Optimizing L2P Mapping of NAND Flash Storage based on Hybrid Memory Architecture

Sung Kyun Lee[†*], Dong Hyun Kang[†], Young Ik Eom[†],
[†]*Sungkyunkwan University, South Korea*
[*]*Samsung Electronics, South Korea*
{sk07, kkangsu, yieom}@skku.edu

1.Introduction

In NAND flash storage, *logical to physical (L2P) mapping* is one of the fundamental storage functionalities since it is responsible for mapping logical block addresses to physical flash pages to hide out-place update characteristic of the flash memory. L2P mapping information is temporally stored in DRAM inside the flash storage for improving the performance of the storage and then it is flushed to the underlying flash memory by internal (e.g., *garbage collection*) or external events (e.g., *synchronization system calls*). Unfortunately, such flush operation (i.e., FLUSH command) generates extra write operations inside the flash storage because mapping information should be stored in flash memory after flushing out all the data on the internal write buffer to the underlying flash memory. For example, a synchronization system call after a single write operation would need to write two pages: one for data page and the other for mapping page corresponding to the data page. In addition, traditional L2P mapping scheme is inefficient in terms of the flash memory because 4KB mapping page should be flushed to the flash memory even though only a few mapping entries (e.g., 4 or 8 bytes) are updated.

2. Main idea

In order to resolve the synchronization overhead described in Section 1, we propose a novel L2P mapping scheme based on the hybrid memory architecture, which is composed of DRAM and Non-volatile Random Access Memory (NVM). Especially, employing NVM for L2P mapping table has a positive effect on the overall storage performance because it is 100 times faster than NAND flash memory. We follow the basic DRAM-based L2P mapping policy and just utilize NVM to persistently maintain the L2P mapping table because it has byte-addressability and non-volatility features. Our scheme is very simple and straightforward, but it significantly cuts down on the latency of the flash storage by reducing the number of flash writes for L2P mapping table. Now, we introduce our L2P mapping scheme in detail. We first split the L2P mapping area in DRAM into 32KB segments and classify them into four utilization levels: full, dense, sparse, and clean segment. The utilization level of each segment is decided on-the-fly by the number of modified entries on a single mapping page and our scheme monitors this by using dirty count of each segment. If all mapping entries on a segment are modified (high utilization), we mark it as a full segment. Otherwise, we mark the other segments as either dense or sparse segments according to the pre-fixed thresholds. If NVM has enough free space for copying L2P mapping information, all segments are copied to NVM regardless of the utilization level when a flush operation is triggered. As a result, all flash writes caused by flush operation are completely eliminated. Otherwise, we first copy sparse or dense segments from DRAM to NVM to make them persistent and then flush full segments to the underlying flash memory. After that, mapping table update caused by write operation is serviced again on DRAM because write latency of NVM is slower than that of DRAM. As a result, update of mapping segments on NVM are delayed to the next flush operation. Finally, to make free space on NVM without performance overhead, we also periodically flush mapping segments on NVM to the flash memory during idle time.

3. Evaluation

To clearly understand the performance effect of the L2P mapping in the flash storage, we implemented our scheme on the OpenSSD Jasmine board which includes ARM core, SATA interface, and 64 MB DRAM for handling L2P mapping table with page-level FTL. Our experiments were conducted using FIO micro-benchmarks on Linux kernel version 4.6.4 with EXT4 filesystem. We evaluated sequential write performance with *fsync* and *fdatasync*. Our evaluation results show that our scheme outperforms the traditional DRAM-based L2P mapping scheme by up to 3.7 times.

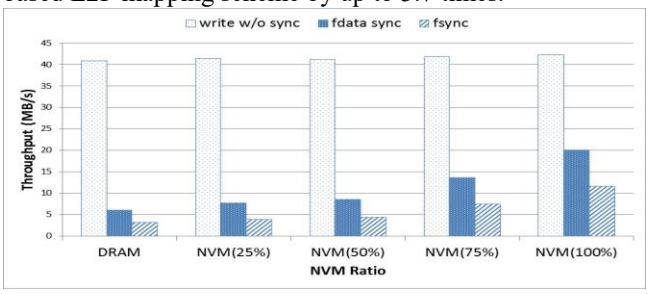

Figure1 : Performance improvements with NVM