Journal of Systems Architecture 56 (2010) 641–653

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

ComboFTL: Improving performance and lifespan of MLC flash memory using SLC flash buffer ☆

Soojun Im, Dongkun Shin *

School of ICE, Sungkyunkwan University, Suwon 440-746, Republic of Korea

ARTICLE INFO

Article history:
Received 20 November 2009
Received in revised form 8 September 2010
Accepted 8 September 2010
Available online 24 September 2010

Keywords:
Flash memory
Multi-level cell
Flash translation layer
Embedded storage

ABSTRACT

Multi-level cell (MLC) flash memory has lower bit cost compared to single-level cell (SLC) flash memory. However, there are several obstacles to the wide use of MLC flash memory, including slow write performance and shorter lifespan. To improve the performance and lifespan of MLC flash memory, we propose an FTL (flash translation layer) for MLC flash memory, called ComboFTL. By exploiting the SLC mode of MLC flash memory, ComboFTL manages a small SLC region for hot data and a large MLC region for cold data. To provide the performance and lifespan similar to those of SLC flash memory, ComboFTL identifies the hotness/coldness of data effectively. It can also adjust its several policies based on workload changes. Our experimental results showed that ComboFTL improves the write performance and lifespan of MLC flash memory significantly.

1. Introduction

Since NAND flash memory has many advantages over hard disk drives such as low-power consumption, small size, and high shock resistance, it has been widely used for mobile consumer devices such as MP3 players, digital cameras, personal digital assistants, and cell phones.

The read performance of flash memory is high because it requires no seek time. However, flash memory has a low write performance due to its "erase-before-write" constraint, which forces a block to be erased before data can be written into the block. While the write operation is performed by the page unit, the erase operation should be performed by the block unit, which is composed of several pages. To handle the special features of flash memory, most systems use a flash translation layer (FTL) which maps the logical page address from the file system to the physical page address in the flash memory devices. The address mapping schemes of FTL can be divided into three classes depending on their address mapping granularities, i.e., block-level mapping [1], page-level mapping [2], and hybrid mapping [3–5].

Flash memory has a grid of columns and rows with a memory cell that has two transistors at each intersection. The electrons in the cells of a flash-memory chip affects the threshold voltage, V_{th}, and this threshold voltage determines the state of the cell. While each cell has two states in a single-level cell (SLC) flash memory, two or more bits can be stored on each cell in a multi-level cell (MLC) flash memory [6].

For two-bit MLC, one cell retains two bits for two paired pages, the least-significant-bit (LSB) page and the most-significant-bit (MSB) page. Fig. 1 shows the conceptual structure of a MLC flash block and how the value of a bit is determined by the threshold voltage. Two paired pages, LSB page and MSB page, share the memory cells that belong to the same word line. Each page only uses its own bit position of a bit pattern stored in a cell and each cell can be programmed twice.

For example, if the threshold voltage is changed to a value between 0 and 1.0 V from a negative value by the first programming, it is interpreted as a '10', therefore, the MSB and LSB pages interpret the cell as a logical '1' and a logical '0', respectively. If the threshold voltage is increased to be larger than 2.4 V by the second programming, the MSB bit is changed into a logical '0' additionally.

MLC flash is cheaper than SLC flash, because it provides a larger storage capacity than SLC flash for the same-sized die. Therefore, MLC flash is a promising solution for large-scale flash memory systems such as SD cards or solid-state disks (SSD). However, the critical obstacles to the wide adoption of MLC are its poor write performance and short lifespan. To store multiple bits in a memory cell, MLC flash memory should assign narrow threshold voltage ranges for each cell state. Therefore, MLC flash memory requires more precise charge placement and charge sensing, which in turn reduces the performance and endurance of MLC flash memory in comparison to SLC flash memory. The write performance of MLC is about half that of SLC and the available program/erase (P/E) cycles of MLC is about one-fifth that of SLC [7].

☆ 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 (2010-0010387).

* Corresponding author. Tel.: +82 31 299 4584.
E-mail address: dongkun@skku.edu (D. Shin).

1383-7621/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.sysarc.2010.09.005

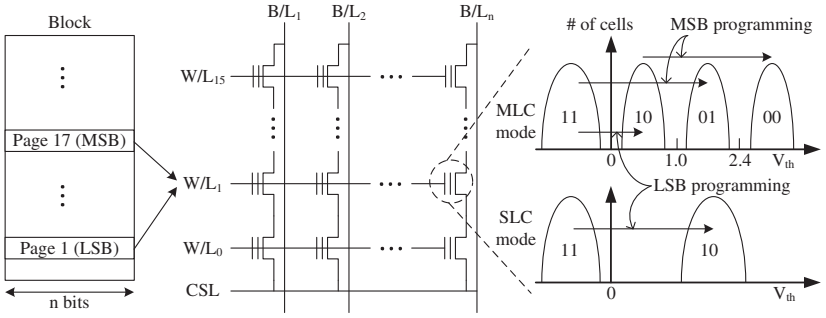


Fig. 1. The structure of MLC flash memory.

As a solution for the limitations of MLC flash memory, two SLC/MLC combined flash architectures were introduced. One uses both SLC flash chips and MLC flash chips to construct a large scale flash storage [8]. The other approach uses an SLC/MLC combined flash chip which has both SLC blocks and MLC blocks in a single chip. By programming only the LSB of a cell in the MLC flash memory, we can use the cell as SLC [9,10]. In this case, since only two states of a cell, '11' and '10', are used in the SLC mode as shown in Fig. 1, we can use a wider voltage range for each cell state. Therefore, when we programmed only LSBs into a particular block, the effective properties of that block become similar to those of an SLC flash memory block. That is, MLC flash is used in SLC mode. Conversely, if both LSB and MSB are programmed (MLC mode), we are able to utilize the high capacity provided by the MLC flash memory. Therefore, we can say that the SLC/MLC combined flash uses dual modes, SLC mode and MLC mode. The SLC and MLC modes are applied to each block. That is, if one flash block is set to the SLC mode (or MLC mode), all the memory cells within the block are used for the SLC mode (or MLC mode). The dual mode programming allows two different types of blocks to exist in the same flash chip simultaneously.

One real product example of SLC/MLC combined flash memory is Samsung's Flex-OneNAND [11], which enables users to configure to partition it into SLC and MLC regions by specifying the last SLC block index. As shown in Table 1, the MLC block using SLC mode has a shorter write latency than does the MLC block using MLC mode.

For the SLC/MLC combined flash memory chip, we can divide the flash memory blocks into two regions to ease management, an SLC region and an MLC region. Depending on the size of each region, the total storage capacity of the flash memory is determined. For example, if there are 1024 blocks in a flash memory chip where 256 blocks use the SLC mode and 768 blocks use the MLC mode, the total capacity is 458MB (=65 + 393MB) when we assume the block sizes in the SLC and MLC modes are 256KB and 512KB, respectively.

To exploit the SLC/MLC combined flash architecture effectively, heterogeneous memory regions should be transparent to the file systems layer. By utilizing a small-sized SLC region as a write buffer for the frequently-updated data, the SLC/MLC combined flash can provide similar write performance to that of an SLC-only flash chip with a capacity similar to that of an MLC-only flash chip. In such architecture, we need an efficient hot/cold data separation technique which is able to determine whether to write a page sent from the file system at the SLC region or the MLC region. In addition, the architecture should be able to detect the changes of data state and move the data to the proper region.

Table 1
Performance comparison of different types of cell programming (μs) [7].

Block type	Read (page)	Write (page)	Erase (block)
SLC	399	417	860
MLC w/SLC mode	409	431	872
MLC w/MLC mode	403	994	872

In this paper, we propose a flash translation layer called ComboFTL for SLC/MLC combined flash memory that effectively supports the above described storage architecture. ComboFTL attempts to achieve the highest I/O performance and to prolong the lifespan of flash memory by handling hot data efficiently. Moreover, ComboFTL uses an adaptive policy which can handle workload change.

Our proposed technique is the first practical and efficient approach for SLC/MLC combined flash memory including the following main contributions:

(1) The hot/cold separation algorithms for SLC/MLC combined memory are proposed.
(2) The SLC write buffer replacement and garbage collection algorithm, which is an adapted version of GCLOCK algorithm, is proposed.
(3) Several adaptive approaches to handle workload variation are proposed.
(4) The efficient block management and hot data detection techniques for the MLC region are proposed.

By simulation-based evaluations using real mobile phone workloads, we show that ComboFTL provides 84% of the write performance of SLC flash memory on average (1.48 times the write performance of MLC flash memory) when using SLC/MLC combined flash memory. In addition, ComboFTL significantly reduces the erase count of the MLC region and thus prolongs the lifespan of flash chip.

The rest of the paper is organized as follows. In Section 2, related works and their drawbacks are introduced. Section 3 describes the overall storage architecture provided by ComboFTL. Sections 4 and 5 explain the management policies for the SLC and MLC regions, respectively. Experimental results are presented in Section 6. Section 7 concludes with a summary and future works.

