Advanced Science and Technology Letters
Vol.98 (CES-CUBE 2015), pp.73-76
http://dx.doi.org/10.14257/astl.205.98.19

A Light-Weight Redundancy Technique for Limited Embedded Flash Storage Device

Seung-Ho Lim

Division of Computer and Electronic Systems Engineering
Hankuk University of Foreign Studies
slim@hufs.ac.kr

Abstract. As the capacity of NAND Flash memory is getting larger, its raw bit error rate is also getting larger, which is crucial for developing storage system with NAND flash memory. To improve reliability of Flash memory-based storage device, several redundancy schemes were developed. However, the redundancies give much overhead inherently, management for the redundancy in embedded flash memory system was not preferred. In this paper, we propose light-weight redundancy technique for limited embedded flash storage device. In the light-weight redundancy scheme, redundancy is generated as the same way of RAID, with the cluster of RAID is composed of in-block level. Foe each incoming data in a block, parity data is updated within main memory until pages are exhausted for the block. The parity is flushed to last page for the block once. By doing this, block level redundancy is maintained with minimal overhead

Keywords: Flash Memory, RAID, redundancy, bit error rate

1 Introduction

Recently, the density of Flash memory has dramatically increased by moving to smaller geometries and storing more bits per cell with enhanced memory technologies. If one bit is represented by one cell, there are two levels in the one cell, and if four bits are represented by one cell, there are four levels in the one cell, and so on. Currently, three-level cell (TLC) is main stream for NAND flash storage [1]. As the capacity of NAND Flash memory is getting larger, its raw bit error rate is also getting larger, which is crucial for developing storage system with NAND flash memory. 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.

. To improve reliability of Flash memory-based storage device, several redundancy schemes were developed [4-7]. These schemes are mainly based on the Redundant Array of Inexpensive Disks (RAID) that was applied to traditional rotational disk. Generally, when RAID technique is applied to NAND flash-based system, the redundancies are created with cluster of several pages that are across to several blocks. The pages can be grouped across the blocks or chips to spread out the redundancies.

ISSN: 2287-1233 ASTL
Copyright © 2015 SERSC

However this spread out of redundancy generation and distributed places of these cause additional complex and operational overhead in both memory and IO operations.

In this paper, we propose light-weight redundancy technique for limited embedded flash storage device. In the light-weight redundancy scheme, redundancy is generated as the same way of RAID, with the cluster of RAID is composed of in-block level. Foe each incoming data in a block, parity data is updated within main memory until pages are exhausted for the block. The parity is flushed to last page for the block once. By doing this, block level redundancy is maintained with minimal overhead

2 Implementation of Light-Weight Redundancy Management in Main Memory

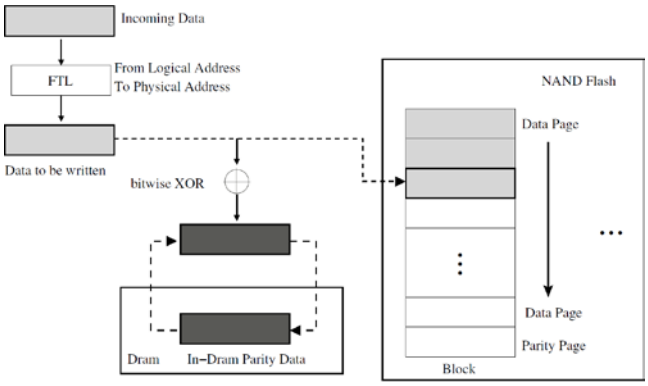

Fig. 1. Light-Weight Redundancy Generation in Main Memory

The proposed light-weight redundancy scheme was developed with Linux-based Flash simulator, with Linux Memory Technology Device (MTD) device driver layer [9].

The light-weight level redundancy scheme is described in Figure 1. In the scheme, a Flash memory block consists of one parity page and remaining pages are used for data store. The last page is preserved for parity page. With this composition, the redundancy generation is as follows. When data is arrived from host system, its corresponding address is logical address, so, at first in Storage system, FTL translate logical address to physical address [2,3,8]. Since FTL management is out of scope of this paper, we omit the specific description of the translation mechanism. After the FTL translation, the physical address of data is set. Then In light-weight redundancy scheme, the parity data is updated with incoming data within in main memory and it is maintained in Dram memory. The parity data is calculated with bitwise exclusive operation between in-dram parity and incoming data, and the updated parity data is maintained in Dram memory. The incoming data is then written to NAND flash memory. If the physical address of the incoming data is not the last-1 page of the

 Copyright © 2015 SERSC

block, the parity data is not flushed to NAND flash memory, but just maintained in the memory.

