International Journal of Computing and Applications | **ISSN: 0973-5704**
Volume 13, Number 2, (July-December 2018), pp 397-404

A COMPARATIVE STUDY OF FLASH TRANSLATION LAYER TECHNIQUES

Shailesh Kumar[1] and P.K. Singh[2]

[1,2]Computer Science & Engineering Department, Madan Mohan Malaviya University of Technology, Gorakhpur, India.
Email: [1]shailesh23jan@gmail.com, [2]topksingh@gmail.com

Abstract: Flash memory with characteristics such as lightweight, shock resistance and low power consumption will be better candidate to use as storage device in *IoT-based* appliances. However due to the erase and then write constraints, write operation is expansive. To avoid, frequent write and erase operation the flash layer translation technique has been adopted. In this paper we surveys the different FTL techniques on the base of six parameter, mapping table size, address computational overhead, read cost, write cost, erase cost, space utilization, and energy consumption.

Keywords: Flash Memory; Flash translation layer; Log block; Data block.

1. INTRODUCTION

There are basically two types of flash memory: NOR and NAND flash memory. The read speed of NOR flash memory is much more faster than NAND flash memory but the write speed and the erase speed of NAND flash memory is faster than NOR flash memory (Aritome, 2013). The bit density of NAND flash memory is higher than NOR flash memory. Therefore NAND flash memory prefers over NOR flash memory as secondary storage. Flash memory with characteristics such as shock resistance, lightweight and low power consumption will be better candidate to as storage device (Kuo, Chang, Huang, & Chang, 2008). The main concern with flash memory is read and write asymmetric speeds (J. Hu, Xue, Tseng, Zhuge, & Sha, 2010); out-place updates, and a limited lifetime. NAND flash memory write speed is approximately eight times slower than its read speed. One other drawback of flash memory is its erase than write constrain. Whenever it is required to update any page , the block that containing that page have to erase (Takeuchi, 2013). Write operation on flash memory is costly due to erase than write constrain (X.-Y. Hu, Eleftheriou, Haas, Iliadis, & Pletka, 2009). To avoid the frequent erase of flash memory blocks the flash layer translation techniques have been adopted. Flash layer translation is the layer of software which will map the logical address generated by cup with the physical address of flash memory. Table 1 (Li, Yang, & Tseng, 2008) shows the energy consumption of read, write and erase operation. Write and erase operation consumes more time as well as more energy. So minimization of write and erase operation will improve the performance of IoT-based appliances which uses flash memory as storage media.

Table 1
NAND Flash characteristics

Operation	*Time*	*Energy Consumption*
Read	47.2 μs	679 nJ
Write	533 μs	7.66 μJ
Erase	3 ms	43.2 μJ

2. RELATED WORK

In flash memory page overwriting is not possible so when write operation on the flash memory is issued by file system the requested page have to erase before performing write operation. In flash memory the write operation performs on page unit, but the erase operation can only perform on the block unit, which

contains numbers of pages. Erase cost of flash block, generally containing 32 pages, is very expensive. Erase operation of flash memory block takes approximately 2ms. To avoid, frequent write and the erase operation of the flash layer translation technique has been adopted(Qin, Wang, Liu, & Shao, 2012).

