ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645
Indian Journal of Science and Technology, *Vol 11(23), DOI: 10.17485/ijst/2018/v11i23/125641, June 2018*

A Survey on Flash Translation Layer for NAND Flash Memory

Shailesh Kumar*, Kumkum Dubey and P. K. Singh

Computer Science and Engineering, Madan Mohan Malaviya University of Technology, Gorakhpur - 273016, Uttar Pradesh, India; shailesh23jan@gmail.com, dubeykumkum0607@gmail.com

Abstract

The requirement for storage performance and capacity are increasing rapidly. NAND flash-based SSDs have been proposed as a reliable and speedy and low power consumption storage device. An important part of each SSDs is its flash translation layers (FTL). Flash translation layer is highly impact overall performance and it manages the internal data layout for storage. There are many different trade-offs involved in FTL implementation. This survey focuses on address translation technologies and provides a broad overview of existing schemes. In flash memory, flash translation layer is a very important structure and so many techniques have been proposed.

Keywords: Address Mapping, Bast, Fast, Ram Table

1. Introduction

Flash technology has been in use for a long period. Before being used for main system storage, they have been use in embedded system as well as in small-size NVRAM. Today's multimedia storage servers provide online access to large collections of delay-sensitive data such as audio and video with various playback requirements[1-3]. In the recent advance in semiconductor technology has allowed the implementation of NAND flash-based main storage with a lower cost with a much higher density. One basic hardware characteristics of flash memory is that it has an erase- before- write architecture[4]. In order to use flash memory as a storage device, a flash translation layer (FTL) is widely used[5]. The key role of an FTL is to redirect each write request to ant the empty area of flash memory, thereby avoiding the "erase -before write" limitation of flash memory[6]. FTL supports address translation as well as provides other useful component like garbage collector and wear-leveler, maintain the same level of wear for each block in NAND flash and optimize the space utilization. Therefore in NAND flash management, FTL plays an important role.

2. FTL Functionalities

An FTL should give the following functionalities:

- Logical to physical address mapping: most important functionality of an FTL is to change logical address from the file system to physical address in flash memory.
- Power off recovery: during FTL operation, when a sudden power-off event occurs then FTL data structures should be saved and data consistency should be maintained.

**Author for correspondence*

- Wear-leveling: flash translation layer should include a wear leveling function to wear down memory blocks.

2.1 Architectures

There are two methods to implement FTL in the system. In embedded system, FTL is generally implemented in the system shown in Figure 1.

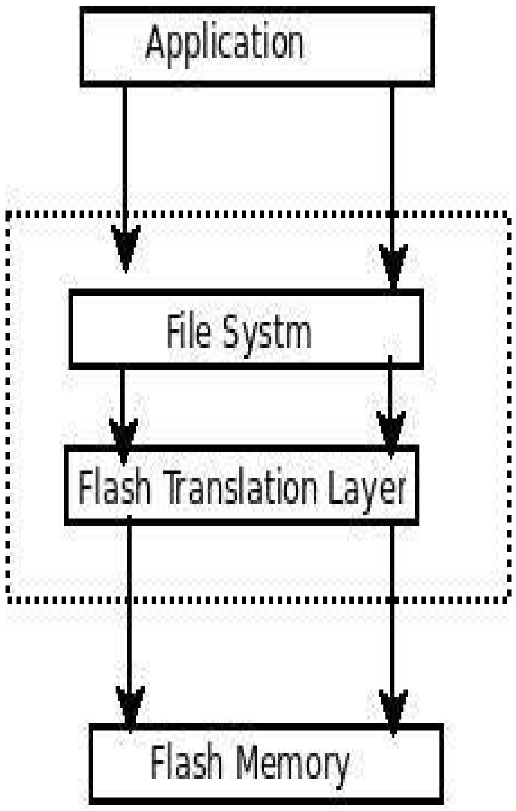


Figure 1.