2. Related works

Several hot/cold separation techniques were proposed for flash memory management. There are two motivations in separating hot data from cold data. The first is to minimize the garbage collection (GC) cost. The GC selects victim flash blocks to be reclaimed, moves all valid pages to other free clean blocks and erases the victim blocks to change them clean blocks. If a victim block has only hot data, most of pages may be invalid due to frequent updates and thus the page migration cost will be small. The second is to increase the lifespan of flash memory by making even the wear

counts of flash blocks. By writing hot data in the block with a small erase count, the block will be erased frequently and thus its erase count increases.

Wu and Zwaenepoel [12] proposed a locality gathering algorithm for their eNVy flash storage system. The algorithm determines the hotness of a block with the frequency with which the block is cleaned. Kawaguchi et al. [2] proposed a flash file system based on LFS, which separates cold segments and hot segments in order to prevent hot data from being mixed with cold data.

Chiang et al. [13] proposed the Cost–Age–Times (CAT) garbage collection method which employs a hot–cold identification mechanism. They also proposed the DAC policy [14], which clusters the data with the similar write access frequencies in the same regions. Chang and Kuo [15] proposed two-level LRU list-based hot-cold identification mechanism. Hsieh et al. [16] used a multi-hash-function for hot-data identification which has scalability considerations on precision and memory-space overheads.

Lee et al. [17] proposed an FTL called LAST, which uses a hot/cold detection technique in order to separate hot data and cold data. Different from other techniques, it is proposed for the log buffer of hybrid mapping FTLs [4].

Although these techniques focused on optimizing garbage collection and wear-leveling, their target was not the SLC/MLC combined flash memory. We need a different hot/cold separation approach considering the characteristics of SLC/MLC combined flash memory. The motivation of hot data identification in this paper is to store hot data at the SLC region which provides high performance and longer lifespan. In addition, we need an adaptive approach which can adjust its policies depending on workload variation on each region.

Several studies have been performed on SLC/MLC combined flash memory. Park et al. [18] proposed an FTL called MixedFTL for an SLC/MLC combined flash chip. The MixedFTL sends all the write requests to the SLC region and then moves the cold data into the MLC region if there are no updates during a long time. Thus, many cold data invoke unnecessary write operations on the SLC region and migrations to the MLC region. Moreover, since it allocates an SLC mode block for each logical block, the SLC mode blocks have low utilizations thus invoke frequent GCs.

Lee et al. [7] proposed a file system called FlexFS for SLC/MLC combined flash memory, which determines the target region for a file depending on the storage size of the SLC and MLC regions, rather than the hotness of the file. That is, if there are many free spaces, it sends most of file write requests to the SLC region even though the file will not be updated frequently. FlexFS exploits the hotness/coldness of the file only to determine whether to move it into the MLC region. Since FlexFS also writes all files to the SLC region and then moves non-updated files into the MLC region, it wastes SLC region space and may invoke large migration costs. In addition, it demands a file system change, while our approach can be used along with any file system.

The main difference between the previous schemes and our work is that the former send all write requests to the SLC region, while our work writes only hot data to the SLC region. In addition, the previous schemes assumed that the mode of a block can be reconfigured at run time. However, if the SLC block is reconfigured into an MLC block, it is difficult to estimate the remaining P/E cycles of the block, which is required for wear-leveling. The flash memory chip venders do not provide detailed test results for the block reconfiguration because it is impossible to test all potential reconfigurations. The venders provide only the P/E cycles of the SLC block and MLC block assuming no reconfiguration. Therefore, MixedFTL and FlexFS are unrealistic solutions.

Chang [8] proposed the hot-data filtering and utilization throttling techniques for hybrid SSD which uses both SLC flash chips and MLC flash chips. The SLC flash chips are treated as a circular log space with the size of k number of blocks. The hot-data filter uses the 2-means clustering algorithm to identify the small hot data and periodically redefines the threshold of small writes in order to adapt to workload change. The utilization throttling adjusts the value of k to balance the wearings of SLC blocks and MLC blocks. Chang's scheme is quite similar to our ComboFTL.

However, we adjust the threshold of small hot data based on the migration traffic from SLC region to MLC region and therefore we can also regulate the wearings of SLC and MLC regions. In addition, we attempt to exploit the SLC region fully since it provides a high performance while the utilization throttling technique exploits only a portion of SLC region. One critical problem of Chang's approach is that there can be many cases where the 2-mean clustering algorithm cannot find the two majorities. Fig. 2 shows the distribution of write requests with respect to different request sizes in four real storage traces. Although there are two explicit majorities in Phone-1 trace, it is not trivial to find the two majorities in other traces. Table 2 summarizes the differences between Chang's technique and ComboFTL.

Our previous work [19] proposed an FTL for SLC/MLC combined flash memory which also uses a hot/cold separation technique. Compared to the previous work, ComboFTL has four main enhancements. First, it uses an adaptive approach for hot/cold separation to handle workload variation. Second, it provides a special warm data handling technique, called an N-chance algorithm. Third, it uses the hot data detection techniques for the MLC region. Finally, ComboFTL uses enhanced address mapping schemes for the SLC and MLC regions.

3. Storage architecture

3.1. Management of the SLC region and MLC region

Fig. 3 shows the overall storage architecture of the SLC/MLC combined flash memory. The sizes of the SLC and MLC regions are fixed at design time and no block is able to be reconfigured at run time. Several researchers showed that small data is generally hot data [17,8]. Therefore, we use a size-based hot/cold separation technique, which predicts the hotness of data based on the write request size. Hence, small-sized write requests from file systems are first sent to the SLC region (write buffer), which is managed by page-level mapping. A sequential large data whose size is larger than the size threshold θ is sent to the MLC region. There may be counterexample cases where small data is cold or large data is hot. In such cases, ComboFTL will move the data into the proper region using the auxiliary hot/cold detection techniques such as cold data early migration and hot data detection explained at Sections 4 and 5.

When there is no free space in the SLC region, the GC is invoked to make free space by reclaiming the space occupied by invalid pages or by moving cold data into the MLC region. To simplify garbage collection, the SLC region is maintained as a circular buffer. The tail pointer of the circular buffer points to the oldest block. The oldest block is selected as a victim for GC since it is probable that the oldest block has a large number of invalid or cold pages.

The MLC region of SLC/MLC combined flash memory should be handled differently than normal MLC flash memory because the MLC region contains only cold data due to the SLC write buffer, thus it is infrequently updated. In addition, the MLC region has several requirements. First, the GC cost should be small since the copy operation has a high cost in the MLC flash blocks. Therefore, block-level mapping or hybrid mapping, which invokes frequent copy operations to merge blocks, is not a proper solution. Therefore, page-level mapping is more suitable. Second, since the storage size of the MLC region is large, the mapping level should be selected

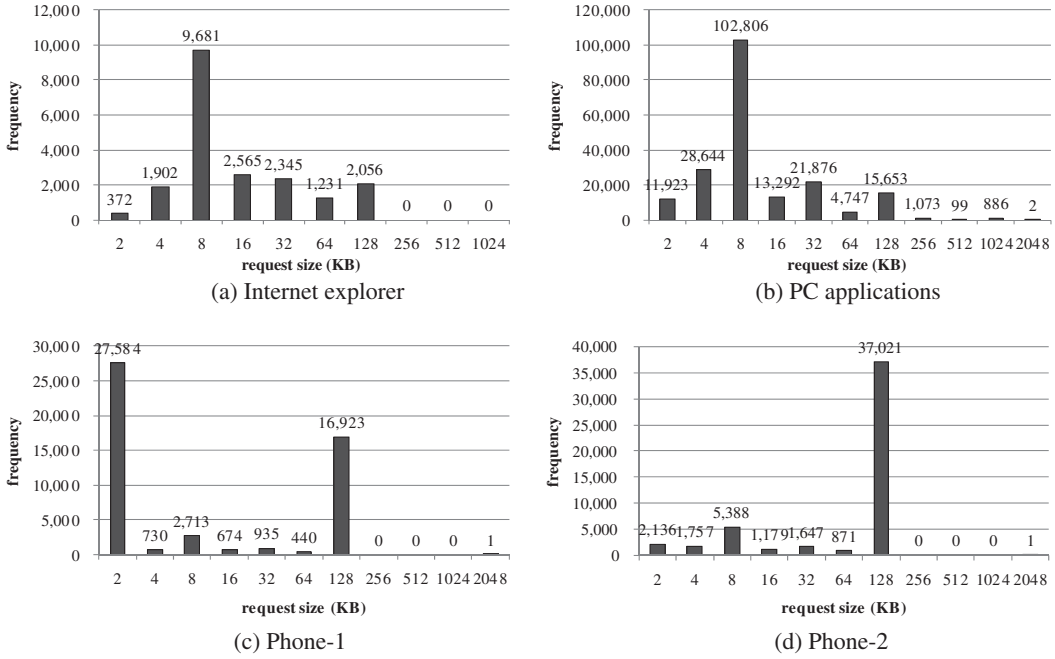


Fig. 2. Distributions of different request sizes.