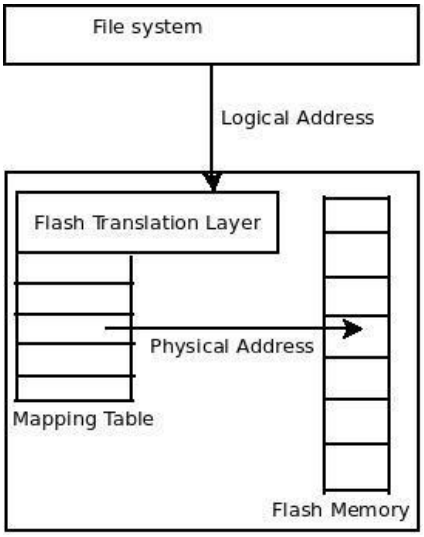

Figure 1: Architecture of Flash Memory System

Flash memory contains a number of blocks and each block contains a number of pages. In sector level mapping, the logical address of a sector is mapped to a physical sector address of flash memory (Amir Ban, 1995). Mapping table contains the logical sector address and its corresponding physical sector address. When sector update command is generated by file system, the FTL writes it to any empty location (sector) in flash memory and maintains mapping information in the mapping table (Ryu, 2011). If there is no empty sector (already erased) in flash memory then FTL selects a victim block, copy data of victim block to spare free block, update mapping information in mapping table and then erase that victim block which will then become spare block. In sector mapping, every logical sector is mapped with only one physical sector so mapping table contains the address information of every logical block and its corresponding physical block. This mapping table is saved in the flash memory and main memory as well, because mapping information is necessary to read sector later. For larger size flash memory the mapping table will be large and it will take large amount of memory.

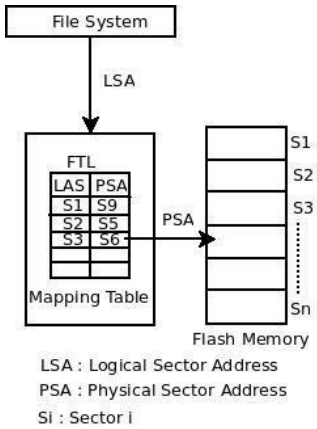

Figure 2: Sector Mapping

Block Mapping FTL

In sector mapping, mapping table takes bulk amount of memory. Block mapping is used to shrink the size of mapping table. It is similar to sector mapping in which logical block is mapped with physical block. Logical sector address is divided by block size of flash memory (Ban, 1999).

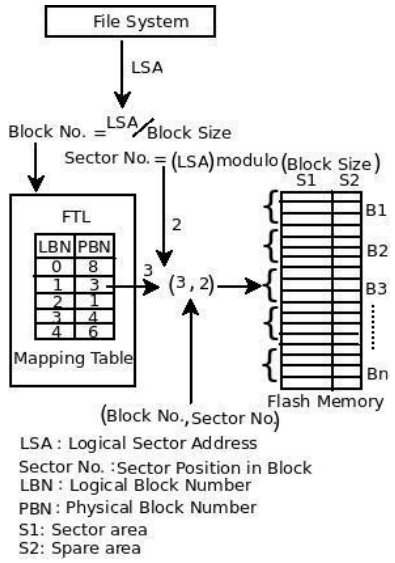

Figure 3: Block Mapping

The quotient of this division operation gives logical block number, which will be mapped to the physical block number by mapping table. The remainder of this division operation gives physical sector number in physical block of flash memory. This approach successfully reduces the mapping table size but the main disadvantage of this technique is that if the file system issues similar logical sector address it would be take many erase and copy operation.

Hybrid Mapping FTL

Hybrid mapping is the joined approach of sector mapping and block mapping (B. Kim, 2002). In this technique block mapping technique is used to get physical block number form logical sector address. After computing the physical block number, sector mapping technique is used to get free sector in this block, and write operation will be performed. Spare area of physical block contains the logical sector number (logical sector address) because this information is essential to read data later (Mittal & Vetter, 2015). For reading the data, after getting the physical block address form logical block address using mapping table, FTL algorithm scans logical sector number from spare area of the physical block and reads the data form sector area. This approach suffers higher read complexity then pervious approaches.

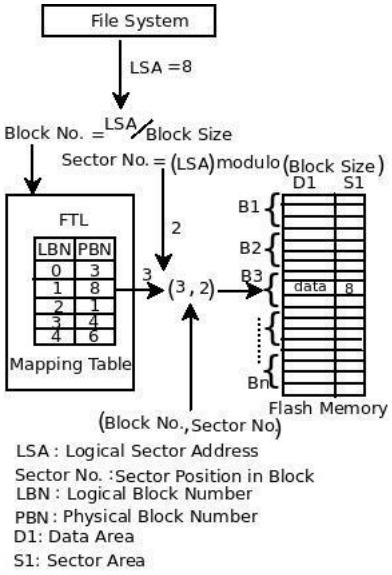

Figure 4: Hybrid Mapping

The Log block Based Hybrid FTL Approach

In log block approach the flash blocks are categorized in two categories log block and data block. Log block uses to store temporary data (J. Kim, Kim, Noh, Min, & Cho, 2002). Log blocks are free blocks which have been erased in advance and ready to write. Whenever file system issues a command to update a page of data block, this update command is redirected to log block. Log block method utilizes both block level mapping as well as sector level mapping (S.-W. Lee et. al., 2007).