2.2 Data Structure

There are two important data structures to implement address translation: a direct map and an inverse map [Gal and Toledo 2005]. In the direct map logical address is map to physical address and this map is fundamental data structure of an FTL. The address procedure can be as easy as an array lookup, although it may also include searching a tree.

2.3 Matrices

Before further introduction, we provide some important matrices that are helpful to understand the pros and cons of different FTL design. Some of them depend on the mapping granularity whereas other rest with the selection of data structures and algorithms.

2.4 Distributions

Most early FTL implemented for NOR-type flash only, which is widely to replace older on-board chips and also makes the basis of early flash-based removable media, such as Compact-flash [Wikipedia 2012a]. Therefore, some of early FTLs may not work with NAND flash because NAND flash is not byte addressable. In the early 2000s, NAND started to lead the market [Samsung 2003], and recent FTLs are mostly implemented for NAND flash memory.

3. Taxonomy for FTL Algorithm

In this section, taxonomy for FTL algorithm is proposed according to features that contain address mapping, mapping information management, and the size of Ram table.

3.1 Address Mapping

3.1.1 Sector Mapping Scheme

In this scheme, every logical sector is mapped to corresponding physical sector. Therefore if there is no logical sector in the file system the row size of the logical to physical mapping table is n. In the below Figure (2), it is consider that a block is composed of four pages, where each page consist of data sector and spare area. In another case, the FTL determine the address of an empty physical sector, writes data to it, and maintain the mapping table. If an empty sector does not exist, the FTL will choose a victim block from flash memory, copy the valid data in the victim block and update the mapping table. After that, it will remove the victim block, which will become the spare block.

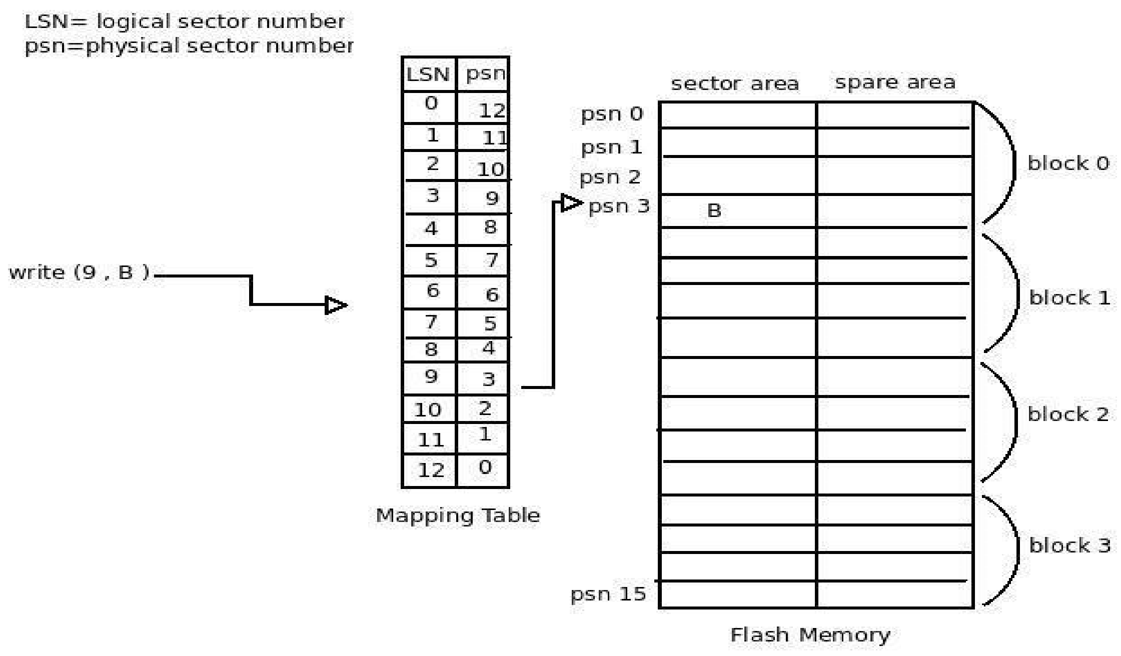


Figure 2. Sector Mapping.

