. Figure 3 shows the proposed locality-aware write buffering technique. This technique individually handles each page in a write request. FTL first checks whether the requested page exists in the buffer or not. To shorten this checking time, FTL maintains a hash table in its volatile RAM where the hash table directs the physical address of all data stored in the buffer. The physical addresses are initially stored in the buffer and are loaded into the volatile RAM when the storage starts up.

If the requested page exists in the buffer, FTL directly

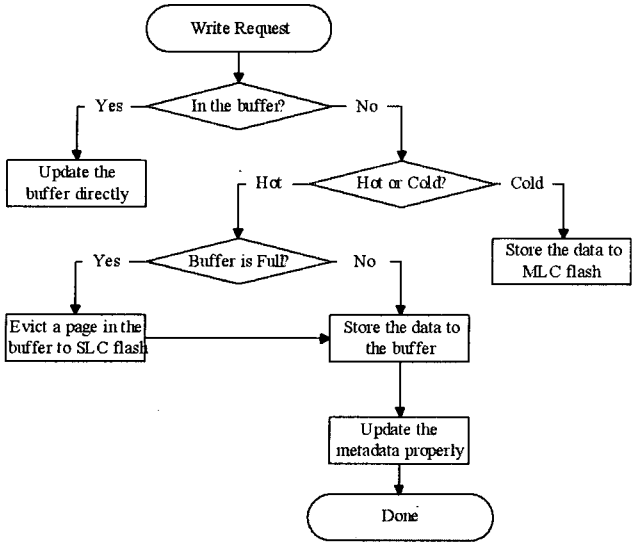


Fig. 3. A write algorithm of the proposed system.

updates the page in the buffer to the requested one. Otherwise, FTL analyzes the access history of the requested page. If there was a write in the near past, FTL considers that the page has a strong temporal locality, and thus it classifies the page into a hot page. If not, the requested page is classified into a cold page. In practice, a simple least-recently-used (LRU) queue is used to classify the hot and cold pages [7]. Specifically, the LRU queue keeps the most recently accessed data in its top position and evicts the data stored in its bottom position first if a new data is arrived. This guarantees that a data stored in the queue has been accessed by one of the previously written N different pages, where N means the LRU queue size in page.

In addition, FTL can explicitly aware the layout of file system. For example, by checking the content of the first logical block, typically a super block of file systems, FTL is able to detect the type of file system (e.g., FAT or FFS). If it is a FAT-compatible file system, FTL can easily find out the location of important metadata (e.g., FAT tables). Then, these important metadata can be explicitly classified into the hot in order to improve the I/O performance.

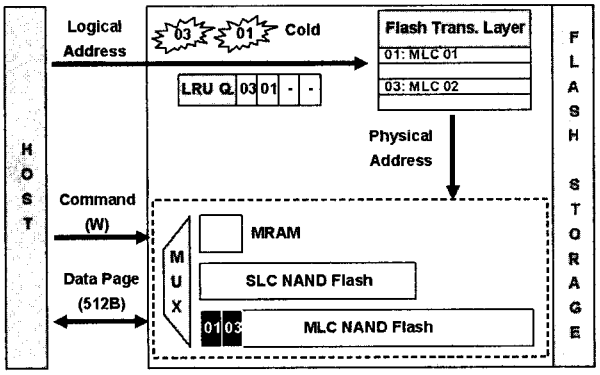
Based on this previous locality information, we predict the future write patterns. Specifically, we use a heuristic algorithm, which assumes that the future write pattern will be similar to the past pattern. This guides us that the hot page will be modified in the near future, while the cold pages will not. Thus, only hot pages are stored in the write buffer. If a write request is generated to a hot page stored in the buffer, we can directly update the page in the buffer, and this reduces a write operation of flash memory.

Figure 4 is an example of this locality aware data management. Initially, the LRU queue is empty. If write operations are serially requested to page 1 and 3, both are cold pages, and thus they are stored in the MLC flash memory as shown in Figure 4(a). Then as shown in Figure 4(b), if the page 1 is modified, the LRU queue realizes that the page is hot, and thus it stores the page to the non-volatile RAM buffer, e.g., MRAM. If the page 1 in the buffer is modified again, the modification is directly performed in the buffer as exemplified in Figure 4(c). If the page is not stored in the buffer, the modification needs a write operation to the flash memory. Thus, this reduces a write to flash. After handling several writes, the buffer becomes full. Thus, a page in the buffer should be evicted. In this case, we select an oldest page from the buffer. Then, as shown in Figure 4(d), the oldest one is copied to the SLC flash memory because it was a hot page, vulnerable to be accessed.

