World Academy of Science, Engineering and Technology
International Journal of Electrical and Computer Engineering
Vol:3, No:6, 2009

Mounting Time Reduction using Content-Based Block Management for NAND Flash File System

Won-Hee Cho, GeunHyung Lee, and Deok-Hwan Kim

***Abstract*—** The flash memory has many advantages such as low power consumption, strong shock resistance, fast I/O and non-volatility. And it is increasingly used in the mobile storage device. The YAFFS, one of the NAND flash file system, is widely used in the embedded device. However, the existing YAFFS takes long time to mount the file system because it scans whole spare areas in all pages of NAND flash memory. In order to solve this problem, we propose a new content-based flash file system using a mounting time reduction technique. The proposed method only scans partial spare areas of some special pages by using content-based block management. The experimental results show that the proposed method reduces the average mounting time by 87.2% comparing with JFFS2 and 69.9% comparing with YAFFS.

***Keywords*—**NAND Flash Memory, Mounting Time, YAFFS, JFFS2, Content-based Block management.

I. Introduction

FLASH memory increasingly becomes a popular storage medium in mobile devices, because it provides solid state storage with high reliability and high density at a relatively low cost[1]. Besides, the smaller size is the more suitable for mobile devices. However flash memory is 5~10 times more expensive than hard disk. The reading speed of the flash memory is very fast, but its erasing and writing speed is relatively slow [2].

Two popular types of flash memory are NOR flash memory and NAND flash memory. NOR flash memory offers faster reading speed and random access capabilities. It makes NOR flash memory suitable for code storage in devices such as PDAs and cell phones. However, with NOR flash memory technology, write and erase functions are slow compared to NAND flash memory. NOR flash memory also has a larger memory cell size than that of NAND flash memory so that it limits scaling capabilities and achievable bit density compared to NAND flash memory [3].

Flash memory is composed of blocks and pages. Entire flash memory is divided into blocks and each block is divided by the same number of pages. For small block, it consists of 32 pages and each page has a size of 512 bytes. On the other hand, for large block, it consists of 64 pages and each page has a size of 2048 bytes [3]. Fig. 1 shows the structure of the NAND flash memory.

Won-Hee Cho and Geun Hyung Lee are with the Department of Electronic Engineering, Inha University, Incheon, Rep. of Korea (e-mail: {chowon, ghlee}@iesl.inha.ac.kr).

Deok-Hwan Kim is with Department of Electronic Engineering, Inha University, Incheon, Rep. of Korea (corresponding author to provide phone: + 82-32-860-7424; fax: 82-32-868-3654; e-mail: deokhwan@ inha.ac.kr).

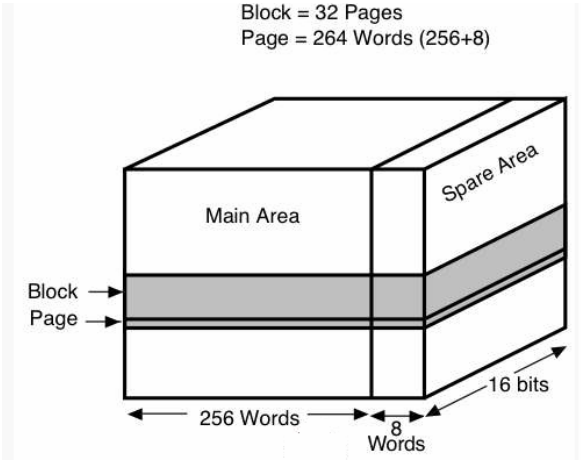


Fig. 1 The structure of [1]the NAND flash memory

Microsoft Flash File System is a flash native file system using linked-list data structure for NOR flash memory [4]. Later, Log-structured File System (LFS) structure [5] was adopted to flash memory file system. JFFS is the first flash memory file system which adopts LFS structure based on NOR flash memory [6]. Later, it was upgraded to JFFS2. It used in Linux operating system. JFFS2 was then later extended to support NAND flash memory, but it could not show optimized performance because NAND flash memory has different characteristics. Thus YAFFS (Yet Another Flash File System) was designed for NAND flash memory[9][10]. YAFFS is another log-structured flash file system which is optimized for NAND flash memory. It shows a better I/O performance than JFFS. However, YAFFS has an inherent mounting time problem since its mounting time increases rapidly as the size of flash memory becomes large.