Table 2
Comparison between Chang's technique and ComboFTL.

Technique	Chang [8]	ComboFTL
Hot/cold separation	2-Means clustering	Size-based separation
Threshold adaptation	2-Majority update	SLC-to-MLC traffic *N*-chance
SLC-to-MLC	1-Chance	Hot/warm partitioning Early migration
MLC-to-SLC	–	Hot unit detection
Wear-leveling	Utilization throttling	SLC-to-MLC traffic

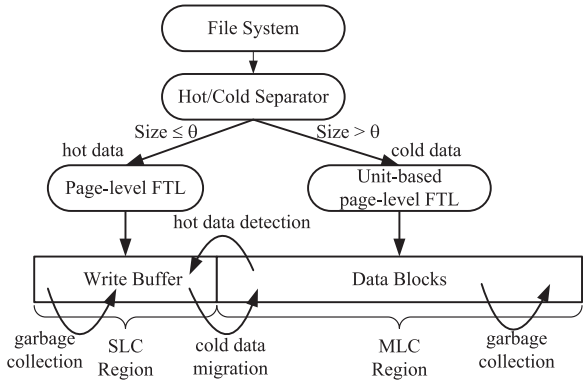


Fig. 3. Storage architecture for SLC/MLC combined flash memory.

such that it does not require a large memory space. Therefore, pure page-level mapping is also unsuitable. Considering these two conflicting requirements, ComboFTL adopted the unit-based page-level mapping [20], instead of the block-level mapping for the MLC region used in our previous work [19].

In the unit-based page-level mapping, one unit consists of multiple sequential logical blocks similar to the superblock FTL [5], and several physical blocks can be allocated to the unit. There is a limit on the number of physical blocks that can be allocated to a unit. Each page can be written at any physical page within the physical blocks allocated to the corresponding unit. Therefore, page-level mapping information within only the unit boundary is required and the size of mapping information is smaller than that of pure page-level mapping. In addition, this method invokes a small GC cost compared to those of block-level mapping or hybrid mapping since there are no merge operations in page-level mapping.

Fig. 4 shows the examples of unit-based page-level mapping. Fig. 4a and b shows the one-block unit and two-block unit, respectively. Three physical blocks are allocated for each one-block unit. Unit0 has only the pages of the logical block LB0, while Unit1 has only the pages of the logical block LB1 in the one-block unit scheme. Each page is written in the order of request arrival. By the garbage collection for Unit0, the valid pages in the physical block PB10 are moved into PB14, and PB10 can be erased. In Fig. 4b, Unit0 has the pages of logical blocks LB0 and LB1.

3.2. Dynamic adaptation for hot/cold threshold size

The separation for hot data and cold data is based on the write request size. The optimal value of θ depends on the workload and the SLC region size. If θ is too large, many cold data will be sent to the SLC region resulting in many cold data migrations from the SLC region to the MLC region. If θ is too small, there will be few cold page migrations but many write requests will be sent to the MLC region. So, we adjust the value of θ observing the amount of cold data migrations.

We select the value of θ among 8KB, 16KB, 32KB, and 64KB. At every adjustment period, the total amount of cold data migrations is checked. The adjustment period is set to be the same to the SLC region size. That is, after receiving the write requests as the amount of SLC region size, we adjust the threshold. The dynamic adaptation is performed based on the target cold data migration ratio, ϕ, which represents the ratio between the amount of cold data moved from the SLC region into the MLC region during an adjustment period and the SLC region size. To maintain the cold data migration ratio to be ϕ, if the cold data migration ratio is larger than $\phi + \epsilon$, the value of θ is decreased in order to send less data to the SLC region. If it is less than $\phi - \epsilon$, the value of θ is increased. We used 10% and 5% for ϕ and ϵ at the experiments, respectively.

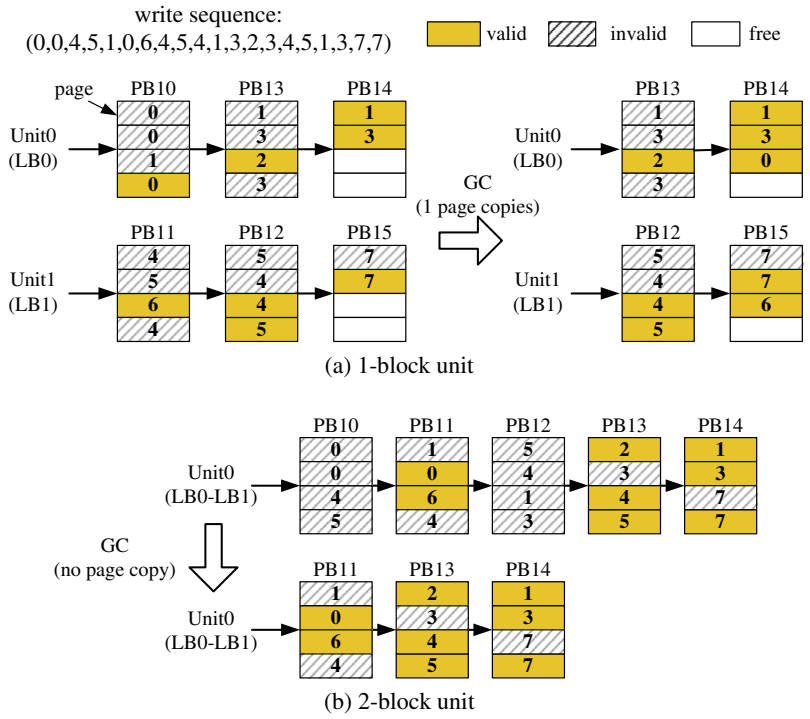


Fig. 4. Unit-based page-level mapping.

The optimal value of target cold data migration ratio should be determined considering the balance between the erase counts of SLC region and MLC region. However, determining the optimal ratio is beyond the scope of this paper.

4. SLC region management

4.1. N-chance algorithm

The SLC write buffer is manipulated as a circular buffer, which has two pointers for head and tail. Newly arrived data is written to the block indicated by the head pointer. The garbage collector selects the oldest block in the tail pointer as a victim. Since it may have a large number of invalid pages, it is a good target for the GC. To reclaim free space, the GC moves the valid pages in the oldest block into the MLC region, erases the block, and then moves forward the tail pointer.

This algorithm can be considered as a flash-memory version of CLOCK [21] replacement algorithm. The algorithm CLOCK maintains a "page reference bit" with every page. On a hit to a page, its page reference bit is set to one. Replacement is done by moving a clock hand through the circular buffer. The clock hand can only replace a page with page reference bit set to zero. However, while the clock hand is traversing to find the victim page, if it encounters a page with page reference bit of one, then it resets the bit to zero. When the CLOCK algorithm is used for flash memory, the page reference bit is not required since a page cannot be overwritten at flash memory. The updated page is written into the location pointed by the head pointer and, therefore, the recently-updated pages are far from the tail pointer. ComboFTL only considers the update hit since the write cost is larger than the read cost in flash memory.

Since the valid pages in the oldest block have not been updated during the wrap-around time, which is the time required for write requests to consume the entire circular buffer, we can regard them as cold data that should migrate to the MLC region. However, if the size of the SLC buffer is too small, several data whose update periods are longer than the wrap-around time will be identified as cold data. Such data will migrate to the MLC region, and thereby invoke a high write cost at the MLC region due to following update requests on the data. We call such data warm data.

Therefore, it is necessary to provide more than one chance for the warm data to remain in the SLC region during GC. To do that, we partition the SLC buffer into a hot partition (from $\texttt{HOT}_{\texttt{start}}$ to $\texttt{HOT}_{\texttt{end}}$) and a warm partition (from $\texttt{WARM}_{\texttt{start}}$ and $\texttt{WARM}_{\texttt{end}}$) as shown in Fig. 5. Each partition is handled as a circular buffer. The write request from the file system is first sent to the hot partition. By the GC of the hot partition (GC_{hot}), the valid pages in the victim block migrate to the warm partition instead of the MLC region. If the hot data and the warm data share one partition, the warm data

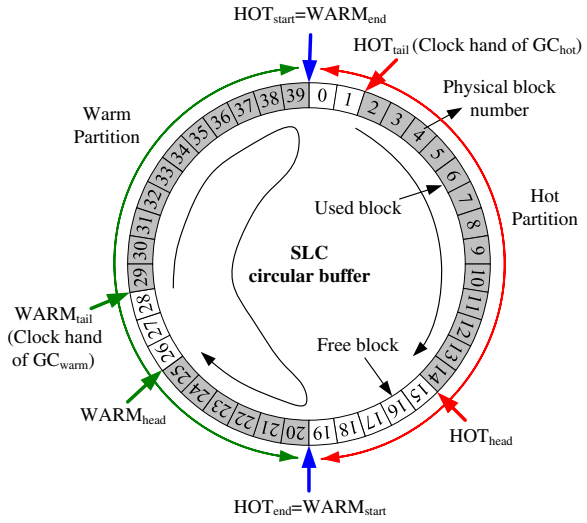


Fig. 5. Structure of SLC write buffer.

are frequently moved by the GC. By separating two partitions, the GC for the hot partition will find many invalid pages at the victim block thus the GC overhead is reduced.

