Two-Level Logging with Non-Volatile Byte-Addressable Memory in Log-Structured File Systems

Yeonseong Hwang Hyunho Gwak Dongkun Shin
College of Information and Communication Engineering, Sungkyunkwan University, Suwon, Korea
{sami0708, gusghrhkr, dongkun}@skku.edu

ABSTRACT

The file system durability is provided by flushing dirty pages periodically into the non-volatile storage. Since the traditional storage devices such as hard disk and flash memory can be written in the unit of block, the file system writes a whole block even when only a small number of bytes are modified. To resolve such a wasting write traffic problem, we propose a two-level logging scheme by exploiting non-volatile and byte-addressable memories (NVMs). Whereas the previous approach which exploits the NVM device is targeted for EXT4 file system, our scheme uses log-structured file systems in order to guarantee the file system reliability even for sudden system crashes. While the NVM is used for fine-grained logging, the flash memory is used for coarse-grained logging. Experiments with a real NVM device show that the proposed scheme reduces the write traffic on storage by up to 78% and improves the I/O performance significantly.

Keywords

Sub-page logging, Log-structured file system, Non-volatile memory

1. INTRODUCTION

Traditional block storage devices such as hard disk and solid state disk can be read or written in the unit of block. Therefore, file systems also manage the storage space in the unit of block. Generally, the block size is 4 KB. Even though only several bytes are modified, the file system writes a 4 KB of block at the storage device. We observed how many blocks are updated partially during the execution of mobibench SQLite benchmark. About 45% of written blocks are partially updated. As a result, 32% of total written data are uselessly written at the storage.

Recently, non-volatile byte-addressable memories (NVMs) such as PRAM and STT-MRAM are emerging. These NVMs offer latencies that are comparable to DRAM but they are also non-volatile. Since NVMs support byte-level write operations, they can be solutions for the wasting block writes due to the partial block updates. While NVMs provide short latencies for small-sized I/O requests, their I/O bandwidths for large-sized or burst requests are lower compared to the traditional block devices. Therefore, in order to provide high performance and remove wasting block updates, it is better to use a hybrid storage that has both NVM and flash memory. In this paper, we propose a two-level logging technique for the hybrid storage.

CF'15, May 18-21, 2015, Ischia, Italy
ACM 978-1-4503-3358-0/15/05.
http://dx.doi.org/10.1145/2742854.2742892

The delta-journaling [1] technique supports the sub-page logging for EXT4 journals. It is targeted for a hybrid storage which has PRAM and flash memory for journal space and normal data space, respectively. Only the differences between the old and new data pages are recorded at PRAM during journal commits. The journals are called delta-journal. During the checkpointing, a new data page is built with the old data page and the delta-journal, and the new data page is overwritten at the old page. However, the delta-journaling has a file system reliability problem. If there is a sudden system crash during the overwrite operations on the old pages, the corrupted data cannot be recovered even under the journaling technique. This is because the delta-journals have only partial information.

To solve the file system reliability problem of the delta-journaling, we use the log-structured file system (LFS). Since LFS maintains the old data until it is reclaimed by garbage collection, the file system reliability can be guaranteed even though there is a system crash while the sub-page logs in NVM are written to the file system in flash memory. Our two-level logging technique maintains fine-grained logs (FG-logs) at NVM and coarse-grained logs (CG-logs) at flash memory. When there are dirty page write requests on flash memory storage device, the write requests are intercepted by the NVM device and only modified sub-pages are recorded at the NVM device. If there is no free space in the NVM device or there are needs for writing full-sized blocks at the file system, our scheme recorded the CG-logs at the flash memory.

2. TWO-LEVEL LOGGING

We use the F2FS [2] as a target file system. F2FS is a log-structured file system optimized for flash memory devices. We modified the page cache management layer of Linux in order to track sub-page modifications. As a finer-grained tracking is used, the required memory overhead increases. In addition, our target NVM device supports 128 byte of write granularity. Therefore, we determine the size of sub-page modification tracking as 128 byte, and 4 byte of bitmap is required for each page.

For the sub-page modification tracking, we add a bitmap data structure in the page cache management layer. When a page is modified, the corresponding bits in the bitmap are set. In addition, we add another bitmap to manage *logged bits*. The logged bit represents whether the corresponding page is logged at the NVM device. When any sub-pages of a page are logged at the NVM device, the logged bit is set. After the FG-logs of the page are written at the flash memory, the logged bit is cleared.

Figure 1 shows the overall architecture of the proposed two-level logging. In the step of (1), the file system loads three pages, A_0, B_0, and C_0, from the flash memory into the page cache. Their states are clean. In the step of (2), B_0 is changed to B_1 by a write operation, and the page cache has the dirty page B_1. In the step of (3), the flush thread generates a write-back request for the dirty page B_1, only the modified sub-pages are written at NVM, and the state of B_1 is changed to clean and logged. The flash memory

block at the address of 100 is allocated for B_1 by F2FS, but it remains empty. In the steps of (4)-(5), C_0 is also modified, and the NVM has the modified sub-pages of C_1. In the step of (6), C_1 is again modified into C_2, and the state of C_2 becomes logged and dirty.