3.1.2 Block Mapping Scheme

The sector mapping needs a huge amount of memory space and it is feasible for tiny embedded system. To remove this problem, block mapping schemes are proposed. In the block mapping, the logical sector offset is mapped to physical sector offset. The block mapping

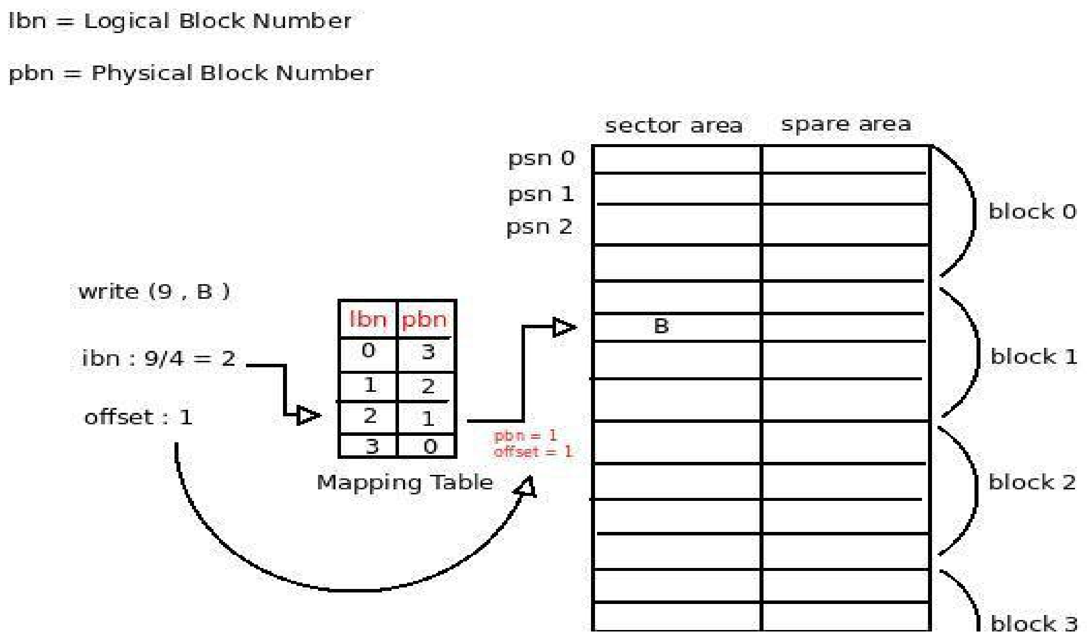


Figure 3. Block Mapping.

requires a less amount of mapping information when compared to sector mapping.

If SSD drive have 236 pages per block. That's mean block-level mapping requires 236 times less memory than page-level mapping, which is great improvement for space utilization. The mapping still needs to be persisted on disk in case of power failure and workloads with a lot of small updates and full blocks of flash memory will be written whereas pages would have been enough. Since this increases the write amplification and makes block-level mapping widely inefficient[12].

3.1.3 Hybrid Mapping Scheme

In the above two subsection, both sector and block mapping have some disadvantages, therefore hybrid mapping schemes were proposed. The trade-off between page-level mapping and block-level mapping is the one of performance versus space. Some researcher have tried to get the best of both worlds, giving birth to the so called "hybrid" approaches[11]. In this approach, obtain the physical block via a block mapping approach. After that, locate an available empty sector within the physical block via sector mapping approach. Hybrid-level FTL has been widely used for large scale flash storage system[13]. Hybrid policy has huge improvement on the performance of FTL[14].

3.1.4 Log Block Based Approach

Kawaguchi et al. First present using a log based structured FTL to give a block interface to flash with a cleaner same as LFS's[16]. The major objective of this approach is to efficiently manage access patterns efficiency. To obtain this purpose, the log block approach manage most of the physical blocks at the block addressing level - data block and a compact fixed number of physical blocks at the sector addressing level-log blocks.

