Apacer
Access the best

Flash Translation Layer (FTL)

White Paper

September 21, 2015

Version 1.0

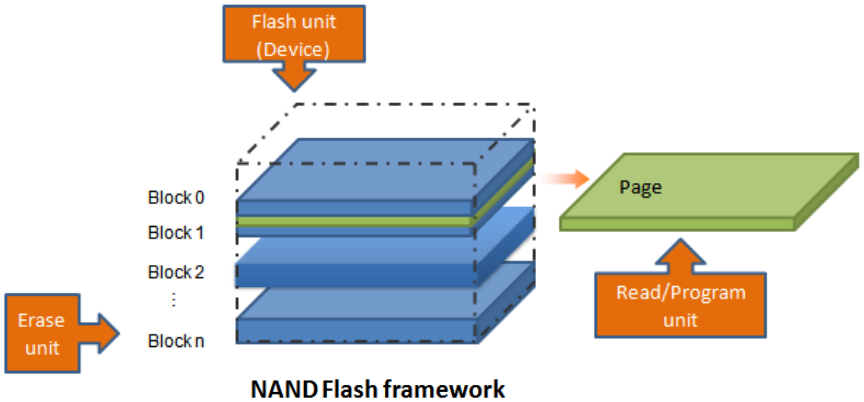
Apacer
Access the best

Apacer Technology Inc.

1F, No.32, Zhongcheng Rd., Tucheng Dist., New Taipei City, Taiwan, R.O.C

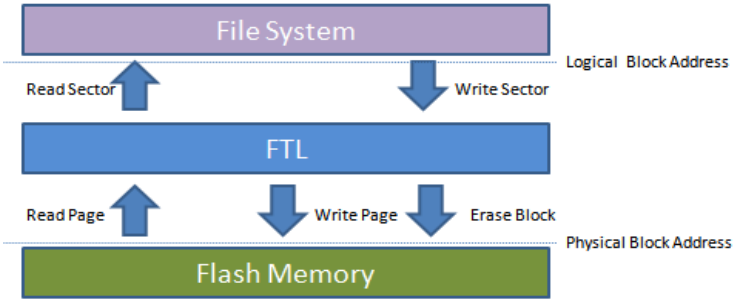
NAND flash

NAND flash memory, a non-volatile storage medium, is able to be electrically read, erased, and programmed. However, a block must be erased before new data is written on one of its already used pages as overwrites are not allowed, called erase-before-write. A flash, known as device, consists of a number of blocks, and each block contains a group of pages. For NAND flash operations, the basic units of read and program operations are on per-page basis, while the basic unit of erase operation is on per-block basis.



NAND Flash framework

Flash translation layer

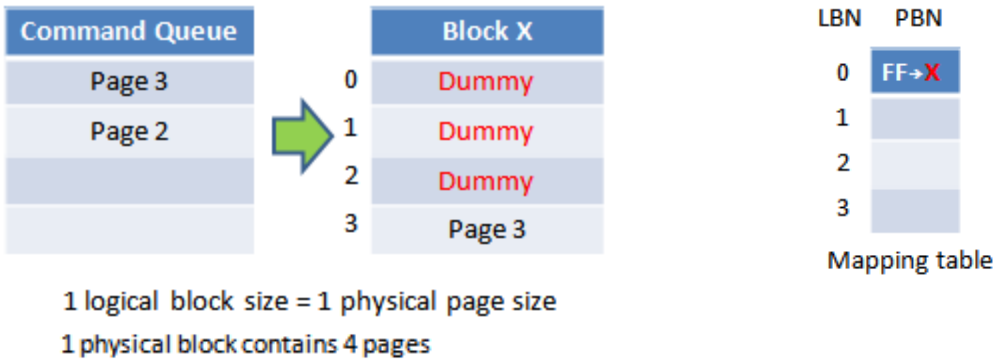


Flash Translation Layer (FTL) acts like a translator to perform communication between the sector-based file system and NAND flash chips. It is a software-based mechanism to support normal file systems with NAND flash memory. In order to increase its efficiency, it hides the complexity of flash and emulates flash as a hard disk drive by providing a logical block interface to the flash device. A FTL maps logical blocks to physical flash pages and erase physical blocks. It functions as a database query system allowing one to send an inquiry and retrieve the data from the database. Based on the unit of mapping, commonly used address mapping types are block-level and page-level mapping.