In this paper, we propose a content-based flash file system using mounting time reduction technique. It only scans partial spare areas of some special pages, which includes header, by using content-based block management. The remainder of this paper is organized as follows: in section II. We review the NAND flash memory file system (YAFFS), and in Section III, we present the content-based block management for mounting time reduction. The experiments and the conclusion are in section IV and section V, respectively.

[1] This research was financially supported by the Ministry of Knowledge Economy(MKE) and Korea Industrial Technology Foundation (KOTEF) through the Human Resource Training Project for Strategic Technology, and ETRI SoC Industry Promotion Center, Human Resource Development Project for IT SoC Architect, and the Korea Research Foundation Grant funded by the Korean Government" (KRF-2008-313-D00822).

II. YAFFS (Yet Another Flash File System)

In the YAFFS, each page has data or metadata of file system. When the file system is mounted, it reads all metadata such as the header information in NAND flash memory and builds the binary tree structure of file system in memory [7]. Page is called the chunk and the chunk is composed of data area and spare area.

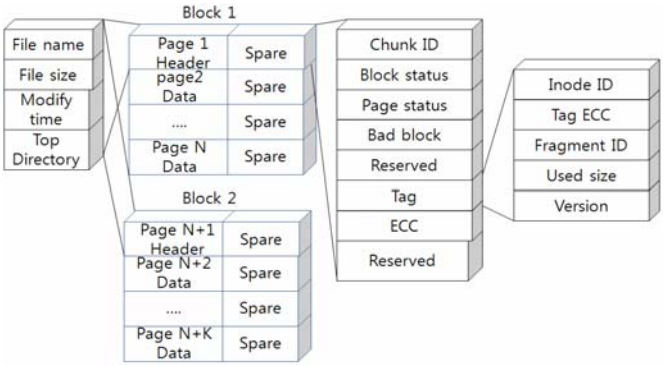


Fig. 2 The structure of the YAFFS

Fig. 2 shows the structure of the YAFFS in the case that a file is stored into two blocks. The file header information is stored in data area of the first page (chunkID 0) of each block, respectively. The header in data area is composed of file name, size, modify time and top directory. Unless chunkID is 0, data area of the page is filled with file data. The spare area of the page consists of chunkID, block status, page status, ECC area and tag, etc. Page status has an identifier representing valid or invalid page. Tag area consists of data size, file version and inodeID, etc.

III. Design and Implementation of Content-Based Block Management File System

The flash memory has different features compared with existing storage medium [8]. These features can be described as the following attributes. First, flash memory's physical inability is to prevent data from overwriting in same addresses. So, outplace update is indispensable for flash memory. Second, flash memory has an erasing limitation. Therefore, wear-leveling is another function for expanding the life time of flash memory. Recent flash file system such as YAFFS, JFFS2 provides such functions. However, their mounting time increase rapidly according as the size of flash memory increases.

In this paper, we present content-based YAFFS using content-based block management in order to reduce its mounting time.

1. Content-Based Block Management

Content-based YAFFS divides the block into pure, mixed and free types. The pure block is completely filled with file data. Mixed block is filled with one or more files and/or empty pages. Free block is composed of empty page only.

The YAFFS file system needs to read the whole spare areas to check the block type. On the other hand, content-based YAFFS determines the block type when writing the file. For example, suppose that a NAND flash memory is with total capacity of 512Mbyte, and it is composed of the block size of 128Kbyte and the page size of 2Kbyte. When 4.4Mbyte mp3 file is written, 35 pure blocks and 1 mixed block are created by applying formula (1).

$$p_n = \lfloor F_{size}/B_{size} \rfloor, \quad M_n = \lceil F_{size} \% B_{size} \rceil \tag{1}$$

Where p_n is the number of pure block, M_n is the number of mixed block, F_{size} is the file size, B_{size} is the block size.

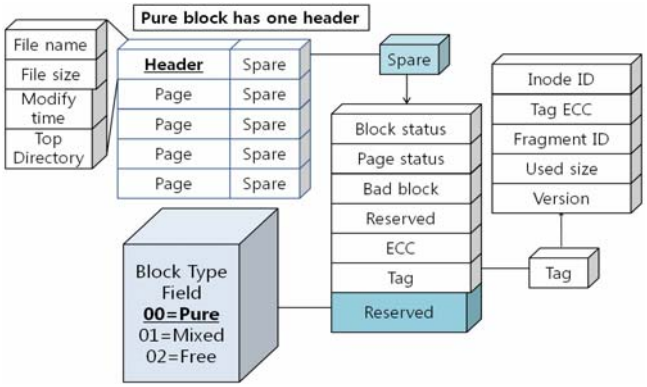


Fig. 3 The structure of the pure block

Fig. 3 shows the structure of the pure block. The header information is stored in the first page of the block. For the content-based block management, the block type field in reserved area of spare area of first page is set to “00”.

