Computers and Electrical Engineering 37 (2011) 1122–1136

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

DeFFS: Duplication-eliminated flash file system ☆

Seung-Ho Lim *

Department of Digital Information Engineering, Hankuk University of Foreign Studies, Yongin 449-791, Republic of Korea

ARTICLE INFO

Article history:
Received 19 September 2009
Received in revised form 20 June 2011
Accepted 20 June 2011
Available online 20 July 2011

ABSTRACT

Whenever files are modified, large parts of existing data must get unnecessarily re-written to storage due to the inefficiency on identifying those portions of the files that are actually new in the latest update. The unmodified data are considered as duplicate data since these do not have to be re-written. If NAND flash memory is used for storage, it is beneficial to reduce the duplicate data as many as possible. The issue is how to identify and eliminate the duplicate region efficiently. In this paper, the advanced architecture of flash file system, called duplication-eliminated flash file system, is introduced for duplicate elimination. The important design issues supporting duplicate elimination are how to manage data blocks and how to detect duplicate region. In the DeFFS, index entries of inodes support variable-sized blocks in order to increase the manageability and flexibility of duplicate regions. In addition, DeFFS uses non-overlapping duplicate checking algorithm to reduce the complexity of duplicate checking algorithm. The duplicate elimination can prolong flash memory life cycles by reducing actual amount of page writes, and increase write bandwidth.

1. Introduction

Data duplication is becoming increasingly common in storage system. There exist a lot of systems that have content similarity in their storage such as email servers, virtualized servers, and version control systems. For email systems, mailing lists, mail-attachments can lead duplication. Virtualized servers run similar programs which make duplicate content, and for version control systems, copies of files are duplicated. In such storage systems, data deduplication can optimize in both storage capacity utilization and I/O utilization. The elimination of duplicate content at both file and block levels for improving storage utilization has been an active research area [1,4,7,10–12,23,24].

In the previous research, storage systems exploit the content-based hash to name and identify data blocks. If the data blocks are hashed with a robust one-way hash function such as SHA-1 [2,3], the result key is unique with high probability. Thus, if hashes of two objects are equal, the system can identify that the corresponding two blocks are equal in their contents, so duplicate data can be excluded from the sending or writing operations. There are two methods in partitioning data blocks and detecting duplication; the first is deduplication with fixed-sized chunks and the other is deduplication with variable-sized chunks. The fixed-sized chunks scheme is used in [5–7]. In these schemes, file contents are hashed by a whole file or by fixed-sized chunks. Since the file can be divided by simple physical offsets, the duplication can be detected by comparison between the fixed size chunks with low complexity. However, the effectiveness of this approach is highly sensitive to the sequence of data modifications. The variable-sized partitioning divides file or object into variable-sized chunks by the contents [8,9,13]. The variable-size partitioning can reduce sensitivity of data modification by localizing changes of blocks to a few chunks around the region of change, since some insertion or modification of file is possible by enlarging or modifying

☆ Reviews processed and approved for publication to Lukowiak.

* Tel.: +82 31 330 4704.
E-mail address: slim@hufs.ac.kr

0045-7906/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compeleceng.2011.06.007

only the corresponding chunks. In general, the variable-sized chunks give better chance to manage data efficiently, even though we do not consider how variable-sized chunks are packed [14].

On the other hand, flash memory has become an increasingly important component in nonvolatile storage media because of its small size, shock resistance, and low power consumption [15–17]. Recently, the capacity of a NAND flash memory chip became huge enough, and it will increase quickly. Based on the NAND flash chip, solid state disks (SSD) have been deployed as storage devices in computer system including personal computers and server storage systems [18]. If we consider data duplication with the consideration of storage media, duplicate data writes might be more efficient method when rotating-disk drives are used for the storage at sometimes. When data are written, new blocks are contiguously allocated from the free blocks pool, which makes sequential write request. The sequential request on the rotating-disk can increase bandwidth since it has no additional seek and rotational latency. However, when flash memory-based devices are used for storage, it is much more beneficial to eliminate the duplicate data as many as possible. It is because write bandwidth is getting worse as the amount of data increases. Moreover, due to the out-of-place update characteristics of flash memory, reducing write requests can also reduce erase operations of flash memory block. Thus, identification and elimination of duplicate data from the write request is very crucial for flash memory based storage system.

The issue is how to identify and eliminate the duplicate region in consideration with flash memory characteristics. In recent years, several flash memory based file systems have been proposed for more efficient management of NAND flash memory, including JFFS2 [19], YAFFS2 [20], CFFS [21]. However, these flash file systems are general purpose file systems and cannot support deduplication in their architecture. In this paper, the advanced architecture of flash file system is introduced for duplicate elimination in NAND flash memory, called duplication-eliminated flash file system (DeFFS). In the file system architecture, the most important design issue is how to manage data blocks and how to detect duplicate region efficiently. In the DeFFS, index entries of inodes support variable-sized blocks in order to increase the flexibility of duplicate region and to identify the amount of duplicate data many as possible. In addition, DeFFS uses non-overlapping duplicate checking algorithm to reduce the complexity of duplicate checking algorithm. To the best of our knowledge, this paper's work is the first work that designs and implements duplicate elimination for flash memory based file system. The DeFFS is implemented in Linux 2.6 kernel with MTD technology [22] and it is experimented with several benchmark programs. The experimental results show that deduplication reduces the huge amount of write data to real medium, as well as IO times for write operations.

The remainder is organized as follows. In Section 2, background and related work for flash memory based storage systems are described. The proposed architecture of flash file system for duplicate elimination is explained in Section 3. The detailed duplicate elimination algorithm of DeFFS is described in Section 4. Section 5 gives the performance evaluation in comparison with the previous flash file systems, YAFFS, and JFFS2. Section 6 concludes this paper.

2. Background

In this section, we describe the characteristics of NAND flash memory, and we introduce previous flash file systems.

2.1. Characteristics of NAND flash memory

NAND flash memory is widely used as storage media with its large capacity and relatively high performance capability for read and write operations. NAND flash memory chips consist of blocks; each block has a fixed number of pages. Typically, page size is 2 KB, 4 KB, or 8 KB, and block size is 64 KB, 128 KB, 256 KB, or more according to chip's types such as SLC and MLC. In NAND flash memory, read and write requests are performed in unit of page. Thus, even if an update of only several bytes is requested, the entire page including these bytes should be updated due to the characteristics of page-level updating. A page is divided into the data region and the spare region for storing page attributes. Due to flash memory characteristics in the form of electrically erasable and programmable read only memory (EEPROM), in-place updates are not allowed. Therefore when data of a page are modified, the data must be written to another free page, which is considered as a live page. Consequently, the page containing the old data is considered as a dead page. After write requests come, large portions of flash memory are composed of dead pages, and the system should reclaim the dead pages for write operations. The erase operation makes dead pages become available free pages. However, because the unit of an erase operation is a block, which is much larger than a write unit, this mismatch results in additional copying of live pages into another block when erasing a block. This process is called garbage collection.

2.2. Flash file systems

The physical features of flash memory described in the introduction derive the needs for native flash file system supporting flash memory characteristics. One of the typical design is the Journaling Flash File System 2 (JFFS2) [19]. The JFFS2 is a log-structured file system that sequentially stores the nodes containing data and metadata in every free region in the flash chip. However, in the design of the JFFS2, the NAND flash memory characteristics, such as the spare regions and read and write units, were not fully considered and utilized. Therefore the performance of the JFFS2 with NAND flash memory storage