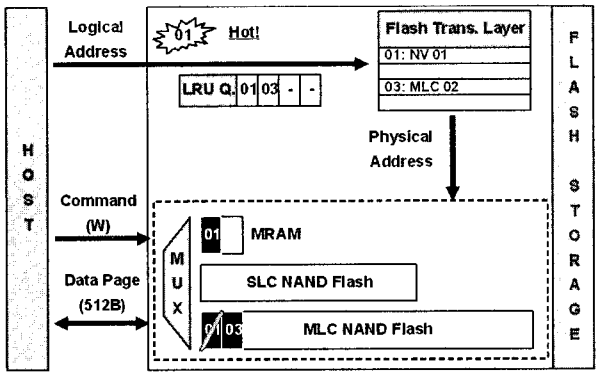
The other part of the buffer is used for supporting a byte-based read operation. In read mode, FTL first tries to find a requested page in the buffer. If it exists in the buffer, the requested byte is delivered to the host. If not, FTL loads the page from the flash memory to the buffer and simultaneously delivers the requested byte to the host. This enables an XIP capability for NAND flash [9]. Also, as the buffer has a faster read speed than the flash memory, this improves the read performance of proposed storage.

IV. Performance Evaluations

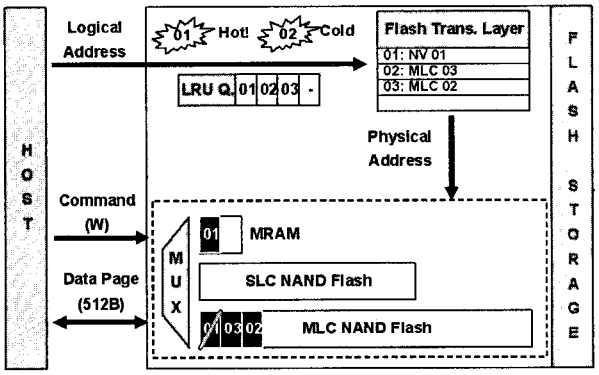
A trace-driven simulation was used to evaluate the I/O performance. The traces were collected from an USB stick, in which we copy, manipulate, and remove both the multimedia files (*media*) and office files (*office*) and also run typical office programs (*program*). In addition, we evaluated the I/O latency of the proposed storage system over an existing flash file systems of JFFS2 [10] on an Intel machine running the latest patch version of Linux kernel 2.4. We modeled both the non-volatile RAMs and flash memory devices whose I/O characteristics are shown in Table 1. Based on this model, we developed virtual



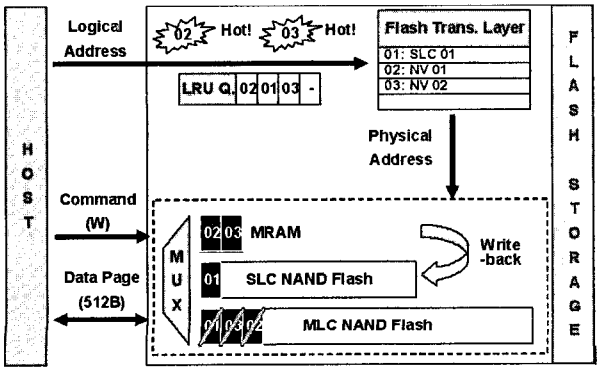
(a) Two cold pages



(b) A hot page



(c) Buffered write



(d) Write-back

Fig. 4. An example of data locality aware management.

flash device drivers which hold CPU during the time required to perform a requested I/O operation in a real non-DMA flash device. The virtual drivers provide a common flash driver interface for the flash file systems and record the numbers of performed I/O, which are further used to analyze the I/O characteristics of the file systems. We also measured the time spent to process the flash file systems on CPU.

We observed that the first technique, which stores a write data to the write buffer and quickly sends an acknowledgement, is useful for the small write requests (e.g., ~ 4KB) because this can make the host system delay short. However, with the large-size sequential write operations, this technique provides a negligible performance benefit in terms of its write speed.

We then evaluated the write latency of the proposed locality-aware write buffering with all appropriate buffer and flash combinations. The write latency means the I/O time observed by the host system by executing the *media*, *office*, and *program* traces. We compared the write latency of the proposed storage with that of a conventional one, which does not have a write buffer. Figure 5 summarizes the speed up ratio of the proposed storage over the conventional one. It shows that all flash memory media take advantages from the fast write buffer.

Specifically, the performance characteristics of the proposed storage are summarized as follows. First, the larger the buffer size is the higher the speed-up ratio is. Second, a faster write buffer (e.g., MRAM) provides the

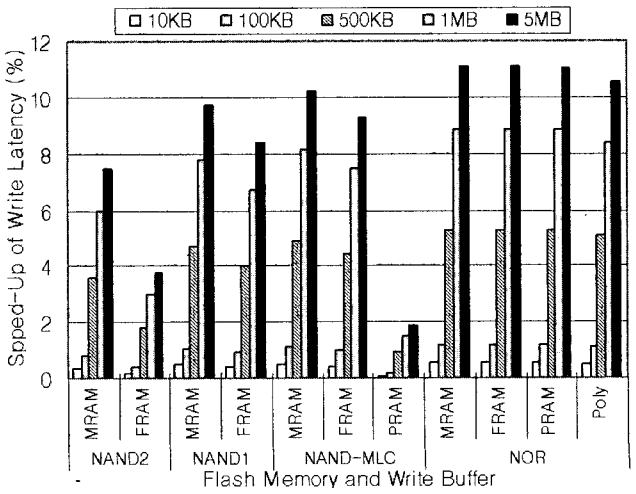


Fig. 5. The speed-up ratio of the write latency of the proposed storage system against to the convention storage.