Fig. 4 shows the structure of the mixed block. Two headers information are stored in the pages of the block. In this case the block type field is set to “01”. The difference between pure block and mixed block is the number of headers. Pure block has only one header whereas mixed block has two or several headers. Therefore, by only scanning the spare area of first page, we can easily determine the block type.

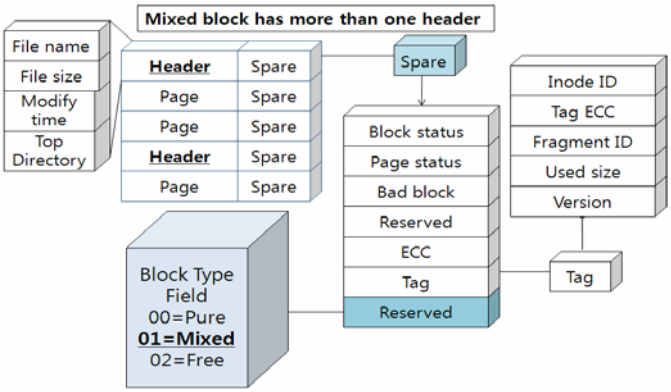


Fig. 4 The structure of the mixed block

2. Mounting Time Reduction

During the mounting time, series of data stored in the flash memory needs to be processed to produce the meaningful data. However, due to the limitation of flash memory, we can’t

reserve any space for this purpose. So, whenever file system is mounted, the open file table needs to be constructed through the file information in the header and Meta data in the spare area of every page of all blocks. Because of this, we can see that very long time is required to mount the file system.

In this paper, we propose a mounting time reduction technique by storing block type field in the spare area of the first page. It reduces the mounting time because we can build up the open file table by only scanning the header and spare area of the first page of each block.

Fig. 5 shows the mounting time of standard YAFFS. Since the block contents are unknown, spare areas of all pages in the block must be scanned to determine the block type. If we assume the scanning time of a spare area of the page as *T*, the mounting time will take more than or equal to 5**T*. In content-based YAFFS, pure block or mixed block can be identified through the block type field as shown in Fig.6. For the pure block, the same file is stored in the same block. In this case, the mounting time will be 1**T*. Likewise, for the mixed block, the same file can be stored in several blocks. In this case, mounting time will be the number of blocks in this file**T*.

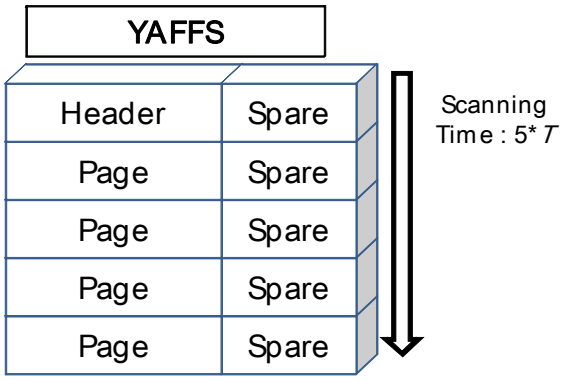


Fig. 5 Mounting time of the YAFFS

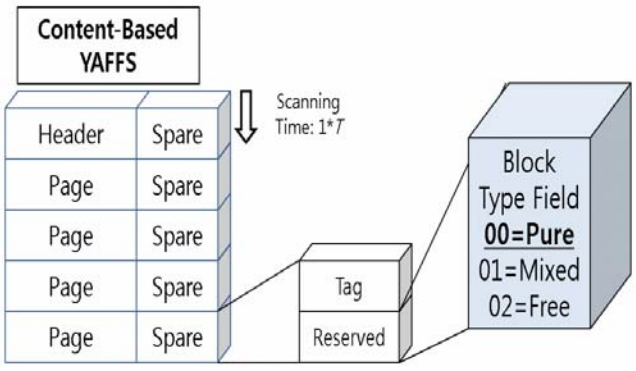


Fig. 6 Mounting time of the content-based YAFFS

After scanning, each block is managed by one of three linked lists. Fig. 7 shows the flow chart for content-based block management.