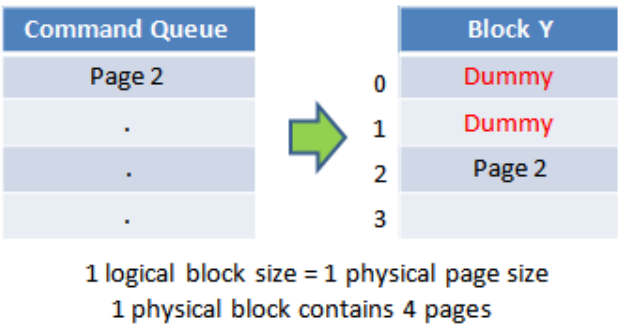
Block-level mapping

Block-level mapping translates logical block addresses (LBAs) from the host into physical block addresses (PBAs) in the flash memory. It can only map a logical page to a fixed offset of a block, which means page N within a logical block refers to page N within its corresponding physical block. The below example shows how block-level mapping performs a write command:

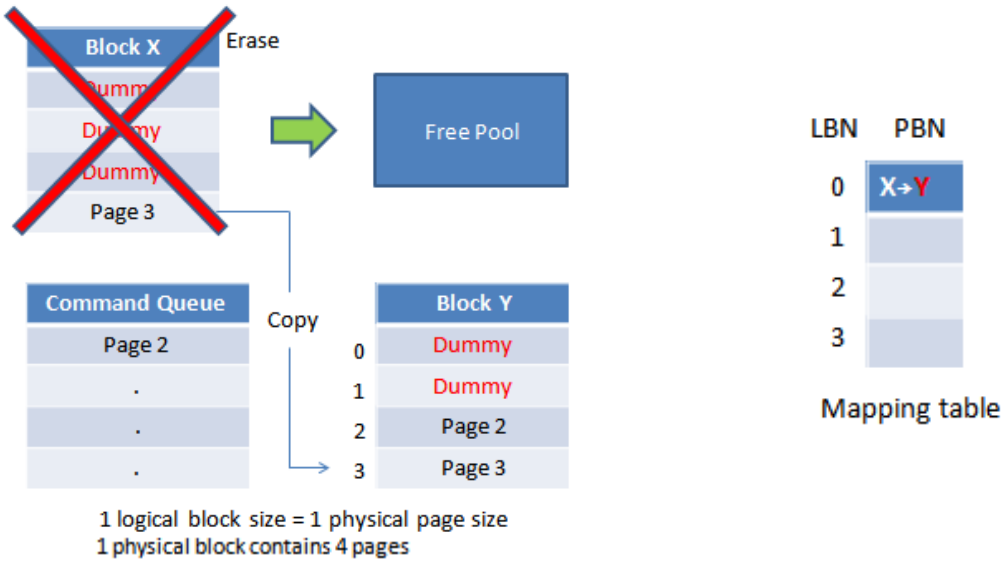
1. Host instructs two write commands: page 3 and 2. Since no data has been written to page 0, 1, and 2 of the destination block X and pages within a block must be written sequentially. The firmware instructs “dummy write” command to pretend these pages have been written in order to ensure the data integrity. If no data is filled in page 0, 1, and 2 or they are left blank, the block will become unable to read. Page 3 then is written to the block X. After the write command is completed, the mapping table updates itself automatically.



2. Since block X has been written and cannot be overwritten, the next write command “page 2” is written to block Y. Page 0 and 1 of destination block Y do not contain any data and the two pages are filled “dummy” data as well by the firmware. Page 2 is then written into block Y.



