KRAFT: K-associative Relative Addressing Flash Transition Layer

Dongho Kang, Jaeseok Park, Ilwoong Kim, Ingeol Lee, Sungho Kang
Yonsei Univ.
{fourier2, bbajaepjs, woong, keor}@soc.yonsei.ac.kr, shkang@yonsei.ac.kr

Abstract

In NAND flash based storage system, a software module between file system and NAND flash called Flash Transition Layer (FTL) is necessary. In this paper, a new FTL called "K-associative Relative Addressing FTL (KRAFT)" is proposed. The goal of this study is to achieve fast response time and high R/W performance. With an efficient merge policy and a novel addressing scheme, KRAFT reduces the merge cost drastically and utilizes twice as much log blocks. Simulation results show that KRAFT surpasses recent FTLs in term of response time and R/W performance.

Keywords: Flash Transition Layer, NAND Flash.

1. Introduction

NAND flash has been deployed as data storage for various computing systems because of the low power consumption and the high performance. However, NAND flash has no capability of overwrite, but it only provides page-unit program and block-unit erase. So, a time-consuming block-unit erase must be done before executing rewrite for a page. To relieve this weakness, a software module between file system and NAND flash called FTL is introduces. The purpose of FTL are mapping logical address to physical address in a way that is suitable to the characteristics of NAND flash, and optimizing its performance at the same time.

Recent FTLs can be classified into demand based FTL and log block FTL. Generally, log block FTLs show batter read performance because demand based FTLs have table-caching delay [1]. However, log block FTLs may suffer from inconsistent response time and limited log blocks [2] [3] [4]. This paper proposes a new log block FTL, K-associative Relative Addressing FTL (KRAFT), which provides fast response time and high R/W performance. With its efficient merge policy and a novel addressing scheme, KRAFT reduces merge cost significantly and utilizes much more log blocks.

2. KRAFT

KRAFT ensures fast response time and high R/W performance with a new merge policy described in Section 2-A, and it utilizes more number of log blocks than conventional FTLs with a relative addressing scheme as shown in 2-B.

A. Merge Policy

For the goal of this study, KRAFT focuses on the effective associativity control and the switch efficiency optimization. Algorithm 1 shows how write operation is performed and presents the merge policy of KRAFT.

Algorithm 1: Write

```
1:  Find log block associated with issued data block A_Log ;
2:  If (A_Log found)
3:    If A_Log is full)
4:      If (A_Log is Serial)  Switch A_Log ;
5:      Else Full-Merge A_Log ;
6:    Else /* L_Log found, Write without increasing associativity */
7:      If (LPN is 0)
8:        If (A_Log is Serial And switch cost for L_Log is low)
9:          Switch A_Log ;
10:         Find free Log F_Log ;
11:         Append  data to F_Log ;  /* make a new serial log */
12:       Else
12:         Append data to A_Log ; /* break seriality of A_Log */
13:     Else
14:       If (A_Log is Serial And Top < LPN And Fill cost is low)
15:         Fill empty pages between A_Log_Top and LPN
16:         Append  data to LSN ;  /* sustain seriality of A_Log */
17:       Else
12:         Append data to A_Log ;
18: Else /*No Log is associated with issued data block*/
19:   Find lowest associative log L_Log ; /* balancing associativity */
20:   If (lowest associativity ≥ K ) /* K : the maximum associativity */
21:     Select victim log V_Log
22:     If (V_Log is Serial)  Switch V_Log ;
23:     Else Full-Merge V_Log ;
24:   Else
25:     If (LPN is 0)
26:       If (L_Log is Free) Append data to L_Log ; /* a new serial log */
27:       Else Append data to L_Log ;
28:     Else
29:       If A_Log is Free And Fill cost is low)
30:         Fill empty pages between 0 and LPN with valid data ;
31:         Append data to L_Log ; /* sustain seriality of A_Log */
32:       Else
31:         Append data to L_Log ;
```

- ***Associativity*:** KRAFT is based on full associativity [2] which provides high space utilization of log blocks but incurs a huge merge cost because too many data blocks can be associated with a log block. So, KRAFT limits the number of data blocks associated with a log block to a user defined value *K*, and it writes data to the log block already associated with the issued data block. If such a log block cannot be found, log block with the lowest associativity is selected. The recommended *K* is 8 to 16 which ensures fast response time and high space utilization at the same time.

- ***Switch operation*:** If a log block is serially written from page 0 to the top, KRAFT can perform a switch operation instead of full-merge operation [3]. The merge operations of KRAFT are shown in Figure 1.

