Duplication-assisted reliability enhancement in flash storage system*

Ki-Jin Kim and Seung-Ho Lim
Division of Computer and Electronic Systems Engineering, Hankuk University of Foreign Studies, 89 Wangsan-ri, Mohyeon-Myeon, Yongin-si, Korea
a) slim@hufs.ac.kr; Corresponding Author

Abstract: To recover the errors occurred in flash-based storage, well-known error correction codes (ECC) are integrated into flash memory controller. However, the existing error correction codes have their inherent limits to recover more error than they recover, so more additional schemes should be considered for the higher reliable flash storage systems. In this paper, we investigate duplicated data management to enhance data reliability as replication. To minimize space overhead of the duplication management, we consider two issues, minimizing fingerprint length and enlarging hash chunk size of duplication checking. At most cases, the proposed duplication management scheme can have better reliability than adding more ECC parity bits in the aspect of space overhead.

Keywords: ECC, duplication, fingerprint, hash chunk, parity bits

Classification: Circuits and modules for storage

References

[1] N. Mielke, et al.: “Bit error rate in NAND flash memories,” IRPS (2008) 9 (DOI: 10.1109/RELPHY.2008.4558857).
[2] Y. Cai, et al.: “Program interference in MLC NAND flash memory: Characterization, modeling, and mitigation,” ICCD (2013) 123 (DOI: 10.1109/ICCD.2013.6657034).
[3] Micron: Enabling Software BCH Error Correction Code (ECC) on a Linux Platform, Technical Note, TN-29-71 (2012).
[4] K. Zhao, et al.: “LDPC-in-SSD: Making advanced error correction codes work effectively in solid state drives,” 11th USENIX FAST (2013) 243.
[5] F. Chen, et al.: “CAFTL: A context-aware flash translation layer enhancing the lifespan of flash memory based solid state drives,” 9th USENIX FAST (2011) 1.
[6] J. Park, et al.: DAC: Dedup-assisted compression scheme for improving lifetime of NAND storage systems,” DATE (2017) 1249 (DOI: 10.23919/DATE.2017.7927181).
[7] Memory Technology Devices, Memory Technology Devices Overview (2017) http://www.linux-mtd.infradead.org/.
[8] J. Yang: “Novel ECC architecture enhances storage system reliability,” Flash Memory Summit (2012) 1.
[9] E. Yaakobi, et al.: “Characterization and error-correcting codes for TLC flash

*The Research was supported by the NRF grant funded by the MEST (NRF-2016R1C1B2009914). This work was supported by Hankuk University of Foreign Studies Research Fund.
memories.” ICNC (2012) 486 (DOI: 10.1109/ICCNC.2012.6167470).

[10] E. Deal: Trends in NAND Flash Memory Error Correction Cyclic Design, White Paper (2009).

1 Introduction

The density of Flash memory has increased by moving to smaller geometries and storing more bits per cell. Currently, three-level cell (TLC) is main stream for NAND flash storage, in which it stores three bits per cell. The growth of number of states per cell raises interference between states since the quantized decision levels of the cell are getting close between adjacent states, which makes errors of detection. In addition, the more bits of data the cell has, the fewer endurance it supports, so the increase of density results in degradation of endurance of flash memory [1, 2]. The severe degradation of endurance and excessive errors are expenses for the density increase. The typical approach against the short endurance and increasing errors is use of Error Correction Codes (ECC) [3, 4] stored in the Out-Of-Band (OOB) area of each page. However, the growing error due to the increasing density requires larger area for ECCs and higher ECC engine complexity. Moreover, ECCs faces the difficulty for correcting burst errors, page, block, and chip-level errors.

Data duplication in storage systems is very common. For example, user can create/delete same files several times, and applications may save backups for currently used data as a frequent manner, and so on. Statistically, there is 5% to 40%, or more or less data duplications in general storage systems [5], which is not negligible at the aspect of replication. In other words, the naturally generated duplication can be used for recovery of error-occurred data if duplication exists. However, the duplication management can be burden for both of space and runtime duplication checking, in comparison with enlarging existing ECC capability.

