IEICE Electronics Express, Vol.*, No.*, 1–6

3DFTL: A Three-Level Demand-Based Translation Strategy for MLC Flash Device

Peera Thontirawong[1a)], **Chundong Wang**[2], **Weng-Fai Wong**[3], **Mongkol Ekpanyapong**[4], **and Prabhas Chongstitvatana**[1b)]
[1] *Faculty of Engineering, Chulalongkorn University*
Phaya Thai Road, Wangmai, Pathumwan, Bangkok 10330, Thailand
[2] *Data Storage Institute, Agency for Science, Technology & Research*
5 Engineering Drive 1, Singapore 117608, Singapore
[3] *School of Computing, National University of Singapore*
13 Computing Drive, Singapore 117417, Singapore
[4] *School of Engineering and Technology, Asian Institute of Technology*
Km. 42, Paholyothin Highway, Klong Luang, Pathumthani 12120, Thailand
a) *peera.t@student.chula.ac.th*
b) *prabhas@chula.ac.th*

Abstract: 3DFTL is a demand-based flash translation layer (demand-based FTL) that can withstand caching data loss due to unexpected events such as power-loss. Its mapping table in the flash memory is designed with the capabilities of being instantaneously updated with zero additional write operations. Moreover, the average cache miss penalty of 3DFTL is also lower than previous demand-based FTLs. As a result, not only the mapping table of 3DFTL guarantees data consistency, but 3DFTL also shows 14.33% decrease in terms of the average system response time comparing with the DFTL.
Keywords: flash memory, MLC, FTL, cache, three-level, compression
Classification: Storage technology

References

[1] D. Ma, J. Feng and G. Li: ACM Comput. Surv. **46** (2014) 36.
[2] A. Gupta, Y. Kim and B. Urgaonkar: ASPLOS (2009) 229.
[3] Micron Technology: MT29F64G08CBAA datasheet (2009) http://www.micron.com.
[4] Storage Performance Council: Traces (2007) http://www.storageperformance.org.
[5] D. Narayanan, A. Donnelly and A. Rowstron: Trans. Storage **4** (2008) 10.
[6] Z. Qin, Y. Wang, D. Liu and Z. Shao: RTAS (2011) 157.
[7] P. Thontirawong, M. Ekpanyapong, and P. Chongstitvatana: ICSEC (2014) 108.

1 Introduction

As flash memory outperforms ferromagnetic materials on access latency, shock resistance, and power consumption, it is more preferred for data stor-

age of computer systems. It can be utilized in many devices, for instance, USB drives, solid-state drives, or even the embedded storage of smartphones. However, the NAND flash memory has several limitations [1]. E.g., a page, the smallest operable unit, has to be erased before reprograming. Since the smallest erasable unit is a block of pages, an out-of-place update is more feasible than an in-place update. The lifespan of a flash memory is limited by program/erase (P/E) cycles, and this limit is lowered with the MLC technique or a smaller fabrication process. As a consequence, a flash translation layer (FTL) is required for handling these limitations. An FTL enables out-of-place update by deploying address translation. It converts a logical page number (LPN), which is referred to by the file system, to a physical page number (PPN) of a flash memory. Since a page is hundreds times smaller than a block, performing address translation at the page-level is necessary for lowering P/E cycles.

An FTL that offers the page-level address translation is called a page-level FTL [1]. A page-level FTL is found on a page-level mapping table (PMT). However, PMT is huge and takes spacious SRAM capacity. In order to save the SRAM space, a demand-based FTL was proposed [2]. A demand-based FTL swaps the enormous PMT between SRAM and flash pages. These pages are called translation pages and can be located by a smaller mapping table that resides in SRAM. An address translation is done by a two-level process. The first level is getting the translation page number from the small table in SRAM, and the second level retrieves the PPN from the read translation page. The retrieved PPN is cached in SRAM for quick reference. The cache is managed by the write-back policy in order to lower program operations, but the delay of the translation page update makes the whole update become a non-atomic operation. Consequently, an inconsistency problem between data pages and translation pages is arisen. This problem is very important for an FTL because the flash memory is vastly employed in mobile devices that have many unexpected events, for example, power-loss. The FTL has to tolerate such and ensures correctness of data locations.