The warm partition is the space for the warm data whose update period is longer than the wrap-around time for the hot partition. If a warm data is updated by a file system request, the old data is invalidated and the new data is written at the hot partition. When there is no free space in the warm partition, the GC for the warm partition (GC_{warm}) is invoked. GC_{warm} gives each warm page N number of chances to remain in the warm partition. We call it the N-chance algorithm. The number of chances used for each page are stored at a volatile memory space for meta-data such as SRAM and are managed by ComboFTL, thus there is no further physical partition within the warm partition. The meta-data are written to a reserved area of flash memory when the storage system is shut down. To balance the program/erase cycles of the hot and warm partitions, the boundary between two partitions is shifted periodically.

The N-chance algorithm is similar to the Generalized CLOCK (GCLOCK) [22] buffer replacement algorithm, which also gives multiple chances of survival from replacement. The N-chance algorithm can be considered as a flash-memory version of GCLOCK. However, the proposed N-chance algorithm has additional features compared with GCLOCK algorithm as follows:

- First, the hot and warm partitions are separated in order to effectively handle hot data. By such a partitioning, the GC overhead for the hot partition is reduced.
- Second, the cold data are kept out of the SLC region by the size-based hot/cold separation technique.
- Third, the early migration for cold data is used. Even though a page does not exhaust all N number of chances, it can be moved into the MLC region by the early migration policy, which will be explained in subsection 4.3.

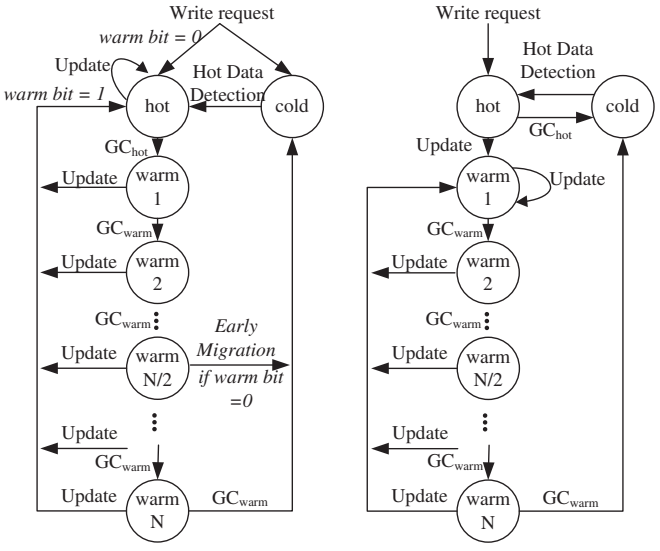
Fig. 6a shows the state change diagram of a page under the N-chance algorithm. When a page is moved by GC_{warm}, the number of chances used for the page increases and its state is changed. If it exhausts all the chances, the page is changed into a cold state and migrates to the MLC region. We denote the set of pages which have used n chances as W_n.

Instead of the size-based hot/cold separation, 2Q [23] algorithm can be applied to admit only hot pages to the SLC region. On the first reference to a page, 2Q places it in a special buffer, the Al queue, which is managed as a FIFO queue. If the page is re-referenced during its Al residency, the page is moved to the Am queue, which is managed as an LRU queue, since it is probably a hot page. If a page is not referenced while on Al, 2Q removes it from the buffer since it is probably a cold page. The 2Q algorithm can be adjusted for SLC/MLC combined storage as shown in Fig. 6b. Each page is first written at the hot partition. If there is no update to the page until it meets GC_{hot}, it is moved into the cold region (MLC region). Otherwise, it is moved into the warm region since the page is not cold data. In experiments, the performances of two versions of hot/cold separation will be compared.

Assume that the numbers of blocks in the hot and warm partitions are B_h and B_w, respectively. Since the hot partition is a circular buffer, a page p will become a target of GC_{hot} after $P \cdot B_h$ number of pages are written at the hot partition, where P is the number of physical pages in an SLC block. If the page p is not updated until the time, it will be moved into the warm partition. Assume that $P \cdot \alpha$ number of pages including the page p in the victim block migrate into the warm partition by GC_{hot}, where α denotes the average ratio of valid pages in the victim block.

The warm partition has $P \cdot B_w$ number of available pages. The warm partition has two sources for write requests, GC_{hot} and GC_{warm}. If we assume the average ratio of valid pages in the victim block of GC_{warm} is β, GC_{warm} consumes $\beta \cdot P \cdot B_w$ number of pages in the warm partition. Therefore, the remaining storage space, $(1-\beta) \cdot P \cdot B_w$, is consumed by GC_{hot}. Since each GC_{hot} sends $P \cdot \alpha$ number of pages into the warm partition, $(1-\beta) \cdot P \cdot B_w/(P \cdot \alpha)$ number of GC_{hot} are required until the page p meets GC_{warm} on the warm partition. Before the page p is evicted to the MLC region, it should meet $N \cdot (1-\beta) \cdot B_w/\alpha$ times of GC_{hot}, and there should be $B_h \cdot P \cdot N \cdot B_w \cdot (1-\beta)/\alpha$ number of page writes on the hot partition.

Consequently, we can say that a cold page can stay at the SLC region while $B_h \cdot P \cdot N \cdot B_w \cdot (1-\beta)/\alpha$ number of pages are written. This is called the SLC staying time of the cold data. Therefore, if the mean-time-between-writes (MTBW) of data is shorter than the SLC staying time, the data is warm data. Otherwise, it is cold data.



(a) Size-based Hot/Cold Separation & N-Chance (GCLOCK) Algorithm

(b) 2Q-based Hot/Cold Separation & N-Chance (GCLOCK) Algorithm

Fig. 6. Page state transition diagrams.

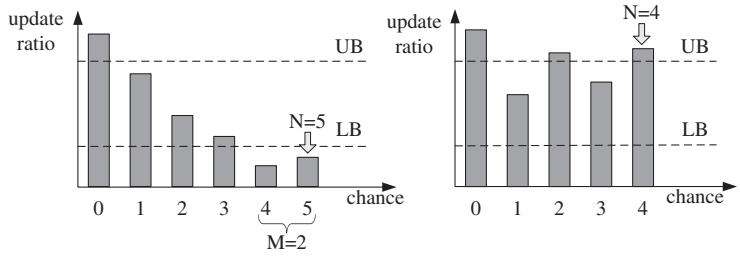
4.2. Dynamic adaptation for N

The value of N should be adjusted based on the locality of workload. If there are many warm data, many chances should be provided. However, if there is little warm data, a small value for N is preferred so as to use the SLC buffer efficiently. The optimal value of the maximum chances for warm data can be estimated by observing the update ratio in each warm set. If most of the pages which have exhausted all the N number of chances are not updated, the number of maximum chances should be decreased. If there are many updates on the pages in W_N, the number of maximum chances should be increased. Thus, we maintain the value of N such that the following two conditions are satisfied. $\varphi(W_k)$ represents the update ratio of the pages in the warm set W_k.

$$N - M < \exists k \leqslant N, \varphi(W_k) \geqslant LB \quad \text{and,} \tag{1}$$

$$\varphi(W_N) \leqslant UB \tag{2}$$

The meaning of first condition is that there should be at least one warm set whose update ratio is larger than the lower bound (LB) among the higher M number of warm sets, $W_{N-M+1}, \cdots, W_{N-1}), W_N$. The value of M is called the observation window. For example, in Fig. 7a, if $N = 5$ and $M = 2$, the first condition is not



(a) Too large number of chances (b) Too small number of chances

Fig. 7. Dynamic N adaptation.

satisfied because both $\varphi(W_4)$ and $\varphi(W_5)$ are smaller than the lower bound. Therefore, we should decrement the value of N. If we change N to 4, the first condition is satisfied. The meaning of second condition is that the update ratio of W_N should be smaller than the upper bound (UB). In Fig. 7b, the second condition is violated when $N = 4$, hence the maximum number of chances should be incremented.

4.3. Cold data early migration

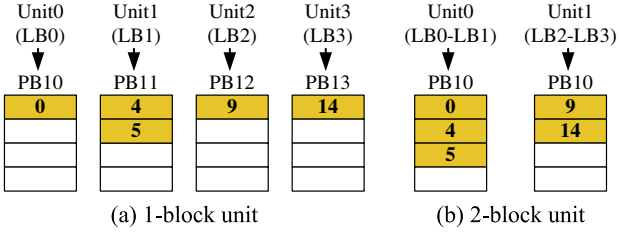
Using the proposed N-chance algorithm, the warm data can remain in the SLC buffer. However, the true cold data is also given N number of chances before it is evicted to the MLC region. Then, it occupies the expensive SLC storage unnecessarily. Using the dynamic adaptation technique for N chances, we can cope with such a case. However, it is better for cold data to migrate to cold region before it exhausts all the N chances.