In this paper, we investigate data duplication management as an aspect of replication and enhancing data reliability with replication. Likewise other duplication techniques [5, 6], hash-based Fingerprint store is also managed in this paper, however we consider two technical issues to minimize the duplication management overhead as an aspect of replication. First, duplicated data are also written at all, so we don’t need strict hash values for checking duplication at runtime. The hash values are used as hint for duplication checking with minimal length, and exact duplication is checked at idle time. It can reduce space and runtime computational overhead. Second, we enlarge the chunk size of duplication checking with group of several pages, such as double or quadruple. Since duplication would be performed at file level at most times, it has high possibility to have duplication at larger chunk size. The larger chunk size can reduce space overhead of Fingerprint store. We identify that the proposed scheme can enhance reliability level in comparison with enlarging ECC parity bits at the same space overhead.
2 Duplication-assisted reliability enhancement system

2.1 Fingerprint store management in running time

The design of Duplication-assisted Reliability Enhancement System (DRES), aims to maintain duplication table with two technical issues described above for minimal space overhead. Generally in deduplication techniques [5, 6], strong hash functions should be used for very low collisions, since it does not allow false positive. In contrast, false positive is allowed in our system since all data eventually written to storage whether it is duplicated or not. Hence, collisions with low probability are allowed in fingerprint store. Also, duplications do not need to be checked at real time since those are not used right after at most cases. In DRES, there are two phases for making up of duplication table; the first is fingerprint management phase during request running time, and second is update of duplication table using fingerprint store during request idle time.

The fingerprint store management at request running time is described in Fig. 1. When write requests are received at flash storage device, those are first buffered in the on-device buffer of flash device. In-buffer write requests are grouped together to make a hash chunk for making fingerprint with the condition that requests are logically consecutive. For example, when hash chunk size is quadruple of page size, the four logically consecutive requests are grouped and hashed to make fingerprint. After chunking of requests, it is hashed with light-weight hashing function that allows collision. Then the resulting value is stored in fingerprint store. If conflict value does not exist in the store, the fingerprint is inserted into the store with logical address, i.e., logical page number. If it is exist, the address of that is added to the existing fingerprint. Hash value with 4-byte length is enough to check duplication hint having low collision rate, as we will show at the evaluation. The 4-byte location denotes the logical address information, so there is 8-byte for each element if no duplication exists. Elements with duplication consume over 12 bytes. Since the element are generated for each hash chunk which is several times than page, moreover, distribution for hash values is skewed [5], the overall space for fingerprint store can be limited in DRES.

![Fig. 1. Limited fingerprint store management for request running time](image-url)
2.2 Update of duplication mapping table

The inspection for exact duplication and update of corresponding mapping table are done with the fingerprint store. This work is being done at idle time because of heavy inspection is required. The duplication mapping table management is described in Fig. 2, in that we consider page level mapping table in designing mapping table. The 4-byte Duplication Page Number (DPN) is added to page-level mapping table for each hash chunk along with the mapping table entries. It denotes the address for duplicated data of its own. Since DPN is required for each hash chunk not every page, the space overhead can be reduced multiple times than page level. For example in the figure, DPN is allocated at every four pages as hash chunk is quadruple of page. In this case, it is like that 1-byte is added to each entry in conventional mapping table, which consumes less space overhead.

The update process is as follows. For each element in fingerprint store, if there is one address, which means there is no duplication, so we skip it. If there exist two addresses that match hash value, two data according to the denoted addresses can be examined whether those are exactly matched each other using heavy hashing algorithm. If it is true, the DPN of corresponding LPN is updated each other. For example in the figure, if fingerprint element has address 1 and 12, and those data have match each other, then DPN for LPN 0 is recorded with 12, and DPN for LPN 12 is recorded with 0. With this duplication mapping table, we can recover error generated region having duplication by searching DPN base and its offset within hash chunk. For instance, if error occurs during reading LPN 14, data can be recovered from the physical location of LPN 2.

