# Understanding the Behaviors of Discard Command in both the Filesystem and the Flash-based Storage Layer *

Dong Hyun Kang and Young Ik Eom
*Sungkyunkwan University, South Korea*
{kkangsu, yieom}@skku.edu

## 1 Introduction

Today, most computing systems, such as mobile, desktop, and enterprise systems, adopt NAND flash storages (e.g., eMMC, SD Card, and SSD) as their local storage devices. Many researchers in both academic and industry communities have focused on optimizing I/O subsystems (e.g., page cache, filesystem, and block layer) on the flash storage. Especially, some researchers redesigned the filesystem to make it flash-friendly. For example, Linux Ext4 filesystem provides the `Discard` command to hide the no-overwrite characteristic of NAND flash storage from the host system.

In this paper, our goal is not to verify the performance of the `Discard` command. Instead, we try to get better understanding on the behaviors of `Discard` command in both the filesystem and the storage layer. Ext4 retains basic layout of Ext3 filesystem with some differences (e.g., block/inode allocation policy and inode size) and the `Discard` command is delivered to NAND flash storage whenever a file is deleted on the filesystem. When a user creates a 4KB file, Ext4 allocates a data block to store the file's contents and requests an inode in an inode block to record metadata of the new file, where the inode block contains 16 inodes (i.e., inode size of Ext4 is 256Byte). Ext4 allocates a new inode block before recording the file's metadata. Therefore, two pages can be allocated in the flash storage for the creation of a 4KB file. However, when the 4KB file is deleted by a user, Ext4 issues only one `Discard` command to the NAND flash storage to invalidate the data block of the deleted file. The reason why Ext4 does not issue a `Discard` command for the inode block is that the inode block of Ext4 is composed of 16 inodes and some inodes belonging to the inode block can have valid information of their corresponding files. However, current Ext4 filesystem never issues the `Discard` command to flash storage despite all inodes belonging to an inode block are invalid. As a result, the page, which is composed of 16 *invalid inodes*, continually remains in the flash storage as a valid page and it may negatively impacts both the performance and endurance of the flash storage because the valid page in the flash storage should be copied to another page during the garbage collection (GC) process of the flash transaction layer (FTL).

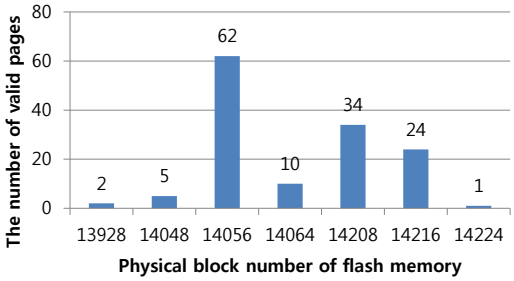


Figure 1: The number of valid pages in a flash storage

## 2 Evaluation

To clearly understand the effectiveness of the `Discard` command in the flash-based storage layer, we implemented the `Discard` command on the SSD extension of DiskSim simulator. We first turned off the journaling of Ext4 to avoid interference from the journaling and collected I/O traces by running FIO Benchmark as follows: (1) creates 1000 files at a granularity of 4KB randomly; (2) deletes the 1000 files one by one; (3) creates 1000 files at a granularity of 4KB sequentially; (4) deletes the 1000 files one by one again. Since current Ext4 remains flash pages for inode blocks of the filesystem in the flash storage of Disksim, we measured the number of valid pages in the flash storage by using the collected traces as the input of the Disksim. Unsurprisingly, Figure 1 shows that Ext4 remains a total of 138 valid pages in the flash storage of Disksim even though all files are deleted on the filesystem. Of course, a few pages are used to maintain the metadata of the filesystem. However, other remained pages are wasted pages, caused by the allocation of inode blocks. This observation clearly confirms that flash pages for inode blocks of filesystem negatively affect the performance and endurance of NAND flash storage. For our future work, we have a plan to redesign the Ext4, considering the `Discard` command for inode blocks of the filesystem.

*This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2015M3C4A7065696)