higher speed-up ratio than the slower buffer. Third, a slower flash memory medium (e.g., NOR flash) results in the higher speed-up ratio than the faster one. When an 1MB MRAM write buffer is used, the NAND flash type-II, NAND flash type-I, MLC NAND flash, and NOR flash provides about 6%, 8%, 8%, and 9%, respectively, faster write speed than the conventional flash storage if we use the proposed locality-aware write buffering technique.

V. Conclusions

In this paper, we have utilized both the non-volatile RAMs and heterogeneous flash chips in flash storages to improve the I/O performance in a cost-efficient manner. First, since the buffer is faster than the flash, write can be done quickly in the buffer, making the write latency short [11]. Second, if a write is requested to a data stored in the buffer, the write is directly processed in the buffer, reducing a write operation to flash memory [12]. Third, based on the buffer, the proposed storage provides a byte-based I/O interface which enables an XIP capability. Fourth, we have used heterogeneous flash memory chips for reducing the cost per byte ratio with a good performance. The experimental results have shown that the proposed memory hierarchy is effective at improving the I/O speed of flash storage systems in a cost-efficient manner.

References

[1] Samsung Electronics, *Data Sheets of NAND Flash Memory*, http://www.samsungelectronics.com/.

[2] Intel Corporation, *Data Sheets of StrataFlash Memory*, http://www.intel.com/.

[3] S. W. Schlosser and G. R. Ganger, "MEMS-Based Storage Devices and Standard Disk Interfaces: A Square Peg in a Round Hole?," *In Proceedings of the USENIX Conference on File and Storage Technologies (FAST)*, pp. 87-100, 2004.

[4] B. Prince, D. Prince, and J. Hartigan, "Emerging Memories: Applications, Device and Technology," *Technical Report, Memory Strategies International*, 2004.

[5] J. Kim, J. M. Kim, S. H. Noh, S. L. Min, and Y. Cho, "A Space-Efficient Flash Translation Layer for CompactFlash Systems," *IEEE Transactions on Consumer Electronics (TCE)*, Vol. 48, No. 2, pp.366-375, 2002.

[6] K. S. Yim, H. Bahn, and K. Koh, "A Flash Compression Layer for SmartMedia Card Systems," *IEEE Transactions on Consumer Electronics (TCE)*, Vol. 50, No. 1, pp. 192-197, 2004.

[7] A. Silberschatz, P.B. Galvin, and G. Gagne, *Operating System Concepts*, 6th Ed., John Wiley & Sons Inc., 2003.

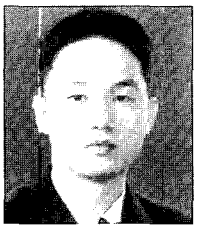
[8] A. S. Tanenbaum and M. van Steen, *Distributed Systems: Principles and Paradigms*, Prentice Hall, 2002.

[9] C. Park, J. Seo, S. Bae, H. Kim, S. Kim, and B. Kim, "A Low-Cost Memory Architecture with NAND XIP for Mobile Embedded Systems," *In Proceedings of the ACM/IEEE International Conference on Hardware /Software Codesign and System Synthesis (CODES+ ISSS)*, pp. 138-143, 2003.

[10] D. Woodhouse, "JFFS: The Journaling Flash File System," *In Proceedings of the Ottawa Linux Symposium (OLS)*, RedHat Inc., 2001.

[11] K. S. Yim (Samsung Elsectronics), "Operation Apparatus Comprising Nonvolatile Memory and Flash Memory Performing Input and Output Operation in Parallel Method and Method of Operating the Same," *Republic of Korea Patent*, Filed No. P2005-0063301, 2005.

[12] K. S. Yim, J.-K. Park, and J. J. Yoo (Samsung Electronics), "Apparatus and Method for Storaging Data Using Flash Memory," *Republic of Korea Patent*, Filed No. P2005-0076368, 2005.



Keun Soo Yim He is a researcher in the computing laboratory of Samsung Advanced Institute of Technology in Korea. His research interests include computer architectures, operating systems, and computer networks for both low-power and high-performance computing. He received double BS degrees in computer science and electronic engineering from Yonsei University; and an MS degree in computer science and engineering from Seoul National University. He was a visiting researcher in the mobile networking group of National Institute of Information and Communications Technology in Japan from January to February 2005. He can be contacted at keunsoo.yim@samsung.com.