For the early migration, the N-chance algorithm uses *warm* bit. When a page is written at the hot partition, its warm bit is initialized to '0'. When the page is moved from the warm partition to the hot partition (i.e., the page is updated while it is in the warm partition), its warm bit is changed to '1'. When a page is in the warm set $W_{N/2}$, if its warm bit is '0', the page is evicted to the cold region. Therefore, $W_{N/2+1}, \cdots, W_N$ have only the pages which have been updated within the warm partition.

5. MLC region management

5.1. Unit size of MLC region

As explained in Section 3, the MLC region is managed by unit-based page-level mapping. As the unit size decreases, the required memory space for mapping information also decreases. Considering this relationship, it is profitable to use a small-sized unit to save memory space. If there is a significant locality, the single-block unit is feasible. However, it is better to use multi-block unit for SLC/MLC combined flash since the MLC region contains only cold data. A small-sized unit may have many valid pages in the victim block causing a high GC cost compared to a large-sized unit as shown in Fig. 4. In addition, if the locality of the workload is low, utilization of blocks under the small-sized unit may be low compared to that of the large-sized unit, as shown in Fig. 8. Low utilization invokes frequent GCs.



(a) 1-block unit (b) 2-block unit

Fig. 8. Block utilization depending on the unit size.

Since the locality of workload in the MLC region is determined by the SLC region size, we select the unit size based on the SLC region size at the design time. The unit size is not adjusted during the run time in this work. However, it will be better to adjust the unit size depending on the locality of the workload at run time, which will be considered in future work.

5.2. Detection of hot data in MLC blocks

Large-sized data are generally cold data and are thereby sent to the MLC region directly. However, this means that hot data can be sent to the MLC region if it has a large size. Hence, it is required to identify hot data in the MLC region and to move them into the SLC region.

Hot data is detected based on the number of write requests for the data during a predetermined period. We count only the write requests since the read-intensive data does not need to be in the SLC region, which provides read performance similar to that of the MLC region. We can consider two kinds of hot data detection techniques, hot page detection (HPD) and hot unit detection (HUD), depending on the detection granularity. While the HUD technique requires a small management overhead, the HPD technique requires a large overhead but may have good accuracy. The HUD technique can give better results when there are high spatial localities. Therefore, detection granularity should be selected depending on the overhead and workload pattern.

ComboFTL uses a utilization-based hot unit detection technique which checks whether the following condition is satisfied:

$$N_{\text{count}} = N_{\text{allocated}} + N_{\text{invalid}} > \delta$$

where $N_{\text{allocated}}$ is the number of allocated pages, N_{invalid} is the number of invalid pages in a logical unit, and δ is the hotness threshold. When a write request is sent to a unit, it will increase the value of $N_{\text{allocated}}$. In addition, if the request updates the pages in the unit, the value of N_{invalid} is also increased. Therefore, the previous condition is designed to weight the update requests.

If a logical unit is detected as a hot unit, then the following write requests for the unit are sent to the SLC region. The N_{count} is reduced by half at every decay period so as to consider only the recent history.

The threshold value of δ is adjusted based on the hit ratio of the HUD technique. The hit ratio means how many units determined to be hot by HUD are updated at the SLC region before they are

Table 3
The characteristics of IO traces.

Trace	Address space (MB)	Total write amount (MB)
PhoneTrace	2036	10,475
PCTrace	2005	7837
IOzoneTrace	1416	18,640

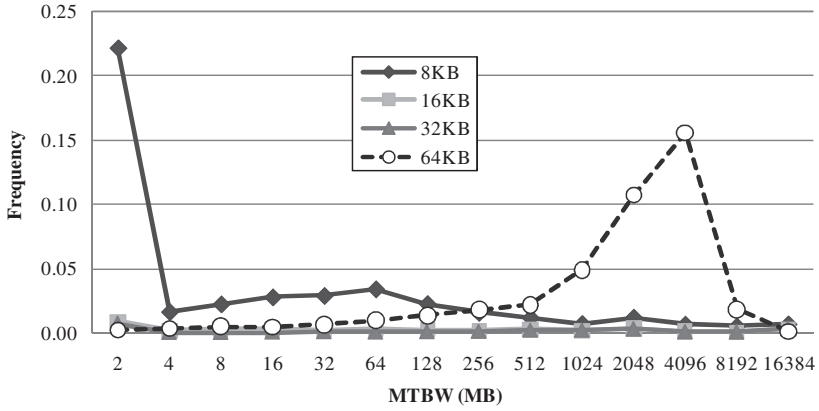
re-evicted to the MLC region. If the hit ratio is too low, the value of δ is increased, i.e., a tighter threshold is used. If the hit ratio is quite high, the value of δ is decreased to select more hot units.

For HPD, we used a two-bit access counter for each page to reduce counter overhead. When a write request for a page comes to ComboFTL, the request is sent to the SLC region if the corresponding counter value is larger than 1. The access counters are also reduced by half at every decay period.

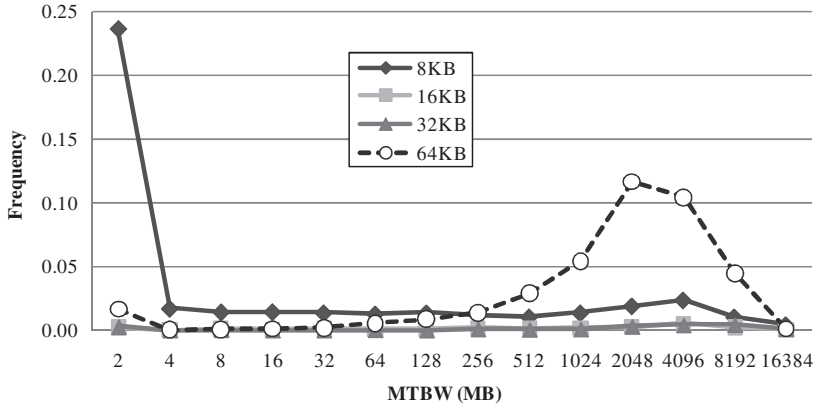
If the update period (MTBW) of a page is larger than the SLC staying time, it is useless to send the page into the SLC region since it will be evicted back to the MLC region without updates. Therefore, we should count only recent accesses using the decaying policy for the counter value. The decay period is determined based on the SLC staying time of a cold page.

6. Experiments

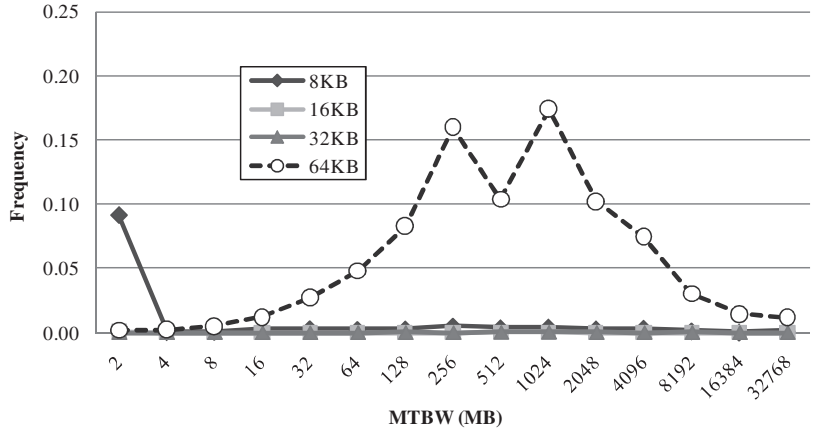
In order to evaluate the efficiency of the proposed techniques, we implemented an in-house simulator for SLC/MLC combined flash memory. The simulator counts the numbers of read, program and erase operations for SLC and MLC blocks during the trace



(a) PhoneTrace



(b) PCTrace



(c) IOzoneTrace

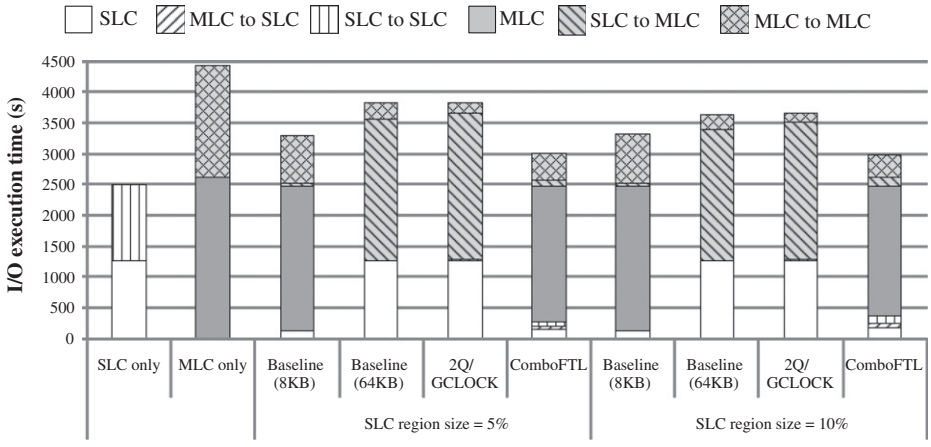
Fig. 9. The distributions of sizes and MTBWs of write requests.