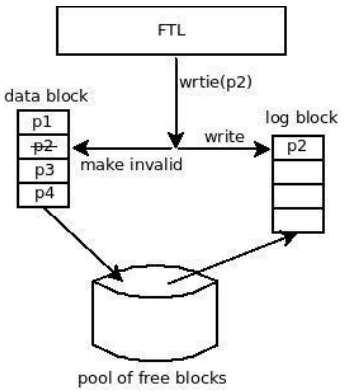

Figure 5: Log Block

3. CASE STUDY

Block Associative Sector Translation

In BAST scheme one log block is associated to one data block. Whenever write command is issued, one log block is allocated to this data block and this write operation is performed on log block from the very first sector (J. Kim et. al., 2002) (Kang, Park, Jung, Shim, & Cha, 2009). For example, if page p2 have to update then one log block is allocated to this data block and FTL writes p2 to very first empty location of the selected log block.

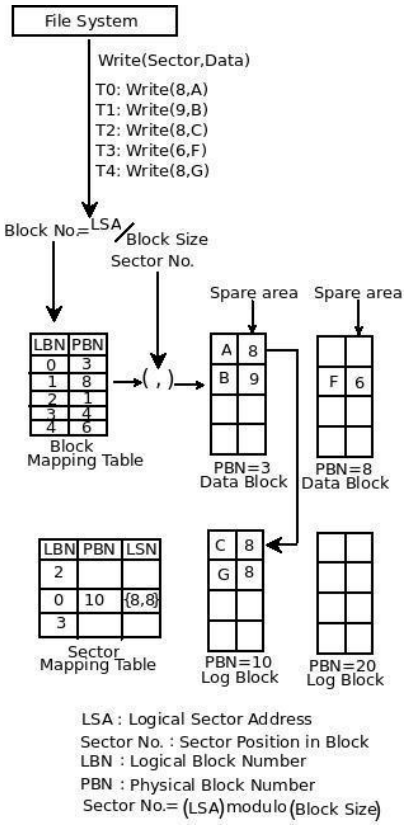

Figure 6: BAST

In Figure 6 let file system issued a command Write (8, A) at time T0, the block mapping technique has been used and data A has been written in Physical block number 3 (PBN = 3). Again file system at time T2 issued a command Write (8, C), means file system again want to write logical sector number 8 with data C. In this case the collision occurs in data block 3, so data is written using sector mapping in a log block (PBN = 10) which is allocated to the data block (PBN = 3) with offset 0. At time T4 again collision occurs due to command Write (8, G) and the data is updated to same log block number at the next empty location.

Fully Associative Sector Translation

In FAST (Sw Lee, Choi, & Park, 2006) scheme one log block is associated to many data blocks. By allowing one log block to multiple data blocks in FAST scheme the space utilization get increases as compare to BAST scheme. In Figure 7 when file system issues command Write (6, H) at time T5, the collision occurs in physical data block 8 (PBN = 8). In FAST two or more data block can share a single log block. The log block, which having physical block number (PBN=10), is shared between these two data block PBN 3 and PBN 8, so write operation is performed on log block (PBN = 10). When all the pages of the log block are consumed then the merge operation has to perform in both approaches BAST and FAST. Merge operation performs to create free blocks.