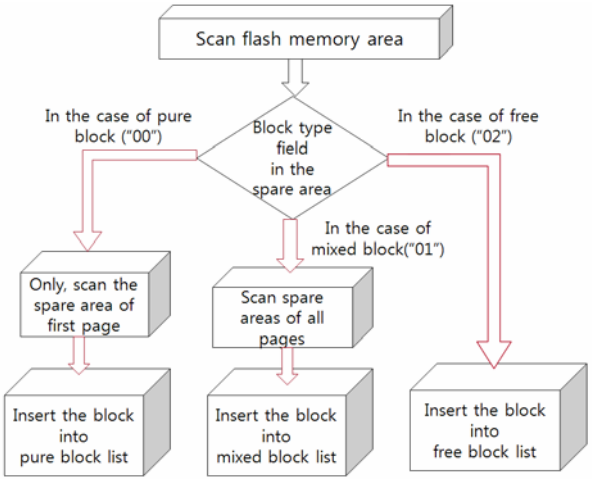
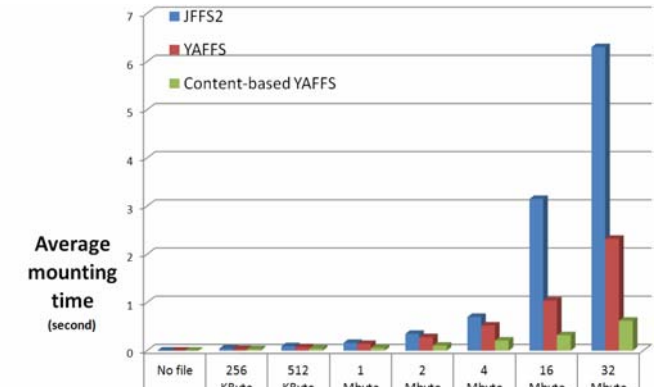


Fig. 7 Identification of block types

IV. Experiment

In this section, we compare the proposed method with other standard file systems like the YAFFS and JFFS2 in terms of mounting time. We conducted two experiments on Linux 2.6.17 environment. We simulated IPTV environment using standalone desktop and USB flash memory.

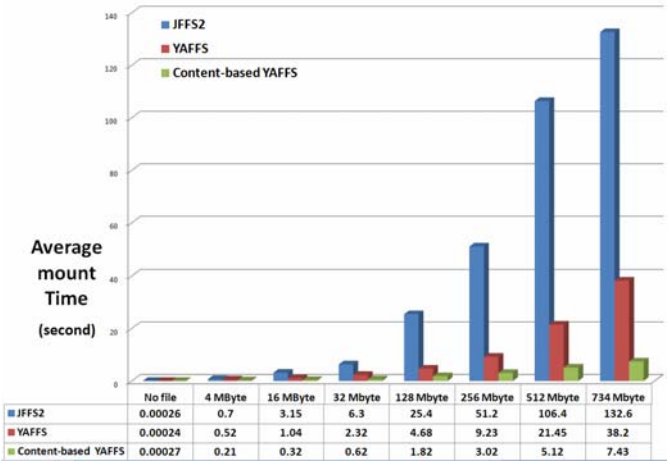
In the first experiment, we compared average mounting time of the proposed content-based YAFFS and the standard file systems using various small files. We measured the average mounting time by scanning the spare areas of the blocks in the flash memory. We used empty file and variable files of sizes with the range from 64Kbytes to 32Mbytes.



	No file	256 KByte	512 KByte	1 Mbyte	2 Mbyte	4 Mbyte	16 Mbyte	32 Mbyte
JFFS2	0.00026	0.045	0.09	0.16	0.35	0.7	3.15	6.3
YAFFS	0.00024	0.035	0.07	0.14	0.28	0.52	1.04	2.32
Content-based YAFFS	0.00027	0.031	0.044	0.056	0.105	0.21	0.32	0.62

Fig. 8 Average mounting time for JFFS2, YAFFS and content-based YAFFS using various small files

Fig. 8 shows the average mounting time for JFFS, YAFFS and content-based YAFFS using various small files. The result shows that the average mounting time of content-based YAFFS is 87% and 69% less than those of JFFS and YAFFS, respectively.