Table 4
FTL schemes comparison.

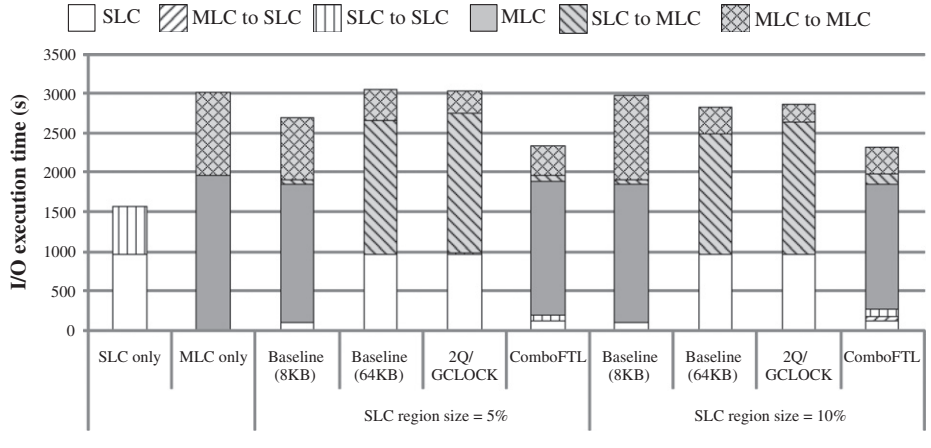
Schemes	Baseline	2Q/GCLOCK	ComboFTL
Hot/cold separation	Static threshold (8KB, 64KB)	2Q	Dynamic threshold
SLC → MLC	1-Chance (CLOCK)	*N*-chance (GCLOCK)	*N*-chance (GCLOCK) Early migration
MLC → SLC	None	Hot data detection	Hot data detection

simulation. The timing parameters of the flash memory used in experiments are those shown in Table 1. The page size is 4KB at both SLC mode and MLC mode. A block has 64 pages (256KB) and 128 pages (512KB) at SLC mode and MLC mode, respectively. The flash memory chip consists of 5120 number of flash blocks. Therefore, the total capacity of the flash chip varies between 1.2GB and 2.4GB depending on the number of SLC mode blocks.

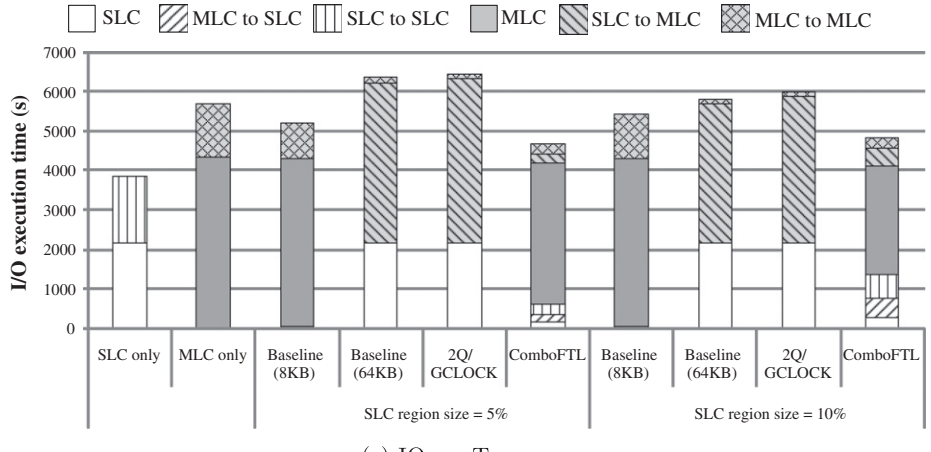
We used three workloads for simulator inputs. The traces are collected on block level with the blktrace or diskmon tool. We



(a) PhoneTrace



(b) PCTrace



(c) IOzoneTrace

Fig. 10. Performance comparison of ComboFTL and other techniques.

focused on only the write requests since the read latencies for SLC-mode block and MLC-mode block are similar. PhoneTrace and PCTrace are real I/O traces collected from a mobile phone and desktop PC, respectively. PhoneTrace are collected under the FAT32 file system executing real mobile phone applications such as SMS, PIMS, media player, game, and web-browsing at the Qtopia Phone Edition [24]. PCTrace are extracted from a Microsoft Windows XP-based desktop PC executing several applications such as document editing, spread sheet, web-browsing, media player, and game. IOzoneTrace is collected by executing the IOzone benchmark program with the option of '-automatic'.

Table 3 shows the total address space and the total write amount of each workload. Fig. 9 illustrates the write request pattern of each workload. In PhoneTrace and PCTrace, there are many requests smaller than 8 KB and most of them are updated before host sends 2MB of data to the storage, i.e., MTBWs (mean-time-between-write) are smaller than 2MB. The MTBWs of the write requests larger than 32KB are longer than 1024MB. Therefore, the hotness/coldness of data is closely related to the request size. IOzoneTrace has many large-sized update requests, thus it has high temporal and spatial localities.

To show the effectiveness of ComboFTL, we performed six kinds of experiments. First, we compared the performance of ComboFTL with other management schemes. Second, we observed the performance change over varying the SLC region size of combined flash memory. Third, we estimated the effects of various techniques of ComboFTL by disabling one of the proposed techniques. Fourth, we examined the dynamic adaptation feature of ComboFTL. Fifth, we measured the erase count of MLC region to know how much it can prolong the lifespan of flash memory chip. Lastly, we evaluated the effect of MLC region unit size.

6.1. Performance evaluation

We compared the performance of ComboFTL with those of FTLs which use baseline techniques and 2Q/GCLOCK techniques as shown in Table 4. The 2Q/GCLOCK scheme uses the 2Q algorithm to identify cold data as explained in Fig. 6b. We also measured the performances of ComboFTL using the SLC-only and MLC-only flash chips, where there are no page migrations between the SLC and MLC regions. Since the storage space of the SLC-only chip is half the space of the MLC-only chip, we used twice the number of blocks for the SLC-only chip to cover the entire workload. Since the read performances of SLC flash and MLC flash are similar, we considered only write performance.

Each technique was tested under two configurations of SLC region size, i.e., 5% and 10% of total flash blocks are configured as SLC mode to be used for the write buffer, respectively. The flash memory device provides only 80% of the available storage as a logical storage space for the file system. The remaining 20% is used for the SLC region or the over-provisioning area for MLC region garbage collection. We assumed that the logical storage space is occupied with data before measuring the I/O performance. For ComboFTL, the HUD technique was used for hot data detection since HUD was better than HPD in exploiting the spatial locality of data.

In Fig. 10, a total execution time comprises six types of flash writes: writes from host to SLC region (SLC), migrations from MLC region to SLC region by hot data detection (MLC to SLC), migrations within SLC region (SLC to SLC), writes from host to MLC region (MLC), migrations from SLC region to MLC region (SLC to MLC), and migrations within MLC region (MLC to MLC).

The relative execution times of ComboFTL are 0.67–0.85 and 1.2–1.49 compared to the execution times using MLC-only chip and SLC-only chip, respectively. Considering that the number of blocks in SLC-only chip is twice that of a combined chip, it can be said that ComboFTL enhances the performance of the combined flash chip close to that of SLC-only chip. ComboFTL shows better performances than do the baseline techniques since it decreases page migrations from the SLC region to the MLC region by increasing migrations within the SLC region. Compared to ComboFTL, the baseline technique with 8 KB threshold has a larger MLC region garbage collection cost. The baseline technique with 64 KB threshold and the 2Q/GCLOCK technique invoke large amounts of SLC-to-MLC migrations since many cold data are written to the SLC region. As explained at Section 2, MixedFTL and FlexFS also send all write requests to the SLC region like 2Q/GCLOCK. Therefore, we can expect the performances of MixedFTL and FlexFS will be worse than that of ComboFTL due to a large amount of SLC-to-MLC data migrations.

ComboFTL shows similar performances irrespective of the SLC region size since it can fully utilize the SLC region with the proposed *N*-chance algorithm. However, the baseline techniques show irregular patterns. For example, when the baseline technique uses 8 KB of static threshold θ, the combined flash memory with a larger SLC region gives worse results since it has a smaller MLC region. When θ is 64KB, the baseline technique using IOzoneTrace shows even worse results than the MLC-only chip since many cold data are written at the SLC region, thus increasing page migrations from the SLC region to the MLC region.