In this paper, we propose a novel demand-based FTL named 3DFTL. Without translation pages, updating data does not require additional page programing; hence, it is inconsistency-free. The cache miss ratio of 3DFTL is kept low by spatial locality exploitation. In addition, omitting translation page programing also decreases the maximum cache miss penalty, which in turn improves the average system response time.

2 Demand-Based Three-Level Address Translation

3DFTL is a page-level FTL with a cache for the mapping table. Typically, demand-based FTLs reduce the spatial requirement of SRAM by moving PMT to data areas of flash memory pages, which in turn causes the inconsistency problem. To overcome this obstacle, 3DFTL places the PMT entries in the spare area of the pages that store their corresponding data instead. As both data and mapping information are stored in the same page, updating

is considered as an atomic operation.

Since the spare area is much smaller than the data area, PMT demands more pages for storing. Adopting the two-level address translation, as same as typical demand-based FTL, will result in large global mapping table (GMT), which resides in SRAM. In order to save precious SRAM capacity, 3DFTL decreases the number of GMT entries by inserting an intermediate mapping table (IMT) between GMT and PMT. In other words, 3DFTL employs three-level address translation via GMT, IMT, and PMT. IMT entries are also placed in the spare area of pages along with PMT entries.

In the first level of address translation, GMT maps an LPN to the PPN of the corresponding IMT entry, and then the IMT entry points to the PMT entry in the second level. Consequently, the PPN of the data, which is kept in PMT, can be retrieved in the third level. Since two read operations are involved, one for an IMT entry and another for a PMT entry, the address translation could take longer time. However, the second read operation can be omitted if the required PMT entry is in the same page as the retrieved IMT entry. In other words, packing more PMT entries in one page can decrease the number of read operations. Due to the spatial locality of data write requests, the most significant bits (MSBs) of PPNs of nearby LPNs are having high likelihood of repetition. 3DFTL takes advantage of this property by employing a compression technique in order to make room for more PMT entries. A PPN, which is the content of IMT and PMT entry, is split into two parts: index (MSBs) and offset. A duplicated index is omitted from the spare area; hence, extra PPNs can be stored.

As illustrated in Fig. 1, every spare area has a compression flag for indicating the format of its metadata. In this example, PPN9 is uncompressed while PPN6 and PPN11 are compressed. Both formats contain an LPN, IMT entries, and PMT entries; however, a compressed format can have more PMT entries. An uncompressed format keeps its contents unaltered since it is used in case of very low compressibility to ensure mapping integrity, while a compressed format stores them as a collection of distinct indices and several pairs of index position and offset. The creation of a compressed metadata begins with splitting every PMT entry that associated with the same GMT entry into index and offset. Then, a PMT entry is replaced with the related IMT entry until every distinct index can be packed into the compressed format. Since the PPN of a page has to be known before accessing, keeping it in the spare area is unnecessary. Therefore, the PPN of the compressing metadata will be replaced by its LPN, which is indispensable for garbage collection and recovery, in the next step. The LPN is also split into an index and an offset, but the index is the least significant bits (LSBs). After substituting the PPN with the LPN, the last step is sorting the indices in chronological order so that IMT entries, which always equal to their latest PMT entries, can be identified. Furthermore, the index of the LPN always holds at the first position. Combining with the fact that offsets are ordered by the LSBs of LPN, the LPN of the compressed metadata can be recomposed.

Another important component of 3DFTL is a cache of PMT called CMT.

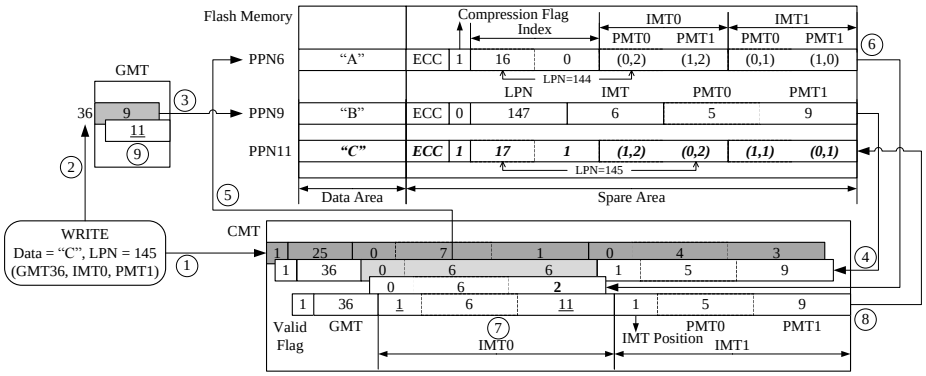

Fig. 1. The example of 3DFTL address translation.