(1) Read A_0, B_0, C_0 (clean);
(2) Write B_1 (dirty); (3) WB to NVM B_1 (logged/clean);
(4) Write C_1 (dirty); (5) WB to NVM C_1 (logged/clean);
(6) Write C_2 (logged/dirty);
(7) Evict A_0;
(8) WB to flash B_1 (clean); Evict B_1; Invalidate log($\triangle B_1$);
(9) WB to flash C_2 (clean); Evict C_2; Invalidate log($\triangle C_1$);

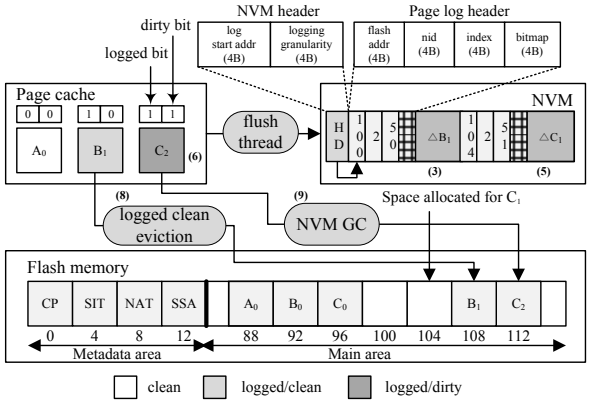

Figure 1. Overall architecture of two-level logging

When the clean page A_0 is evicted in the step of (7), there is no change in both NVM and flash memory. However, when the logged/clean page B_1 is evicted in the step of (8), the page is written at the flash memory. Since the modified sub-pages are logged at NVM, it is not required to write B_1 at flash memory. However, if there is a read request on B_1 after the page eviction, the file system should make B_1 with B_0 in flash memory and ΔB_1 in NVM. The process should read both flash memory and NVM. In addition, there are overheads for searching B_0 and the related log. Since the overhead can increase the read latencies, our two-level logging scheme writes the evicted clean/logged page into the flash memory. Therefore, either the page cache or the flash memory has the latest version of data, and the read operation does not need to access the NVM device. Instead of writing the logged page at its allocated block in the flash memory (i.e., block address 100 for B_1), a new flash memory block is allocated in order to avoid random write requests. After the write operation on flash memory, the FG-logs of the page in NVM are invalidated. In the step of (9), the logged/dirty page C_2 is also evicted to the flash memory because there is no free space in the NVM device.

When a dirty page is sent to NVM, if too many sub-pages are modified, our scheme writes the page at the flash memory directly bypassing the NVM device. This is because the flash memory provides a higher performance than the NVM device for large-sized requests. However, the metadata pages of F2FS should be written at the NVM device irrespective of the size of modified sub-pages. Then, F2FS can access the old metadata in the flash memory during file system recovery process for sudden system crashes.

The NVM device has the NVM header which has the start address of valid log and the logging granularity. Each FG-log for a page has the page log header, which is composed of the allocated flash memory block address of the page, its node ID, the block index within its node, and the modified sub-page bitmap. The node is a data structure of F2FS, which has the pointers for data blocks. If a system crash occurs, the scheme makes the latest version of each logged page by scanning the FG-logs in the NVM. Then, the page is written at the flash memory.

3. EVALUATION

We evaluated the effectiveness of two-level logging with a 512 MB of PRAM-equipped embedded board [3]. It uses 4 GB of SD card as storage. Figure 2 compares the write traffics under the original F2FS and the two-level logging schemes. For the mobibench and filebench workloads, the write traffics on flash memory are reduced by 78% and 49%, respectively. The reduced flash memory traffics are changed into the PRAM write traffics. The PRAM write traffics occupy only 16% and 21% of the reduced flash memory traffics at mobibench and filebench workloads, respectively. Therefore, the proposed two-level logging scheme can improve the lifetime of flash storage. As shown in Table 1, the two-level logging scheme shows 2.2 times higher bandwidth at the mobibench workload, and shows 4.6 times shorter latency at the filebench workload.

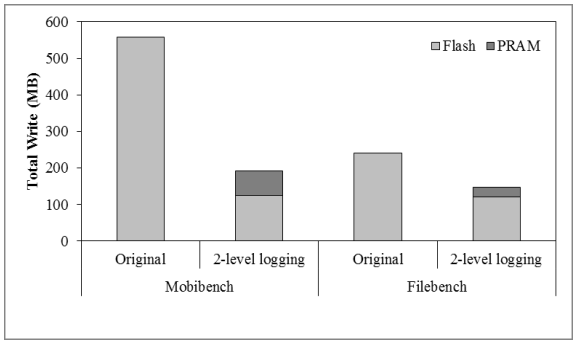

Figure 2. Write traffic comparison

Table 1. Performance comparison

Benchmark	measurement	Original	2-level logging
mobibench	bandwidth (txs/sec)	33	72
filebench	latency (ms)	19.9	4.3

4. ACKNOWLEDGMENTS

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2013R1A1A2A10013598).

5. REFERENCES

[1] J. Kim, et al., "Reducing excessive journaling overhead with small-sized NVRAM for mobile devices," *IEEE Trans. on Consumer Electronics,* 60.2 (2014): 217-224.
[2] C. Lee, et al., "F2FS: A New File System for Flash Storage," *FAST* 2015.
[3] T. Lee, et al., "FPGA-based prototyping systems for emerging memory technologies," *RSP* 2014.