6.2. The effect of SLC region size

Fig. 11 shows the performance changes while varying the SLC region size. As the portion of SLC region increases, the write cost of SLC region increases and the write cost of MLC region decreases. However, when the SLC region is too large, the I/O cost of MLC region increases due to the frequent garbage collection in the MLC region. Consequently, the total I/O cost is minimized when the portion of SLC region is 15%. Therefore, we should determine the optimal SLC region size considering both the SLC region cost and MLC region cost. In addition, the balance between the wear counts of SLC region and MLC region should be also considered when determining the optimal SLC region.

6.3. The effect of each technique

To show the effect of each technique used in ComboFTL, we evaluated several incomplete versions, which disable one of several techniques of ComboFTL as shown in Fig. 12. Disabling the early migration imposes little performance degradation because the dynamic hot/cold separation technique and the dynamic warm set adaptation technique effectively eliminate cold data from the SLC region. The 8KB of static threshold version shows also similar results to the complete version since the request size is closely

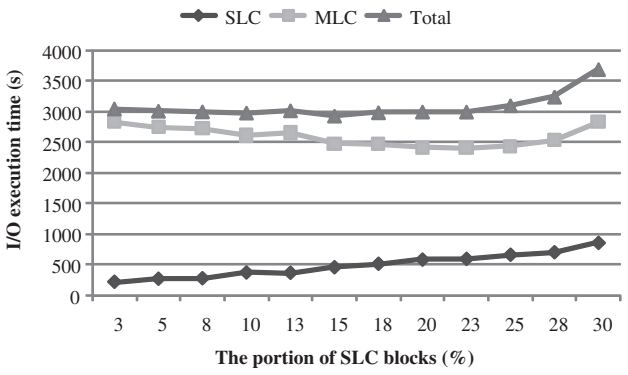
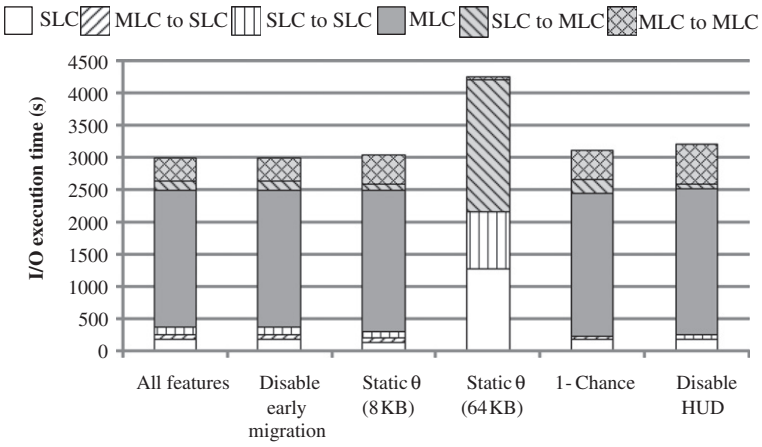
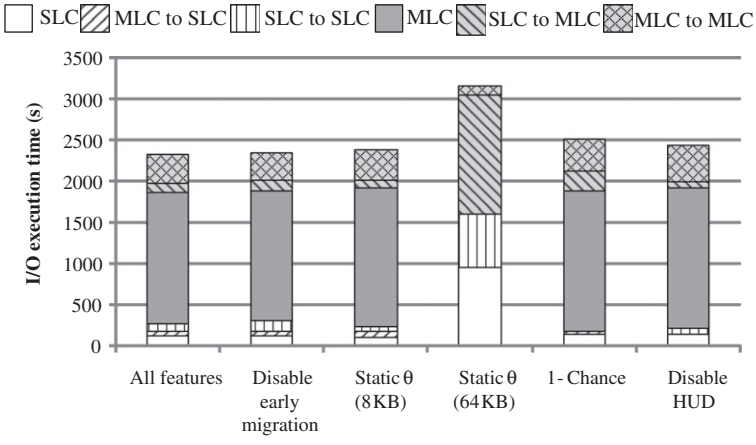


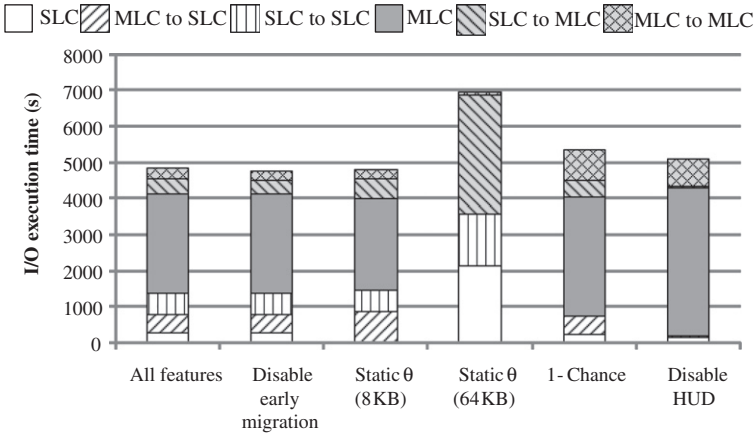
Fig. 11. ComboFTL performance while varying SLC region size (PhoneTrace).



(a) PhoneTrace



(b) PCTrace



(c) IOzoneTrace

Fig. 12. Performance comparison of incomplete ComboFTL versions.

related to the hotness of data in the traces. However, it writes more data to the MLC region and thus increases the MLC region garbage collection cost slightly. When the 64KB of static threshold instead of dynamic threshold is used, too many data are written to the SLC region, thus too many migrations from the SLC region to the MLC region occur. For all traces, the HUD-disabled version shows significantly degraded performance since it increases the write costs of the MLC region. For IOzoneTrace, if we use the one-chance algorithm instead of the *N*-chance algorithm, the garbage collection cost within the MLC region increases since many warm data are updated at the MLC region.

6.4. Dynamic adaptation of θ and N

We also evaluated how well ComboFTL adjusts the threshold value for hot data (θ) and the number of warm sets (N) dynamically

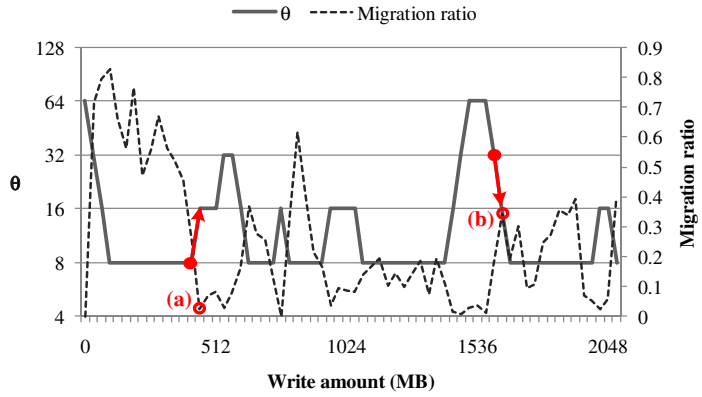


Fig. 13. Dynamic adaptation of hot data threshold θ (PhoneTrace).

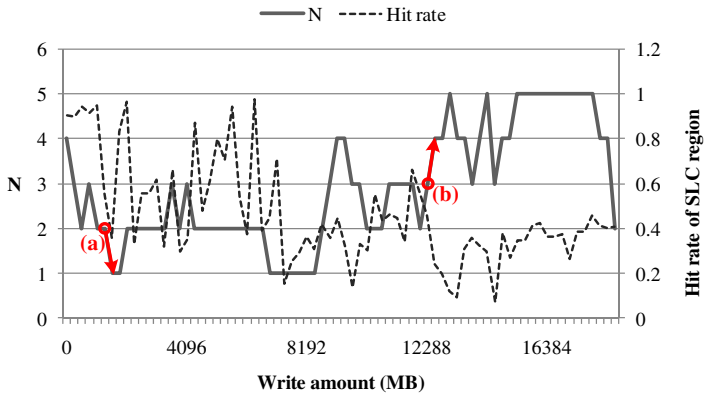


Fig. 14. Dynamic adaptation of the number of warm set N (IOzoneTrace).

depending on workload change. Fig. 13 shows the change of θ and page migration ratio from the SLC region to the MLC region for PhoneTrace. At the point marked with (a), the page migration ratio is 0.02. Therefore, ComboFTL increased the value of θ because the migration ratio is smaller than $\phi - \epsilon(= 0.5)$. However, at the point marked with (b), the page migration ratio is 0.34, which is larger than the upper bound for the page migration ratio $\phi + \epsilon(= 0.15)$. Therefore, the value of θ is decreased.

Fig. 14 shows the changes of N (the number of warm sets) and the overall hit ratio of SLC region for IOzoneTrace. Table 5 shows the hit ratio of each warm set at two points marked with (a) and (b) in Fig. 14. The observation window M is 2 and the upper bound (UB) and lower bound (LB) are 0.7 and 0.3, respectively. ComboFTL decrements the value of N since both the hit ratios of W_1 and W_2 are smaller than LB at (a), but increments it at (b) since the hit ratio of W_N is larger than UB.