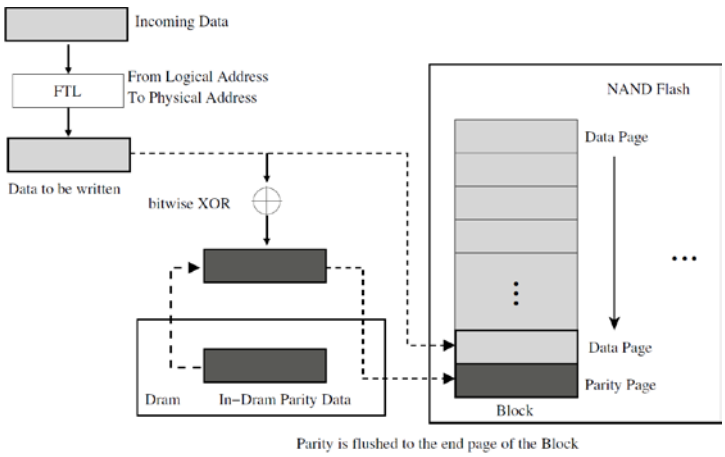

Fig. 2. Parity flush Sequence for Light-Weight Redundancy

When the physical address of incoming data is last-1 page of the block, the updated parity data is flushed on last page of the block after calculating new parity, as shown in the Figure 2. The physical address of current incoming data is last-1 page means all the pages in that block are consumed to store data. The last page of the block is used to store parity data of the pages in the block, i.e., each bit of the parity page represents bitwise exclusive-or data of pages above all the pages in the block even if the page has invalid data due to data update. The parity in Dram memory is cleaned and ready for another block.

3 Conclusion

In this paper, we propose light-weight parity redundancy scheme for providing reliability of flash memory based storage system with minimal redundancy overhead. In the proposed scheme, the redundancy is generated in flash memory block level and the generated redundancy is stored at the end page of a flash block. Usually, the programing operation of flash memory has a logging manner, which means that the write of pages in a flash block is from first page to last page. During the programming stage of a flash block, the redundancy is kept in Dram memory until all data pages are exhausted. The in-keeping redundancy is used to re-calculated up-to-date redundancy with incoming data. If a block is exhausted to last-1 page to record incoming data, the in-keeping redundancy of main memory is flushed to last page of the block. Bo doing this, redundancy overhead can be minimized in IO operations.

Copyright © 2015 SERSC

References

1. Flash Memory, http://en.wikipedia.org/wiki/Flash_memory1.
2. A. Ban, Flash file system, United States Patent, no.5,404,485, 1995.
3. Intel Corporation, "Understanding the flash translation layer(FTL) specification", http://developer.intel.com/.
4. P.M. Chen, E.K. Lee, G.A. Gibson, R.H. Katz, and D.A. Patterson, "RAID: High-Performance, Reliable Secondary Storage," ACM Computing Surveys, vol. 26, no. 2, pp. 145-185, 1994.
5. S. Im and D. Shin. "Flash-Aware RAID Techniques for Dependable and High-Performance Flash Memory SSD", IEEE Transactions on Computers, 60(1):80 92, Jan. 2011.
6. J. Kim, J. Lee, J. Choi, D. Lee, and S. Noh, "Enhancing SSD Reliability through Efficient RAID Support", in Proceedings of 3th Asia-Pacific Workshop on Systems, July, 2012.
7. Y. Qin, D, Feng, J. Liu, W. Tong, Y. Hu, and Z, Zhu, "A Parity Scheme to Enhance Reliability for SSDs", in Proceedings of 7th International Conference on Networking, Architecture, and Storage, 2012
8. Seung-Ho Lim, "Implementation of metadata logging and power loss recovery for page-mapping FTL", IEICE Electronics Express, Vol. 10, No. 11 pp1-6, May, 2013.
9. Memory Technology Devices, "Memory Technology Device Overview", Retrieved 1 September 2012. http://www.linux-mtd.infradead.org/

 Copyright © 2015 SERSC