Flash Translation Layer Using Multi-Thread

Jaehyun Kim¹, Yuli Aria Winata¹ and Ilhoon Shin*¹

¹ Department of Electronic Engineering, Seoul National University of Science &
   Technology
   * Ilhoon.shin@seoultech.ac.kr

Abstract. NAND flash memory, which has shock-resistant, less noise, and high
I/O operations per second (IOPS) compared with Hard Disk Drive (HDD), has
been widely used in various fields such as mobile, PC, server system. Despite
of its advantages, NAND flash memory has erase-before-write characteristic
which needs to be handled by Flash translation layer (FTL). Thus, NAND flash
memory performance depends on the FTL design itself. Existing FTL studies
mostly focused on improving FTL algorithm and further improve NAND flash
memory performance. However, there are not enough studies that exploits
multi-core processors which become popular nowadays. In this paper, we
proposed a new multi-thread FTL which uses the advantage of multi-core
processors.

Keywords: NAND flash memory, Flash translation layer, Multi-thread

1 Introduction

NAND flash memory with high I/O operations per second (IOPS) is widely used as a
storage device, but it has a limitation which is its erase-before-write characteristics.
Moreover, unlike Hard Disk Drives (HDDs) which uses sector as its operation unit,
NAND flash memory read/write operation is in unit of a page and erase operation is
in unit of a block. In order to overcome this problem, NAND flash memory must
employs Flash translation layer (FTL). One of the FTL features is out-of-place-update
which solves erase-before-write problems. When a page is updated, instead of
overwriting the page, FTL writes to another empty page and manages physical page
and logical page location in a mapping table. If NAND flash memory continuously
uses out-of-place-update, SSD will lack of empty pages and several pages will be
dirty. To claim the dirty pages and change it to empty pages, FTL triggers garbage
collection process. It selects victim block which is generally a block that has the most
invalid pages. If there are valid pages in the victim block, it first will be copied into
another free block. After all valid pages copied, victim block then is erased and
becomes a new free block. Since garbage collection contains copies and erases
operations, garbage collection greatly affects NAND flash memory performance
compared to read and write operation.

* Corresponding Author

ISSN: 2287-1233 ASTL
Copyright © 2015 SERSC
Several papers primarily focused on reducing garbage collection such as hot/cold scheme [1]. Hot/cold scheme divides data into frequently updated (hot) data and rarely updated (cold) data. Grouping hot/cold data can lower the valid page rate in the victim block, thus improve performance and life time. The other way to improve NAND flash memory lifetime is using wear leveling and balancing workloads across NAND flash memories inside an SSD. This is possible because SSD has multiple independent processing units. A study about SSD internal configuration has been done [2]. However, they never consider multi-core processors inside SSD architecture.

Knowing this fact, we try to study about multi-thread FTL. One major problem in implementing multi-thread FTL is how to protect shared resources such as mapping table and metadata. If FTL does not properly protect the shared resources, race condition and deadlock could occur. When multiple processes access one resource at the same time without proper sequence order, race condition could happen. To prevent race condition, system usually implements lock system to protect a resource. However, if a lock is accessed in circular manner by multiple processes, those processes could be stopped and leads to a system deadlock.

Knowing the importance of shared resources, we proposed a multi-thread FTL in a multi-core processor system while still considers and protects shared resources. To implement our idea, we used a semaphore to ensure each thread can use shared resource exclusively when needed. We designed a multi-thread FTL based on page mapping scheme [3] and optimized it for multi-core processors using multi-thread.

2 Background

2.1 Page mapping

![Page mapping scheme](image)

Fig. 1. Page mapping scheme

To overcome erase-before-write problem in NAND flash memory, FTL uses out-of-place-update instead of overwrites. FTL records logical page number (LPN) and physical page number (PPN) relationship in every out-of-place update process inside a mapping table as illustrated in Fig.1. Mapping table management scheme can be divided into three categories based on its management unit; page mapping [3], block mapping [4] and hybrid mapping [5-6]. Unlike the other mapping scheme, page mapping manages the mapping table in unit of a page, which is the same as read/write units. Thus, this scheme has the best performance compared to other schemes.
However, since it has to maps all pages inside an SSD, this scheme has the highest memory usage [3] compare with others.

### 2.2 Multi-thread

Multi-thread programming is a method to fully optimized multi-core system. It shares code segments, heap segments and data segments, but not stack segments. Using multi-thread can improve performance because processes can be parallelized. When system designer decided to use multi-thread, they faced several major problems. The first one is context-switch overhead. Since multi-thread needs to change stack when context-switch occurs, frequent context-switch could cause system performance overhead. The other problems appear when shared resources are not properly protected such as race condition and deadlock. Lastly, for programmer, debugging a multi-thread system is hard and leads to a longer development time [7].

### 3 System design

This paper proposed a novel FTL scheme optimized for multi-core system. We inherit the page mapping code used inside openSSD firmware [8]. OpenSSD uses real hardware thus it is more reliable than other software simulators [9-10]. However, since this board does not support multi-thread, we make a new software simulator based on openSSD firmware then implement multi-thread FTL on this simulator. After studied OpenSSD firmware, we decide that every access related with mapping table, metadata and NAND cell is a critical region and need to be protected by a semaphore. We only use one semaphore in this research for design simplicity.

Fig. 2 shows the main function in FTL page mapping code which are ftl_open(), ftl_read(), and ftl_write(). ftl_open() is in charge of initiating mapping table and metadata. ftl_read() is responsible for read operation in NAND flash memory. Since this function simply gets PPN from mapping table and read the NAND flash cell, this function is easy to understand. On the other hand, ftl_write() is in charge of write operations and also triggering garbage collection. Thus it has the most complicated code. First, it calculates page location using sector number and sector count. Using these two information, it invokes

![Flash translation layer code structure](image-url)
write_page() function to perform real write operation in NAND flash cell. 
write_page() function also call assign_new_write vpn() to get new free 
page, and garbage_collection() to trigger garbage collection process. 

Note that all functions we mentioned are related with shared resources. Thus we 
protect these functions using one semaphore to prevent race condition and deadlock. 
However, when only using one semaphore, the performance is similar with single 
thread FTL. This is because when two process need to access two different shared 
resources, it needs to wait each other. To solve this problem, we are planning to 
implement multiple semaphores in our design so that SSD performance can be 
improved.

4 Conclusion

In this paper, we proposed a novel multi-thread FTL optimized for multi-core system. 
To implements multi-thread FTL, critical regions need to be properly protected. If 
critical regions are not properly protected, race condition and deadlock might occur. 
To solve this problem we use one semaphore to manage three resources which are 
metadata, mapping table and NAND flash cell access. Because we only use one 
semaphore, multi-thread FTL has similar performance with single thread performance. 
For future work, we are planning to use multiple semaphores to protect critical 
regions and further improve SSD performance.

References

on Consumer Electronics. 57: pp. 1728-1731 (2011)
No. 5,937,425 (1999)
Emerging Directions in Embedded and Ubiquitous Computing. Springer Berlin Heidelberg. 
879-887 (2006)
memory in real-time systems. Proceedings of the Conference on Design, Automation and 
Test in Europe. European Design and Automation Association (2009)
Inc. (1981)
8. The OpenSSD Project, http://www.openssd-project.org/wiki/The_OpenSSD_Project
based solid-state drives.Advances in System Simulation. SIMUL'09. First International 
Conference on. IEEE (2009)