3 Experiments

The proposed schemes are implemented within Linux MTD layer in Linux Kernel 3.18.17 [7]. For the simulation, NANDSim in Linux MTD has been used to simulate NAND flash devices with 8KB page sizes and 1MB block sizes. As NANDSim does not provide bit error generation in accordance with P/E cycle increment, we used TLC error modeling for the behavior of flash memory cell, based on the previous research works [8, 9]. For the encoding scheme, BCH ECC encoding is used since it is default ECC in Linux MTD layer [3]. As to evaluate
the proposed schemes on different user environments, we collected 3 file system disk traces and 2 database traces used in our departments for developing and maintenance.

At first, we evaluate the impact of hash length and hash chunk size in DRES, since those two configuration parameters have influence on space overhead as well as duplication rate. We have collected collision counts for each traces with various hash length from 1 byte to 20 bytes with SHA-1 hashing. The Fig. 3a plots discriminate rate over hash length. In the figure, y-axis represents relative discriminative rate. As discriminative rate is getting higher, false positive rate is getting lower. From the figure, we identify that 4 bytes hash length is enough to achieve nearly same collision probability as 20 bytes, so using 4 bytes as hash value is reasonable choice in the aspect of performance and space overhead. Fig. 3b shows duplication rate of each traces as hash chunk size increase from 8 KB to 128 KB. From the figure, we identify that the duplication rate is gradually decrease as hash chunk increase, however the duplication drop is not so much. For example, if we compare the duplication rate of 8 KB and 32 KB, about 15% duplication rate is dropped in average, however, absolute gap between two is not so large. Therefore, large hash chunk can reduce space overhead with small sacrifice of duplication rate.

Next, we have evaluated reliability enhancement with DRES. To see the feasibility of DRES with various space overheads, we evaluate DRES with several configurations by varying hash chunk size and limiting fingerprint store size. For fair comparison with ECC, same amount of parity bits are added for ECC by calculating the overhead of DRES. The space overhead of DRES can be evaluated about 6 bytes, 4 bytes, 2 bytes and 1 byte per page, when hash chunk is 8 KB, 16 KB, 32 KB, and 64 KB, respectively. In theory, 14 bits are required for increasing 1 another bit error correction for 1 KB sub-page in BCH ECC [10]. To be compared fairly, we enlarge ECC parity length 6, 4, 2, and 1 bytes of original ECC in accordance with hash chunk. Fig. 4 shows evaluation results, in which, x-axis represents P/E cycles and y-axis represents Uncorrected Page Error Rate (UPER). In the figures, Original and ECCadd plot UPER for original ECC scheme and enlarging ECC parity bits, respectively, while dup – x plots results for DRES with original ECC scheme for five IO traces. Among all plots, the results of ECCadd are plotted with thick line in all figures for easy comparison. From the results, we
identify that DRES gives better reliability at higher duplication rates such as \( \text{dup} - f_1 \) and \( \text{dup} - f_3 \) while enlarging ECC parity bits gives better performance than lower duplication rate such as \( \text{dup} - f_2 \) and \( \text{dup} - d_2 \) at small hash chunk, as shown in Fig. 4a. However, as hash chunk size increases, DRES shows better performance than enlarging ECC parity bits even if small amount of duplication at the same space overhead, as shown in Fig. 4c and Fig. 4d.

4 Conclusion

Typical approach against the short endurance and increasing errors is use of ECC, however, current state of ECC scheme does not catch up with the evolution of errors of current state of flash memory. It is required to consider other error recovery schemes at system level. In this paper, we investigate duplication management for data reliability. In the design of duplication fingerprint and mapping table, we consider two issues to minimize space overhead, the one is minimizing fingerprint length and the other is enlarging hash chunk for duplication checking. The duplication management can have better performance than adding more ECC parity bits at most cases.