3.1.4.1 BAST

BAST[10] gives the permission to each data block to have at most one dynamically allocated log block containing overwrites of that data block. If allocated log block cannot contain the current write, it is merge with its data block. This FTL have log block thrashing problem such as fre-

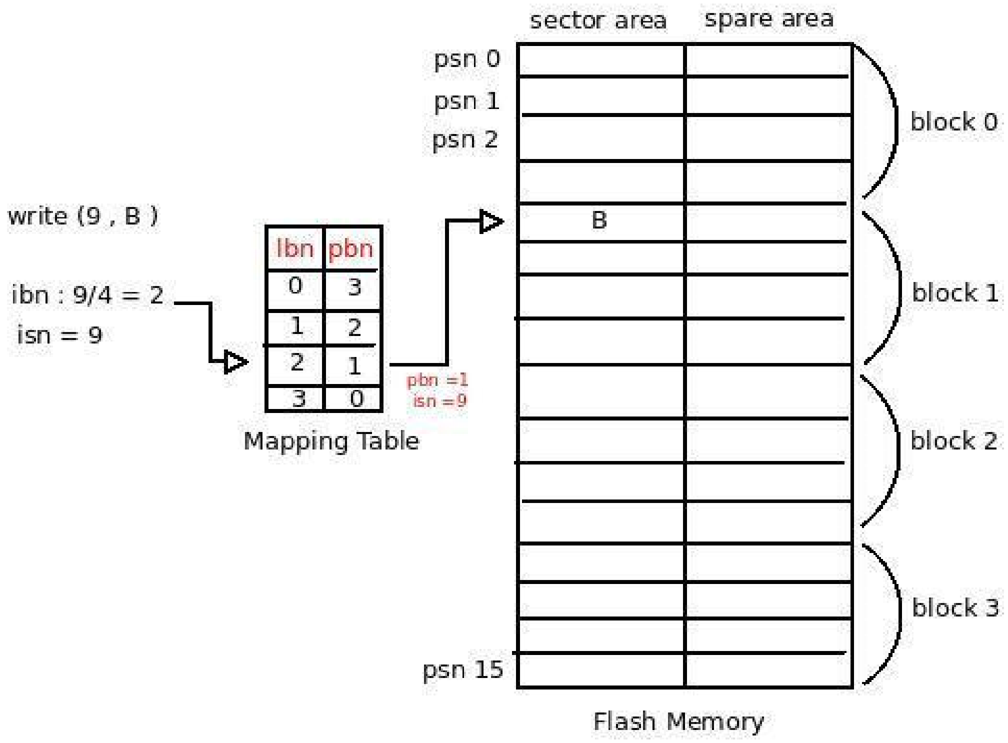


Figure 4. Hybrid Mapping.

quent deletion of log block with low utilization and one of them is the high miss ratio in direct mapped associativity. *Kim et al.* Present the BAST FTL scheme for small dense flash devices. Using an overflow strategy that chains log block off the primary data storage same as the range storage overflow containers[10].

3.1.4.2 FAST

FAST is the part of the FTLs in several studies. Hybrid-mapped FTL is called as the Fully Associative Sector Translation (FAST) FTL and it gives good performance in random writes. FAST FTL divides the whole flash memory into a large area containing data area and log area. Page overwrites are reconnected to the log area and that contain sequential write log block and random write log block. Sequential write log block (SW) and random write log block (RW) are reserved for sequential and random overwrites respectively. The SW log block corresponds to a single data block whereas RW log block correspond to multiple data block. If sequential overwrite cannot be fulfilled by current SW log block then the SW log block is combine with its corresponding data block. On other hand if a random overwrites cannot be fulfilled by the RW log block then FAST choose a victim RW log block by using round robin technique and combine the victim with its corresponding data block.