3. Then, page 3, the old data from block X, is copied to block Y and block X is marked as erased and return to the free pool. Again, after the entire process is completed, the mapping table updates itself automatically



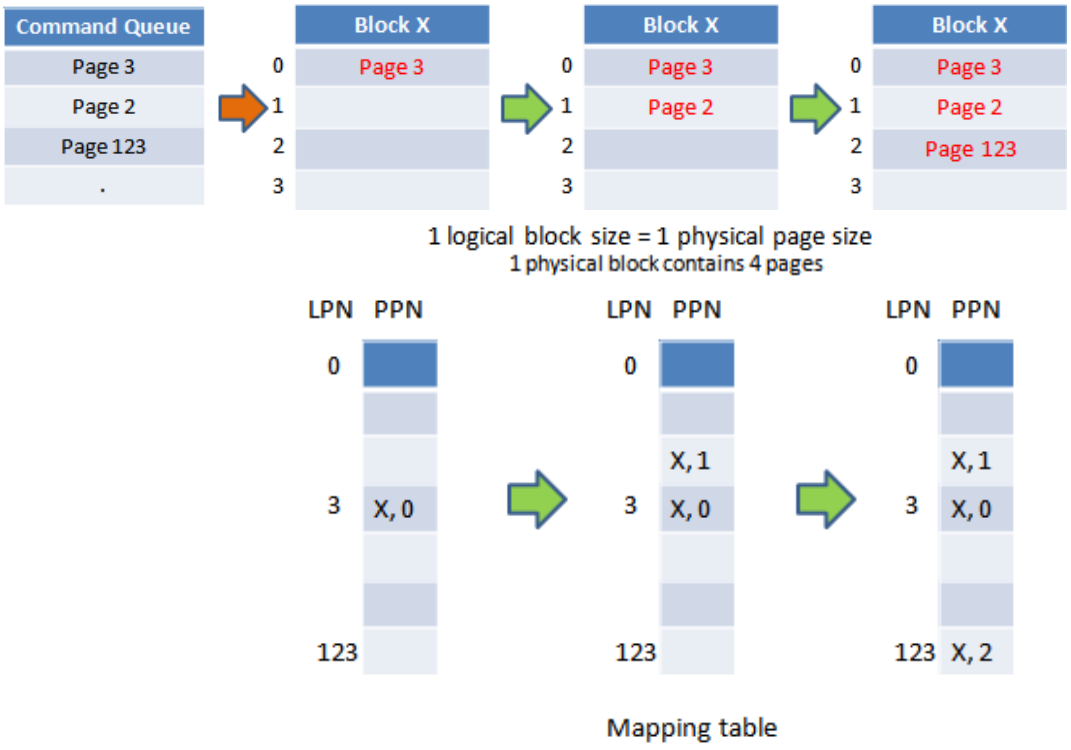
Note: the example only shows the concept of how block-level mapping work and do not necessary happen in an actual case

Block-level mapping is an ideal method to perform sequential data, and its mapping table does not require too much memory since the data is stored on block basis. However, when the data is not written sequentially, the mapping behavior will take extra operations and affects its performance (one erase and a number of read/write) as shown in the example above, while it is unnecessary for page level mapping.

Page-level mapping

Page-level mapping uses one page as the unit of mapping. The most important characteristic of page-level mapping is that each logical page can be mapped to any physical page on the flash memory device. This mapping algorithm allows different size of data to be written to a block as if the data is written to a data pool and it does not need to take extra operations to process a write command. The below example shows how page-level mapping performs a write command:

Host instructs three write commands: page 3, 2, and 123. The three pages are written into block X in sequence of command queue. Once all write commands are completed, the mapping table updates itself automatically.



Note: the example only shows the concept of how page-level mapping work and do not necessary happen in an actual case

This fine-grained page-level mapping scheme makes better capability for handling random data, and increases overall performance and endurance significantly. However, page-level mapping requires SSDs to incorporate a larger RAM in order to maintain its mapping table.

Various effects on mapping structures

FTL Structure	Page Mapping	Block Mapping
Memory Requirement	Large	Small
Random Write Performance	High	Low

In summary, based on these above findings, block-level mapping offers excellent sequential write and only requires less memory resource to handle the mapping table. But, the algorithm takes extra operations result in affecting its performance and endurance. On the other hand, page-level mapping provides better reliability, endurance, and random write performance. However, it requires more memory resource in order to manage and maintain its mapping table.

Revision History

Revision	Date	Description	Remark
0.1	09/01/2015	Preliminary	
1.0	09/21/2015	Official release	

Apacer Technology Inc.

1F., No.32., Zhongcheng Rd.,
Tucheng Dist., New Taipei City 236, Taiwan R.O.C
Tel: +886-2-2267-8000 Fax: +886-2-2267-2261
www.apacer.com