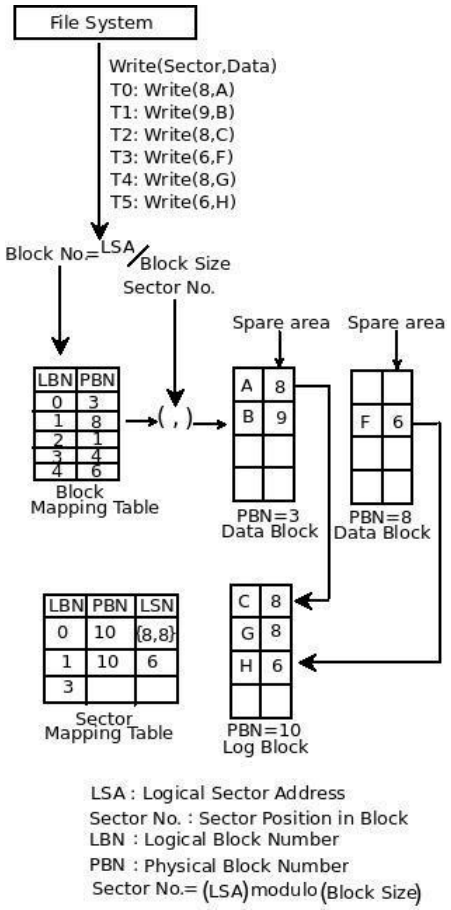

Figure 7: FAST

There are basically three ways of performing the merge operations:

- **Switch merge:** If file system issues command to update all pages of the data block in sequential manner then the write operation performed on log block will also be sequential and log block contains updated pages of data block in same sequence as it was in the data block. In this case, the log block is altered as data block the data block is erased and become free block.

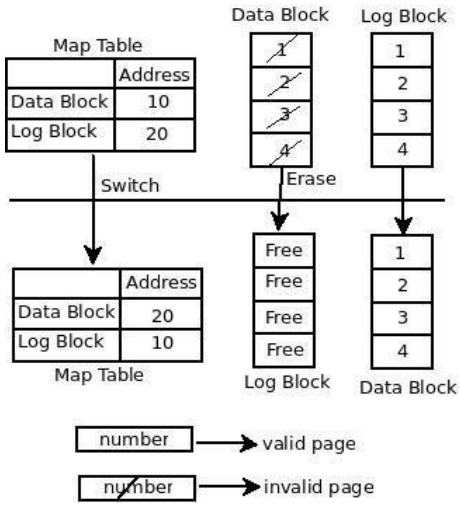

Figure 8: Switch Merge

- **Partial merge:** In partial merge the file system issues command to update contiguous subset of pages of data block in sequential manner. The log block contains some pages of data block in sequence as it was in data block. The pages which are not updated is copied to log block, log block is then changed as data block and data block is erased to become free log block (Chung & Hsu, 2014).

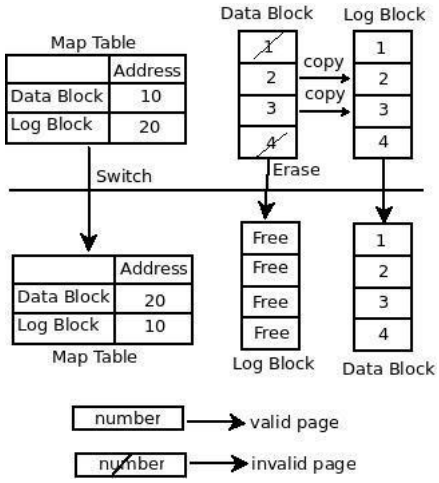

Figure 9: Partial Merge

- **Full Merge:** If file system issues command to update the pages of data block randomly, then the corresponding log block contains the updated pages of data block but not in sequence as it was in data block.

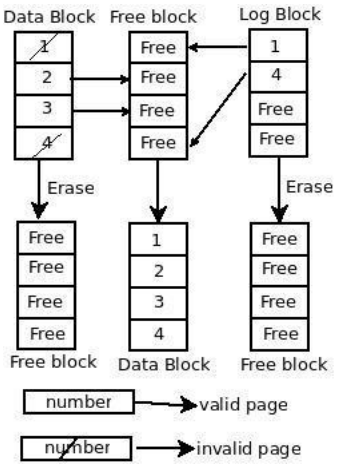

Figure 10: Full Merge in BAST

So in this merge operation one free block is allocated and valid pages of data block and updated pages of data block, which is in log block, will be copy to free block in sequence as it was in data block. Now this free block will be the new data block. After than the log block and original old data block are erased to create new free block (Sw Lee et. al., 2006).