6.5. Lifespan evaluation

One of the aims of SLC/MLC combined flash memory is to prolong the lifespan of flash memory. Since ComboFTL exploits the SLC region efficiently, it can reduce the erase count of the MLC region significantly thus increasing the lifespan of flash memory as shown in Fig. 15. The graph shows the erase counts of MLC region normalized by that of the MLC-only chip. ComboFTL reduces the erase counts by 13–20% and 20–27% compared to the MLC-only chip when the SLC region is 5% and 10%, respectively. As we allocate more blocks to the SLC region, the erase count of MLC region decreases.

Compared to the baseline technique with 8KB threshold, ComboFTL reduces the erase count by 10–25%. The baseline technique shows even worse results than the MLC-only chip when the portion of SLC region is 5% since it cannot utilize the SLC region efficiently and should manage fewer MLC blocks than the MLC-only chip. Consequently, we can say that ComboFTL elevates the advantage of SLC/MLC combined flash memory in terms of flash memory lifespan.

Table 5
The hit ratio of each warm set at two points marked with (a) and (b) in Fig. 14.

	$\varphi(W_0)$	$\varphi(W_1)$	$\varphi(W_2)$	$\varphi(W_3)$
(a) $N: 2 \rightarrow 1$	0.72	0.21	0.28	
(b) $N: 3 \rightarrow 4$	0.38	0.24	0.32	0.98

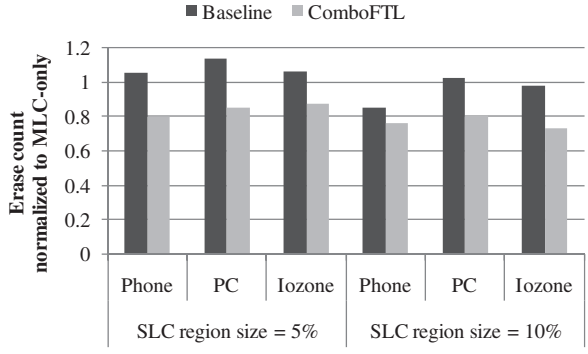


Fig. 15. Comparison of MLC region erase counts.

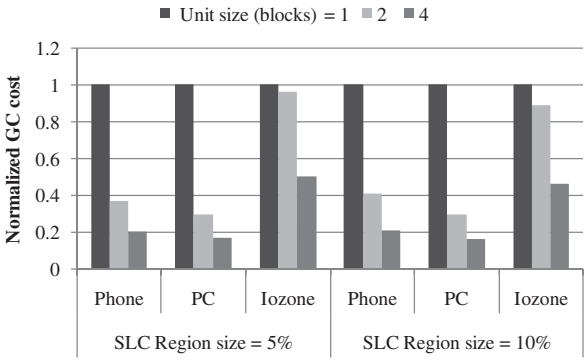


Fig. 16. MLC region garbage collection cost varying unit size.

6.6. Locality in MLC region

Finally, we evaluated the effect of MLC region unit size explained at Section 5. Fig. 16 shows MLC region garbage collection costs of ComboFTL while varying the unit size. Since the MLC region has only cold data, a large unit is profitable to increase the utilizations of MLC blocks, thus reducing the GC cost. If we increase the unit size, more memory space is required for address mapping since page-level mapping is used within a unit. Therefore, we should consider both GC cost and space overhead to determine the unit size.

7. Conclusion

SLC/MLC combined architecture can increase both I/O performance and lifespan of flash memory compared to the use of MLC-only flash memory. To maximize the advantage of such a hybrid architecture, it is important to efficiently exploit the SLC region of SLC/MLC combined flash memory. ComboFTL utilizes the SLC region for frequently updated data using a hot/cold data detection algorithm and the *N*-chance algorithm for cold data migration. It can adjust several policies to handle dynamic workload changes at run time. In future work, we will study how to determine the optimal size of the SLC region since it affects performance, lifespan, and storage capacity of SLC/MLC combined memory.

References

[1] A. Ban, Flash file system, United State Patent, No. 5,404,485, 1995.
[2] A. Kawaguchi, S. Nishioka, H. Motoda. A flash memory based file system, in: Proc. of the USENIX Technical Conf., 1995.
[3] J. Kim, J.M. Kim, S.H. Noh, S.L. Min, Y. Cho, A space-efficient flash translation layer for compact flash systems, IEEE Transactions on Consumer Electronics 48 (2) (2002) 366–375.
[4] S.-W. Lee, D.-J. Park, T.-S. Chung, D.-H. Lee, S. Park, H.-J. Song, A log buffer-based flash translation layer using fully-associative sector translation, ACM Transactions on Embedded Computing Systems 6 (3) (2007).
[5] J.-U. Kang, H. Jo, J.-S. Kim, J. Lee, A superblock-based flash translation layer for NAND flash memory, in: Proc. of EMSOFT'06, 2006, pp. 161–170.
[6] M. Bauer, A multilevel-cell 32MB flash memory, in: Proc. of the Solid-State Circuits Conference, 1995.
[7] S. Lee, K. Ha, K. Zhang, J. Kim, J. Kim, FlexFS: a flexible flash file system for MLC NAND flash memory, in: Proc. of USENIX Technical Conf., 2009.
[8] L.-P. Chang, Hybrid solid-state disks: Combining heterogeneous NAND flash in large SSDs, in: Proc. of Asia and South Pacific Design Automation Conference (ASP-DAC), 2008, pp. 428–433.
[9] F. Roohparvar, Single level cell programming in a multiple level cell non-volatile memory device, United State Patent, No. 7,366,013, 2007.
[10] T. Cho et al., A dual-mode NAND flash memory: 1-Gb multilevel and high-performance 512-Mb single-level modes, IEEE Journal of Solid-State Circuits 36 (11) (2001).
[11] Samsung Electronics. 4Gb Flex-OneNAND. <http://www.samsung.com/global/business/semiconductor/products/fusionmemory/Products_FlexOneNAND.html>.
[12] M. Wu, W. Zwaenepoel, eNVy: a non-volatile, main memory storage system, in: ASPLOS-VI: Proc. of the Sixth International Conference on Architectural Support for Programming Languages and Operating Systems, 1994, pp. 86–97.
[13] M.-L. Chiang, R.-C. Chang, Cleaning policies in mobile computers using flash memory, Journal of Systems and Software 48 (3) (1999) 213–231.
[14] M.-L. Chiang, P. Lee, R.-C. Chang, Using data clustering to improve cleaning performance for flash memory, Software Practice and Experience 29 (3) (1999) 267–290.
[15] L.-P. Chang, T.-W. Kuo, An adaptive striping architecture for flash memory storage systems of embedded systems, in: RTAS '02: Proc. of the Eighth IEEE Real-Time and Embedded Technology and Applications Symposium, 2002, pp. 187–196.
[16] J.-W. Hsieh, L.-P. Chang, T.-W. Kuo, Efficient on-line identification of hot data for flash-memory management, in: SAC '05: Proc. of ACM Symposium on Applied Computing, 2005, pp. 838–842.
[17] S. Lee, D. Shin, Y.-J. Kim, J. Kim, LAST: locality-aware sector translation for NAND flash memory-based storage systems, SIGOPS Operating Systems Review 42 (6) (2008) 36–42.
[18] S.H. Park, J.W. Park, J.M. Jeong, J.H. Kim, S.D. Kim, A mixed flash translation layer structure for SLC-MLC combined flash memory system, in: Proc. of SPEED'08, 2008.
[19] S. Im, D. Shin, Storage architecture and software support for SLC/MLC combined flash memory, in: SAC'09: Proc. of ACM Symposium on Applied Computing, 2009.
[20] J. In, H. Kim, K. Lee, T. Chung, Method of remapping flash memory, United State Patent, No. 7,516,295, 2009.
[21] F.J. Corbato, A paging experiment with the multics system, in: Honor of P.M. Morse, MIT Press, Cambridge, Mass., 1996, pp. 217–228.
[22] V.F. Nicola, A. Dan, D.M. Dias, Analysis of the generalized clock buffer replacement scheme for database transaction processing SIGMETRICS Perform, Evaluation Review 20 (1) (1992) 35–46.
[23] T. Johnson, D. Shasha, 2q: A low overhead high performance buffer management replacement algorithm, in: Proc. of VLDB Conf., 1994, pp. 297–306.
[24] Nokia Corp., Qtopia phone edition 4.1.2. <http://www.qtsoftware.com/products/>.

Soojun Im received the B.S. degree in computer engineering from Sungkyunkwan University, Korea in 2007. He is currently a Master student in the School of Information and Communication Engineering, Sungkyunkwan University. His research interests include embedded software, file systems and flash memory.

Dongkun Shin received the B.S. degree in computer science and statistics, the M.S. degree in computer science, and the Ph.D. degree in computer science and engineering from Seoul National University, Korea, in 1994, 2000 and 2004, respectively. He is currently an Assistant Professor in the School of Information and Communication Engineering, Sungkyunkwan University (SKKU). Before joining SKKU in 2007, he was a senior engineer of Samsung Electronics Co., Korea. His research interests include embedded software, low-power systems, computer architecture, and multimedia and real-time systems.