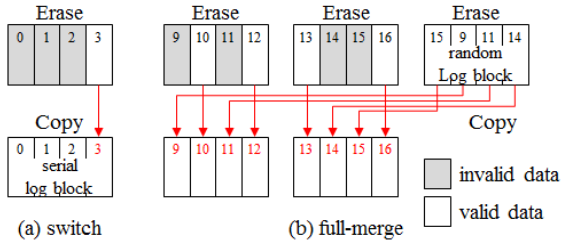

Figure 1: Merge operations in KRAFT

In switch operation, KRAFT copies valid pages from the associated data block to the log block. In full-merge operation, however, KRAFT copies valid pages from multiple blocks to the newly located blocks. The costs of each operation are compared in Table 1.

Table 1: Cost comparison of switch and full-merge

Operation	Cost
Switch	$Tc \times v\_page + Te$
Full-merge	$(Nd + 1) \times (Tc \times v\_page + Te)$

T_e: time for erase
T_c: time for copying a page
v_page: pages with the valid data
N_d: number of associated data block

To maximize this switch efficiency, KRAFT manages log blocks to sustain their seriality as much as possible as shown in Algorithm 1.

- ***Victim Selection*:** If the lowest associativity hits the limit, *K*, KRAFT selects a victim log block and performs a merge operation. Since the switch operation costs less time than full-merge operation, KRAFT selects the mostly filled serial log block. If no serial log block is found, a random log block is selected according to LRU algorithm because less recently used block is likely to holds less valid pages.

B. Relative Addressing

The performance of log block FTL highly depends on the number of log blocks it can utilize. However, conventional log block FTLs have suffered from the lack of log blocks because the log space is managed by a RAM-consuming page-level mapping.

In case of KRAFT, it utilizes more log blocks through a relative addressing scheme. KRAFT holds mapping information of each sector as a relative logical sector number (RLSN) with a reference index (RI). RI represents which logical block number (LBN) to reference, and RLSN indicates the relative value, respectively. To use this scheme, the number of references must be limited. Fortunately, as described in 2-A, KRAFT restricts the associativity of log blocks, so the relative addressing in each log block is possible. The structure of the mapping table is shown in Figure 2.

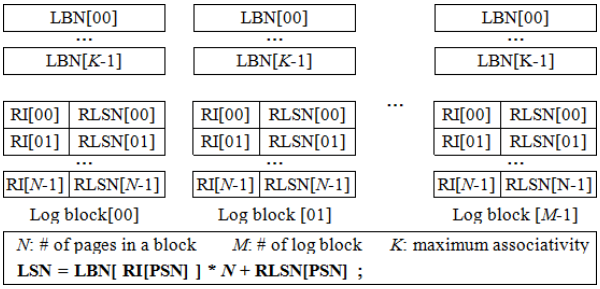

Figure 2: Log mapping table of KRAFT

The maximum number of LBNs is *K*, and the range of a RI and a RLSN are only 0 to *K*-1, and 0 to *N*-1, respectively. So, this table occupies significantly less space of RAM compared to the conventional mapping table which holds physical sector address in absolute value. This addressing scheme introduces extra RAM access time for acquiring references, and ALU time for calculating absolute address, but they are negligibly small compared to the NAND flash R/W time. Figure 3 shows an example of accessing processes for PSN=71, PSNN=82. The value of K is 4, and the flash block is composed of 8 pages in this example.

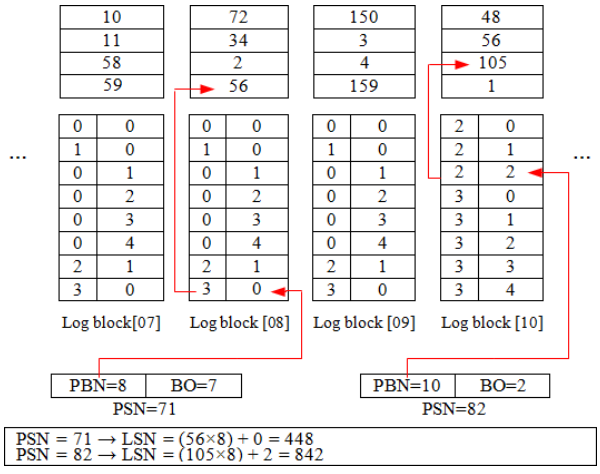

Figure 3: Relative addressing access process

In the example of Figure 3, KRAFT took access request for PSN 71(= 8×8+7). By looking up the 7th entry of table 8 (RI = 4), KRAFT found the reference, the 4th LBN of table 8, and run simple calculation to find absolute address of logical sector, 448. LSN stored in PSN 82 can be found in the same manner.