3.2 Maintaining Address Mapping Information

When implementing an FTL algorithm, it is compulsory to consider an approach to store mapping information. To be able to restore the mapping table during a power-on process, mapping should not be miss during sudden power-off event. Therefore this information should persevering kept somewhere in flash memory. The scheme for storing mapping detail in flash memory can be organized into two classes: the map block method and per block method. A map block method reserve mapping information into some dedicated blocks of flash memory (map block). Map block can store all mapping detail in the case of block mapping. Deletion on map block occurs very regularly when one map block is used. Therefore some map blocks are used to make such regular deletion. In Per Block Method, Mapping information can be maintained to each physical block of flash memory. In this method hybrid mapping is being used.

3.3 Size of RAM Table

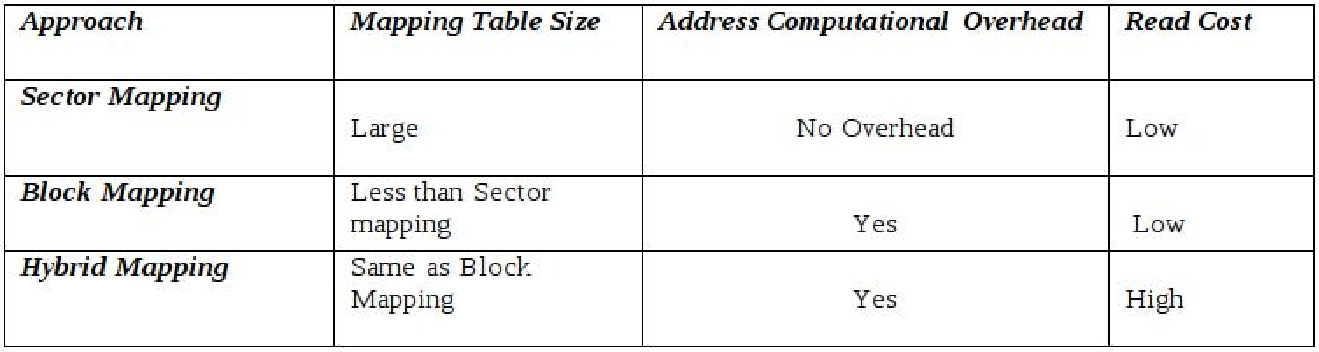
To design FTL algorithm, The size of RAM is very important factor because overall system cost are depend on it. If the systems have lower cost than size of RAM is small. Although if system has enough RAM, the performance can be raised FTL algorithm can be categorized according to their RAM construction. The following information of FTL algorithms are store in RAM.

- **Logical to physical mapping information:** The main usage of RAM is to store the logical to physical mapping information.
- **Free memory space information:** After free memory space information in flash memory is stored in RAM, FTL algorithm can maintain the memory space without again flash memory accesses.
- **Information for wear-leveling:** RAM stored wear leveling information such as deletion count of flash memory blocks may be stored in RAM.

4. Importance of FTL Mapping Schemes in Flash Memory

FTL schemes use an out-of place update procedure to avoid the erase-before-write limitation of NAND flash memory. When flash translation layer receives write/read request along with logical sector address to physical sector address form the file system, then it maps the logical sector address to physical sector address in the flash memory. But if there is any update request arrives in the FTL the previous page is invalidated and requested data is written to an available free page. Therefore, updated request can be adopted without any block erase mechanism. However flash translation layer is essential to conduct garbage collection, which delete the block that hold invalid pages in order to make them present to use. Yoo et al.[15] present a petric-net based parametric framework that can imple-

Table 1. Comparison of FTL algorithms



Approach	*Mapping Table Size*	*Address Computational Overhead*	*Read Cost*
Sector Mapping	Large	No Overhead	Low
Block Mapping	Less than Sector mapping	Yes	Low
Hybrid Mapping	Same as Block Mapping	Yes	High

ment the correct sequence of FTL scheme to be executed for employ an incoming FTL request. This derived sequence is transparent to FTL operation and it can be perform to many FTL operation to calculate the WCET of the request at runtime. FTL deigns can be mainly categorized into three types[7], page-level mapping[8], block-level mapping[9] and hybrid-level mapping[10]. In the page-level FTL maps a logical page number into corresponding physical page number in NAND flash. This mapping policy offers good address translation time, less garbage collection overhead and high space utilization but having some major drawback is that mapping table requires a lot of RAM, which can significantly increase the manufacturing costs. A solution to that would be to map blocks instead of pages, block-level mapping.