The objective of CMT is to accelerate the address translation by exploiting temporal and spatial locality. A cache lines resembles an extracted compressed metadata, except that IMT entries are pointers to their latest PMT entries. Hence, each cache line has sufficient information for generating either a compressed or uncompressed metadata without accessing the flash memory.

An example of address translation is illustrated in Fig. 1. We assume that, firstly, both LPN and PPN are 8-bit. Secondly, an LPN can be broken into addresses of 6-bit GMT, 1-bit IMT, and 1-bit of PMT. In other words, each GMT entry has two IMT entries, and each IMT entry has two PMT entries. Lastly, each compressed metadata is limited to only two distinct indices. In the beginning of a request for writing C at LPN145 (GMT36, IMT0, and PMT1), 3DFTL searches for GMT36 in CMT and the lookup results in a cache miss. GMT25 is selected as a victim by LRU policy and can be evicted immediately because of write-through policy. Since the PPN of LPN145 has not been cached, a three-level address translation is required. The IMT entries of GMT36 are located to be in the spare area of PPN9 according to GMT in step 2. PPN9 is read in step 3 and will be cached in step 4. However, the PPN9 metadata is uncompressed; it does not contain all PMT entries associated with GMT36. CMT will store the IMT value in place of the missing PMT entries. As the LPN of this page is 147 (GMT36, IMT1, PMT1), two PMT entries in PPN9 belong to IMT1. In other words, the PPN of IMT1 is 9, and the PPN6 in the IMT field belongs to IMT0. Since the PMT entries of IMT0 is in another page, only the second level of the address translation, accessing IMT, can be done. Next, PPN6 is read for the PMT entries of IMT0 in step 5. The PPN6 metadata is compressed. This page is PMT0 in IMT0 according to the first index, which is LSBs of the LPN. Hence, the LPN is 144 and the PMT entries are 6, 2, 5, and 0 after substituting the first index and the offset of PMT0 in IMT0 by the index and offset of the current PPN6. In this step, the PPN of LPN145 is known to be 2. The PMT entries of IMT0 will be merged to the cache line of GMT36 in step 6. In order to update LPN145, PPN2 will be invalidated and replaced by PPN11, an empty page, in step 7. In step 8, the updated cache line of GMT36 is compressed and written along with data C to PPN11. Finally, the record 36 of GMT is updated according to the modification.

3 Evaluation

The experiments were conducted by simulating an 8GB MLC NAND flash memory [3] with following parameters. It has 4096 blocks of 256 pages. The data area of a page is 8192B while the spare area is 448B. However, only 112B are usable because of ECC. A page read, a page program, and a block erase operations take 75μs, 1300μs, and 3800μs, respectively. The data transfer rate is limited to 50MB/s. Benchmarks from SPC [4] and MSRC [5] were used for performance evaluation. 3DFTL will be compared against DFTL [2], CDFTL [6], and SCFTL [7]. DFTL is the baseline of demand-based FTLs while CDFTL added the second-level cache in order to exploit spatial locality. SCFTL is a high performance FTL that optimized for spatial locality and large page size. The SRAM sizes of 3DFTL, DFTL, CDFTL, and SCFTL were configured to 96.02KB, 101.00KB, 99.16KB and 101.19KB, respectively. The GMT of 3DFTL takes 64KB because few IMT and PMT entries can be packed into a spare area, and only 32KB is left for the CMT. In other FTLs, their GMTs are only about 2KB, and their CMTs are over 96KB.