Superblock FTL Scheme

Superblock approach overcomes the shortcomings of BAST and FAST by grouping the consecutive logical blocks into a superblock(Jung, Kang, Jo, Kim, &

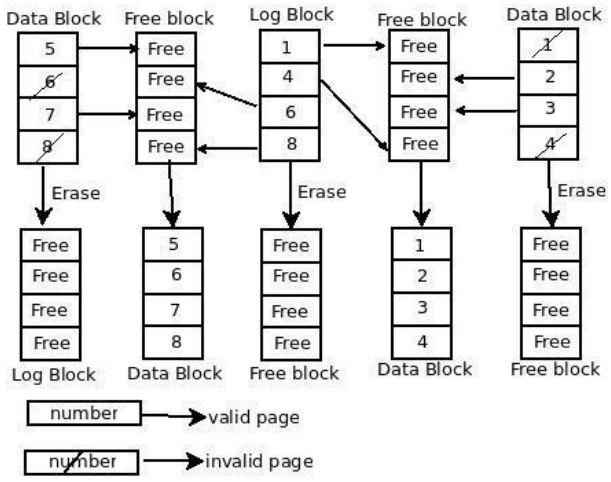

Figure 11: Full Merge in FAST

Table 2
Cost of Merge Operations

Merge Types		*Operation*	
		Write	*Erase*
Switch		Yes	No
Partial		Yes	1 Block Erase
Full	BAST	Yes	2 Block Erase
	FAST	Yes	3 Block Erase (if 2 data block shares 1 log block)

Lee, 2010). In superblock FTL group of consecutive logical blocks shares same number of log blocks to enhance the log block utilization and exploits spatial locality of reference. Superblock FTL uses a three level mapping technique (Lin, Chiao, & Chang, 2010). The SRAM stores the first block level mapping table. Spare area of superblock kept lower two mapping information. The limitation of this approach is that, spare area of superblock use to store error correction code, storing the last two level mapping information in spare area which will reduces space for error correction code. Superblock FTL is difficult to implement and performance is also affected due to OOB and extra searches.

Locality Aware Sector Translation

This scheme categorizes the log block into two category, sequential log block and random log block. Each log block in sequential log block group, is associated with a data block like BAST approach and each log block in random write block group, is link up with multiple

data block like FAST scheme (Sungjin Lee, Shin, Kim, & Kim, 2008). On each write request, locality detector detects type of write request. The LAST scheme separates the sequential write request and random write request and sends request to appropriate log buffer group. By separating write requests this approach reduces the cost of merge operation. Random write log block set is divided into hot blocks and cold blocks to increase temporal locality of reference.

Set Associative Sector Translation

SATA scheme (Park et. al., 2008) divides the flash block into two group, data block group and log block group. Data block group have N adjacent logical data blocks, which can share maximum K log blocks. So the maximum block associativity will be K for a log block. The optimal value of K and K is decided by behavior of workload. SATA stores the page level mapping table in SRAM so the read cost will be to less compare to superblock FTL. This approach also suffers with block thrashing and high block associativity problem.

Adaptive Set-Associative Sector Translation

This approach tries to overcome the shortcomings of SATA scheme. The size of data block group is not fixed and it can change adaptively, according to update pattern, to enhance the spatial locality of reference (Wu, Lin, & Kuo, 2010).

K-Associative Sector Translations

This approach does not create a group of log blocks and data blocks as SAST approach (Cho, Shin, & Eom, 2009). KAST only limits the associativity of all log blocks. The maximum associativity of all log blocks cannot be more than K. The value of K decides the performance of KAST because for very small value of K KAST suffers with block thrashing problem as BAST, and for very large value of K this approach behave like FAST scheme (Cho et. al., 2009).