5. Comparison

FTL's performance levels are compared in terms of read/write performance and mapping information. The read/write performances are calculated by the number of flash operation such as read, write and erase operation. The mapping table is managing in RAM and that access cost of the mapping table is zero.

Another way of comparison is the memory requirement for saving mapping information. Mapping detail should be store in persistent storage, and it can be remove in Ram for best performance. Some FTL use integration of sector, block and hybrid mapping. Hybrid mapping approach has higher read cost as compared to the sector and the block mapping approach, where as in sector mapping the read operation can be determined directly from the mapping table. In sector mapping, the probability of requiring deletion operation per write is relatively low. Write operation have three cases. First, the write operation may be carry out in the in-place location directly. Second, the write operation should be carry out after scanning the empty position in a block. Third the write operation may sustain an erase operation.

6. References

1. Yu H, Zheng D, Zhao BY and Zheng W. Understanding user behavior in large-scale video-on-demand systems. ACM SIGOPS Operating Systems Review. 2006; 40(4):333-44. Crossref.
2. Cha M, Kwak H, Rodriguez P, Ahn Y-Y and Moon S. I tube, youtube, everybody tubes: Analyzing the world's largest user generated content video system. Proceedings of the 7th ACM SIGCOMM conference on Internet measurement. 2007; p. 1-14.
3. Huang C, Li J and Ross KW. Can internet video-on-demand be profitable? Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications. 2007; p. 133-44. Crossref.
4. Samsung Electronics. Nand Flash Memory & Smartmedia Data Book. 2007.
5. Ban A. Flash File System. U.S. Patent 5 404 485. 1995; Apr 4.
6. Lee S, Moon B, Park C, Kim J and Kim S. A case for flash memory SSD in enterprise database applications. ACM SIGMOD. 2008; p. 1075-86. Crossref.
7. Chung T-S, Park D-J, Park S, Lee D-H, Lee S-W and Song H-J. A survey of flash translation layer. Journal of Systems Architecture. 2009; 55(5-6):332-43. Crossref.

8. Ban A. Flash file system. US patent 5,404,485, 1995; April 4.
9. Ban A. Flash file system optimized for page-mode flash technologies. US patent 5,937,425, 1999; August 10.
10. Kim J, Kim JM, Noh S, Min SL and Cho Y. A space-efficient flash translation layer for Compact Flash systems. IEEE Transactions on Consumer Electronics. 2002 May; 48(2):366-75. Crossref.
11. Park et al. A Reconfigurable FTL (Flash Translation Layer) Architecture for NAND Flash-Based Applications. 2008.
12. Kim et al. Parameter-Aware I/O Management for Solid State Disks (SSDs). 2012.
13. Chang L-P and Kuo T-W. An efficient management scheme for large-scale flash-memory storage systems. Proceedings of the 2004 ACM symposium on Applied computing (SAC '04). 2004; p. 862-8. Crossref. PMid:15530111.
14. Memory Technology Device (MTD) Subsystem for Linux. 2010. Available from: http://www.linux-mtd.infradead.org/.
15. Yoo J, Lee J and Hong S. Petri net-based FTL architecture for parametric WCET estimation via FTL operation sequence derivation. IEEE Transactions on Computers. 2013 Nov; 62(11):2238-51. Crossref.
16. Rosenblum M and Ousterhout JK. The design and implementation of a log-structured file system. ACM Transactions on Computer Systems. 1992; 10(1):26-52. Crossref.
17. Lim SP, Lee SW and Moon B. Faster FTL for enterprise-class flash memory SSDs. Proceedings of SNAPI. 2010; p. 3-1. Crossref.