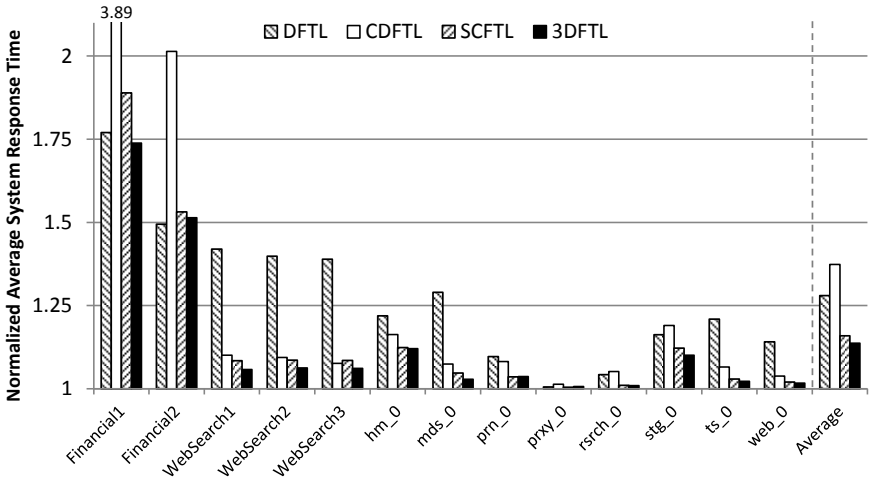

Fig. 2. The normalized average system response times.

As shown in Fig. 2, the average system response time of 3DFTL is the best comparing with other techniques even though its cache size is about one third of the others. The average system response time of 3DFTL is 14.33%, 23.67%, and 2.25% lower than DFTL, CDFTL, and SCFTL, respectively. Besides the low cache miss rate, which is caused by the spatial locality exploitation, the low cache miss penalty is also a major contributor for enhancing the performance. The maximum cache miss penalty of 3DFTL is only two read operations in the worst case. Owing to the compression, the worst case rarely occurs as shown in Fig. 3. On the contrary, the cache miss penalty of other FTLs may include updating a translation page. Since a page programing itself is over ten times slower than reading, the cache miss penalty is considerably high. Furthermore, it may trigger a garbage collection that requires even longer time.

An impact of cache miss penalty is clearly shown in *Financial1* bench-

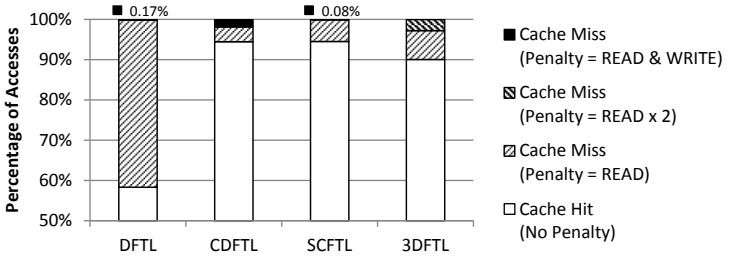

Fig. 3. The percentage of address translation cost.

mark that contains write-intensive requests. CDFTL, which has few large cache lines, exhibits very high overall cache miss penalty since 46.04% of its cache misses needs to update translation pages. As a result, CDFTL is drastically slow even though its cache miss ratio is very low. Moreover, DFTL and SCFTL are also subject to high miss penalty during very stressing cache accesses. However, our proposed FTL, 3DFTL, maintains low cache miss penalty. Regardless of smaller cache size, 3DFTL outperforms other FTLs and even surpasses, the high performance, SCFTL.

As previously stated, 3DFTL not only solves the inconsistency problem, but also enhances the performance. In addition, 3DFTL provides better flash space utilization since it does not occupy special pages for the mapping table. For this reason, 3DFTL shows slight improvement in terms of P/E cycles, which also means prolonging flash memory lifetime.

4 Conclusion

In this paper, a novel demand-based FTL named 3DFTL is proposed. It does address translation at the page-level and employs a cache of the mapping table like other demand-based FTLs. Differently, 3DFTL gets rid of translation pages by utilizing the spare areas of flash memory pages. Since the mapping information and data are simultaneously stored, the inconsistency problem is creased to exist; hence, fault tolerance is improved. However, keeping the locations of the page-level mapping table that stored in many little spare areas demands large SRAM. Thus, the three-level address translation is required for controlling SRAM size. The compression and caching techniques have been applied in order to exploit the spatial locality. The average cache miss penalty is very low owing to zero explicit cache write-back operations. To sum up, 3DFTL is an economical inconsistency-free high-performance demand-based FTL. 3DFTL is more suitable for managing the flash memory in a high performance mobile device than other demand-based FTLs.

Acknowledgments

Peera Thontirawong is in RGJ Ph.D. program by TRF (PHD/0273/2549).