4. FUTURE RESEARCH DIRECTIONS

The research presented in this paper raised some very important research areas where further research should be pursued. There are several issues which still to be resolve.

- Hybrid mapping resolve the huge mapping table issue associated with sector mapping and same sector frequent update issue associated with block mapping. While resolving issues of sector mapping and block mapping, the hybrid mapping suffers the read cost problem. The spare area of all the pages of any flash memory block have to scan first to read any page. In future research can be done to minimize read cost associated with hybrid mapping.
- Log block approach also uses the hybrid mapping technique. The limitation of log block technique is space utilization so further research work can be done to maximize space utilization in log block approach.

5. CONCLUSION

This paper surveys the different FTL algorithms. In this paper we categorize the FTL algorithms into four categories, sector mapping, block mapping, hybrid mapping and log block based hybrid mapping. The sector mapping technique seems better than other approach, but the size of mapping table is huge so this approach requires a large memory to store mapping table. Block mapping technique have small mapping table but this approach have large amount of erase operation when same sector number is updated frequently. The hybrid mapping successfully overcomes the same sector frequent update problem associated with block mapping. Read cost of hybrid mapping technique is more than sector mapping and block mapping because for reading any page the spare area of block has to be scan. The limitation of log block based hybrid mapping is space utilization. NAND flash characteristics table shows the erase operation takes maximum energy and the technique which has less erase operation will consume less energy. Comparison table of different FTL scheme shows the difference between these schemes based on the parameters, mapping table size, address computational overhead, read cost, write cost, space utilization and energy consumption.

Table 3
Comparison table of different FTL scheme

Technique	*Mapping Table Size*	*Address Computational Overhead*	*Read Cost*	*Erase Cost*	*Space Utilization*	*Energy Consumption*
Sector Mapping (SM)	Huge	Less	Less	Less	More	Less
Block Mapping (BM)	Small	More than SM	Same as SM	More	More	More
Hybrid Mapping (HM)	Small	Equal to BM	More than BM	Less than BM	More	More
Log Block Based Hybrid Mapping (LBHM)	Less than SM but more than HM	Same as HM	Same as HM	Less than HM and BM	Less	Less

References

[1] Amir Ban. (1995). Flash File System. *United States Patent*, *Patent Num* (Apr. 4). Aritome, S. (2013). Flash Innovations, (November), 21–29.

[2] Ban, A. (1999). Flash File System Optimized for Page-mode Flash Technologies. *United States Patent*, *No. 5, 937*.

[3] Cho, H.C.H., Shin, D.S.D., & Eom, Y.I.E.Y.I. (2009). KAST: K-associative sector translation for NAND flash memory in real-time systems. *2009 Design, Automation & Test in Europe Conference & Exhibition*, 507–512. https://doi.org/10.1109/DATE.2009.5090717.

[4] Chung, C.C., & Hsu, H. H. (2014). Partial parity cache and data cache management method to improve the performance of an SSD-based RAID. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *22*(7), 1470–1480. https://doi.org/10.1109/TVLSI.2013.2275737.

[5] Hu, J., Xue, C. J., Tseng, W. C., Zhuge, Q., & Sha, E. H. M. (2010). Minimizing write activities to non-volatile memory via scheduling and recomputation. *Proceedings of the 2010 IEEE 8th Symposium on Application Specific Processors, SASP'10*, 101–106. https://doi.org/10.1109/SASP.2010.5521139.

[6] Hu, X.-Y., Eleftheriou, E., Haas, R., Iliadis, I., & Pletka, R. (2009). Write amplification analysis in flash-based solid state drives. *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference on - SYSTOR '09*, 1. https://doi.org/10.1145/1534530.1534544.

[7] Jung, D., Kang, J.-U., Jo, H., Kim, J.-S., & Lee, J. (2010). Superblock FTL. *ACM Transactions on Embedded Computing Systems*, *9*(4), 1–41. https://doi.org/10.1145/1721695.1721706.