3. Experimental Results

In this section, experiments are conducted to evaluate the performance of KRAFT. In the simulation, a 64GB MLC NAND flash [5] is used, and OLTP, Web traces from the SPC [6] are applied to compare FAST [2], KAST [3], FFTL [4]. OLTP1, OLTP2 are write-dominant (more than 50% of request is write) and Web is read-dominant (less than 7% of request is write). FAST, KAST, and FFTL uses 128 log blocks while KRAFT utilizes 256 log blocks using the same size of RAM thanks to the relative addressing scheme.

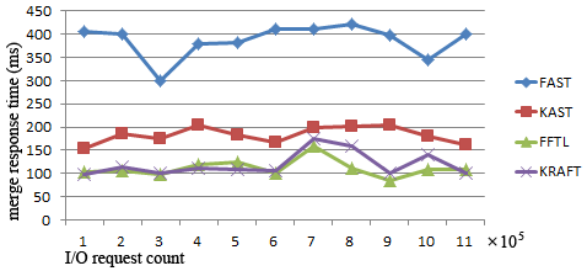

Figure 4: Merge cost comparison

Figure 4 shows the comparison of merge cost of each FTL on a write-dominant workload. According to Figure 4, KRAFT outperforms FAST and KAST in term of merge response time due to the remarkable associativity control and the switch efficiency. Though FFTL incurred the lowest merge cost, KRAFT also achieved very fast merge response time. It is shown that KRAFT has average 6.7% slower response time compared to FFTL, but the difference is negligible in occasional events like merge request.

Table 2: Performance comparison

Test Data		OLTP 1		OLTP 2		Web	
Metrics		MRG (ms)	LATE (µs)	MRG (ms)	LATE (µs)	MRG (ms)	LATE (µs)
FTL	FAST	386.8	278	341.1	175	956.8	61.1
	KAST	183.3	252	195.3	176	601.7	60.4
	FFTL	**111.6**	260	**106.2**	171.2	**591.9**	60.4
	KRAFT	119.9	**198**	121.8	**151.1**	600.3	60.4

LATE: Average latency of R/W MRG: Average merge response time

According to Table 2, KRAFT surpasses the previous works in term of R/W performance. KRAFT executes merge operation less frequently by utilizing more number of log blocks. So, its write performance is especially outstanding. However, any supremacy of KRAFT can be found in read-dominant Web trace because most log block FTLs has very high read performance. Nevertheless, KRAFT is average 14% faster than previous works in term of R/W performance, and it also reduces the time cost for each merge operation nearly as much as FFTL.

4. Conclusion

In this paper, a new log block FTL called KRAFT is proposed. KRAFT has unique features that ensure fast response time and high R/W performance. KRAFT controls the associativity of log blocks and maximizes switch efficiency through the merge policy. It also makes advantage of more number of log blocks with a novel address mapping scheme. Experimental results show that KRAFT has fast response time like FFTL and surpasses concurrent FTLs in R/W performance.

As a future work, an adaptive seriality management scheme is being researched. If KRAFT tries to sustain the seriality of a log block to which random data is being written, this effort can degrade the performance. So, KRAFT must aware the possibility of serial input according to the previous input pattern to enhance its performance.

Acknowledgement

This work was supported by the National Research Fou ndation of Korea(NRF) grant funded by the Korea gov ernment(MSIP) (No. 2015R1A2A1A13001751).

References

[1] Z. Xu, R. Li, and C.-Z. Xu, “CAST: A page-level FTL with compact address mapping and parallel data blocks,” in IEEE 31st IPCCC, Dec. 2012, pp. 142–151.
[2] Lee S., D. Park, T. Chung, D. Lee, S. Park, and H. Song . “A Log Buffer-based Flash Translation Layer Using Fully-Associative Sector Translation,” IEEE Transactions on Embedded Computing Systems, Vol.6 ,2007,pp.18~es.
[3] H. Cho, D. Shin, Y. Ik Eom "KAST: K-associative sector translation for NAND flash memory in real-time systems" Proc. of DATE '09, pp. 20-24 April 2009.
[4] S.J. Kwon, H.-J. Cho, and T.-S. Chung, "Fast responsive flash translation layer for smart devices," IEEE Trans. on Consumer Electronics, Vol. 60, No. 1, pp.52-59, Feb. 2014.
[5] S. Electronics, K9G8G08U0A MLC NAND, 2007.
[6] SPC. http://www.storageperformance.org.