	No file	4 MByte	16 MByte	32 Mbyte	128 Mbyte	256 Mbyte	512 Mbyte	734 Mbyte
JFFS2	0.00026	0.7	3.15	6.3	25.4	51.2	106.4	132.6
YAFFS	0.00024	0.52	1.04	2.32	4.68	9.23	21.45	38.2
Content-based YAFFS	0.00027	0.21	0.32	0.62	1.82	3.02	5.12	7.43

Fig. 9 Average mounting time for JFFS2, YAFFS and content-based YAFFS using large multimedia files

Fig. 9 shows the average mounting time for JFFS, YAFFS and content-based YAFFS using large multimedia files

Empty file and various multimedia files of sizes with the range from 4Mbytes to 734 Mbytes are used. The file with size of 734Mbytes is widely used multimedia contents for 1 hour VOD service. The result shows that the average mounting time of content-based YAFFS is 94% and 80% less than those of JFFS and YAFFS, respectively. Besides, it shows that the enhancement ratio of the mounting time increases as the file size increases.

V. Conclusion

In this paper, we proposed content-based YAFFS to accelerate the mounting speed of the file system. Our experiments show that the performance of the proposed method is much enhanced when the large multimedia files are used. Therefore, we anticipate that the content-based YAFFS can be applied to the storage system for IPTV.

In the future work, we will apply the content-based YAFFS to real IPTV environments which uses network file stream service.

References

[1] F. Douglis, R. Caceres, F. Kaashoek, K. Li, B.Marsh, and J.A.Tauber, "Storage Alternatives for Mobile Computers, "In Proceedings of the 1st Symposium on Operating Systems Design and Implementation, pp.25-37, 1994.
[2] K.H Park, J.S Yang, J.H Chang and D-H. Kim, "Anticipatory I/O Management for Clustered Flash Translation Layer in NAND FlashMemory, "ETRI Journal Vol. 30 No. 6, 2008.
[3] Samsung Electronics Co., "NAND Flash Memory & SmartMedia." Data Book, 2002.
[4] D. Woodhouse, "JFFS: The Journaling Flash File System,"Technical Paper of RedHat inc. Oct. 2001.
[5] M. Rosenblum and J. K. Ousterhout, "The Design and Implementation of a Log-Structured File System, "ACM Transaction on Computer System, Vol 10, No. 1, pp.26-52, 1992.
[6] A. Kawaguchi, S. Nishioka, and H. Motoda, "A Flash-Memory Based File System, " In Proceedings of Usenix Technical Conference, New Orleans, Louisiana , pp.155-164, Jan. 1995.
[7] YAFFS Spec, http://www.aleph1.co.uk/yaffs/yaffs .html.
[8] Intel Coporation, "Understanding the Flash TranslationLayer (FTL) specification" 1997.
[9] H.J Kim, Y.J Won, "Mobile Multimedia file System for NAND Flash based Storage Device, " Consumer communications and Networking Conference, Volume 1, pp. 208-212, Jan.2006.
[10] S.H Kim, Y.K Cho, "The Design and Implementation of Flash
[11] Cryptographic File System Based on YAFFS, "Information Science and Security Conference, No. 10, pp.62-65, Jan. 2008.

Wonhee Cho was born in Ulsan, Korea in June 27, 1985. He received the BS degree in computer and information technology from the Myong-Ji University, Korea, in 2007, and the MS degree in electronic engineering from Inha University, Incheon, Korea, in 2009. His current research interests include intelligent algorithm in storage system, automotive software technology.

GeunHyung Lee was born in Incheon, Korea in July 7, 1983. He received the BS degree in computer engineering from the Inha University, Korea, in 2008, and the MS degree in electronic engineering from Inha University, Incheon, Korea, in 2010. His current research interests include intelligent algorithm in storage system, automotive software technology.

Deok-Hwan Kim is corresponding author. He received the BS degree in computer science and statistics from Seoul National University, Korea in 1987 and the MS and PhD degrees in computer engineering from Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 1995 and 2003, respectively. From March 1987 to Feb. 1997, he was with LG Electronics, as a senior engineer. From Jan. 2004 to Feb. 2005, he was with University of Arizona, Tucson, in a postdoctoral position to work on multimedia systems and embedded software. Currently, he is an associate professor in the School of Electronic Engineering at Inha University, Incheon, Korea. His research interests include embedded systems, intelligent storage systems, multimedia system, and data mining.