[8] Kang, S., Park, S., Jung, H., Shim, H., & Cha, J. (2009). Performance trade -offs in using NVRAM write buffer for flash memory-based storage devices. *IEEE Transactions on Computers*, *58*(6), 744–758. https://doi.org/10.1109/TC.2008.224.

[9] Kim, B. (2002). US6381176B1.pdf. *United States Patent*, *US 6381176*(Apr. 30,2002).

[10] Kim, J., Kim, J. M., Noh, S. H., Min, S. L., & Cho, Y. (2002). A space -efficient flash translation layer for compactflash systems. *IEEE Transactions on Consumer Electronics*, *48*(2), 366–375. https://doi.org/10.1109/TCE.2002.1010143.

[11] Kuo, T.-W. K. T.-W., Chang, Y.-H. C. Y.-H., Huang, P.-C. H. P.-C., & Chang, C.-W. C. C.-W. (2008). Special Issues in Flash. *2008 IEEEACM International Conference on ComputerAided Design*, 821– 826. https://doi.org/10.1109/ICCAD.2008.4694174.

[12] Lee, S.-W., Park, D.-J., Chung, T.-S., Lee, D.-H., Park, S., & Song, H.-J. (2007). A Log Buffer-Based Flash Translation Layer Using Fully-Associative Sector Translation. *ACM Transactions on Embedded Computing Systems*, *6*(3), 18–es. https://doi.org/10.1145/1275986.1275990.

[13] Lee, S., Choi, W., & Park, D. (2006). FAST: An efficient flash translation layer for flash memory. *Emerging Directions in Embedded and …*, 879–887. https://doi.org/10.1007/11807964.

[14] Lee, S., Shin, D., Kim, Y.-J., & Kim, J. (2008). LAST: locality-aware sector translation for NAND flash memory-based storage systems. *ACM SIGOPS Operating Systems Review*, 36–42. https://doi.org/10.1145/1453775.1453783.

[15] Li, H.-L., Yang, C.- L., & Tseng, H.-W. (2008). Energy-aware flash memory management in virtual memory system. *IEEE Transactions on Very Large Scale*

Integration (VLSI) Systems, *16*(8), 952–964. https://doi.org/10.1109/TVLSI.2008.2000517.

[16] Lin, P. K., Chiao, M. L., & Chang, D. W. (2010). Improving flash translation layer performance by supporting large superblocks. *IEEE Transactions on Consumer Electronics*, *56*(2), 642–650. https://doi.org/10.1109/TCE.2010.5505982.

[17] Mittal, S., & Vetter, J. S. (2015). A Survey of Software Techniques for Using Non-Volatile Memories for Storage and Main Memory Systems. *IEEE Transactions on Parallel and Distributing Systems*, *9219*(c), 1–14. https://doi.org/10.1109/TPDS.2015.2442980.

[18] Park, C., Cheon, W., Kang, J., Roh, K., Cho, W., & Kim, J.-S. (2008). A reconfigurable FTL (flash translation layer) architecture for NAND flash-based applications. *ACM Transactions on Embedded Computing Systems*, *7*(4), 1–23. https://doi.org/10.1145/1376804.1376806.

[19] Qin, Z., Wang, Y., Liu, D., & Shao, Z. (2012). Real-time flash translation layer for NAND flash memory storage systems. *Real-Time Technology and Applications - Proceedings*, *2*(1), 35–44. https://doi.org/10.1109/RTAS.2012.27.

[20] Ryu, Y. (2011). A flash translation layer for NAND flash-based multimedia storage devices. *IEEE Transactions on Multimedia*, *13*(3), 563–572. https://doi.org/10.1109/TMM.2011.2114333.

[21] Takeuchi, K. (2013). Nand flash application and solution. *IEEE Solid-State Circuits Magazine*, *5*(4), 34–40. https://doi.org/10.1109/MSSC.2013.2278087.

[22] Wu, C. H., Lin, H. H., & Kuo, T. W. (2010). An adaptive flash translation layer for high-performance storage systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 29*(6), 953–965. https://doi.org/10.1109/TCAD.2010.2048362.