Design of the Journaling File System for Performance Enhancement

Seung-Ju, Jang

Dong-Eui University, Dept. of Computer Engineering

Summary
In this paper, I developed for the purpose of ensuring stability in the data processing of the file to save space. If the file is write-through compression techniques give the effect to reduce storage space. In the result, we can save storage space on small embedded systems to maximize the capacity is used. This paper was implemented using the ext3 file system of the Linux Operating System. The compression algorithm was implemented in ext3 file system. EXT3 journaling file system through selective compression algorithm prevents the performance degradation and uses the available storage space efficiently. In experiment, the time consuming for compression does not take so long. It shows that the selective compression algorithm is better on space-efficient vs. time-consuming.

Key words: Journaling File System, Selective Compression Algorithm, Performance Enhancement, Linux O.S.

1. Introduction

With the development of embedded systems the use of the Linux operating system is rapidly increasing. Linux operating system has been expanded to the use of smart system. Especially to ensure data security technology has become essential in portable devices.

The existing computer system uses fsck(1M) in order to ensure the stability of the file system. When you apply fsck(1M) for a large file system it makes problems. If the file system to a larger capacity, it takes a lot of time to perform fsck(1M) when the capacity of the file system is large.

This paper uses the ext3 file system as Linux journaling file system. The ext3 file system uses a module called jbd. The jbd is a universal library provides the journaling. The purpose of journaling is to keep the consistency of the meta-data. Thus, writing of the block will be conducted in the following order.

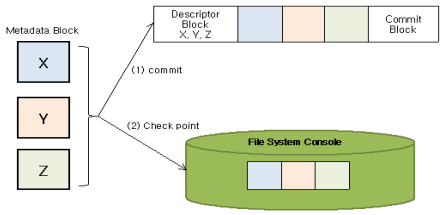

Fig. 1 Journaling File System Structure

(1) The block will be recorded in a journal area in the case of writing at the same time. At this point leading block is descriptor, commit block is placed at the end [4-7].

(2) After the end of treatment committed, write to the actual block. This process is called "check point processing". Write of every block to the ends is called "complete check point".

This paper is developed for journaling capabilities to ensure data reliability in embedded systems. In addition, the proposed function saves file handling space. It has been developed for this purpose. If the file is write-through, It reduces the storage space through compression techniques. This paper was implemented using the EXT3 file system of the Linux operating system. The compression algorithm was implemented in the EXT3 file system.

This paper is composed of followings. Section 2 is a related studies of this paper. Section 3 designs the selective compression algorithm and developing contents. Section 4 is an experimental result. And finally, Section 5 concludes this paper.

2. The Related Research

The previous studies associated with journaling file system are as follows. JFS2 is the first journaling file system used for many years in the IBM AIX operating system before being ported into Linux. Although based on the original JFS, the JFS2 is a 64-bit file system. It has improved to be more scalable and has been enhanced to support multi-processor architecture [1-3].

Manuscript received May 5, 2011
Manuscript revised May 20, 2011

JFS2 file system supports high-performance ordered journaling techniques with additional system recovery. JFS2 also provides extent-based file allocation to improve performance. Extent-based allocation is allocating consecutive block, rather than assigning a single block of contiguous blocks [7-10].

EXT3fs is the most popular journaling file system, and it is based on the EXT2 file system. EXT3fs is actually compatible with EXT2fs. EXT3fs uses the same structure of the EXT2fs and journaling was simply added. Moreover, the partition of the EXT3fs is even mounted as EXT2 file system and EXT2 file system can be converted to EXT3 file system [11-13].

The EXT3fs allows three types of journaling(writeback, ordered, data), but the ordered mode is the default mode. Journal commit policy can be set, but basically commit is executed when one of the commit timer is time-out or a 1/4 journal is full [14-17].

3. Design of the Selective Compression Algorithm

This paper uses the Linux EXT3 file system. A journal area of the EXT3 file system can be used inside area or some other partitions. If you use inside area in the file system, the 8 I-node number is used. I-node 8 is a file that uses the journal to the area. Journals zone exists as one file behind the scenes. It does not seem to have a directory entry. The user cannot see this I-node. Because journal area has the following structure in Fig. 2.

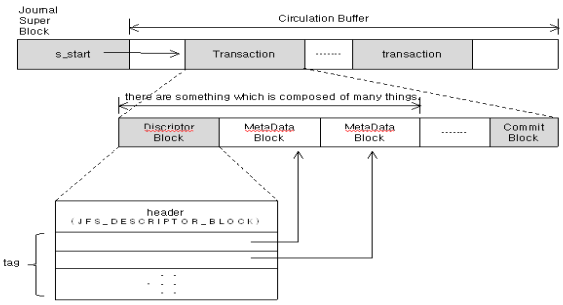

Fig. 2 Journal Area Structure

When the Journaling file systems write the files, they first check for the sequence. The reason for checking the sequence is that the large files with user's compression techniques show the sequential approach characteristics such as multimedia files. Sequential inspection technique is performed in a similar manner with reading the pre-existing Linux sequential test. When previous page of the file ready unit is to read the current file offset and the offset of the page adjacent to the unit is to determine the ordered sequence. And subsequently it increases the size of the window which refers to the set of contiguous pages. When the window size reaches to the threshold set, approach in future decides that the file has sequential order.

Compression filter checks the file with a compression ratio of sequential test. Compression test procedure compresses each page and calculates cumulative compression rate. When the window size approaches to the threshold and the cumulative compression ratio is higher than the threshold, the suggested algorithm decides that it is a user file. After that, write operation does not make compression procedure.

The selective compression algorithm of this paper associated with the journaling function is as follows.

- In the basic operation, the first part of the file is compressed and the compression ratio is checked.
- Compression rate of the compressed file decides whether compression is handled or not.
- If you use a compression algorithm, compress the journal information
- If you do not need to compress, the file should be saved in the traditional way without compression.

I apply compressed techniques to the file for the purpose of efficient space utilization and improvement of writing speed in the EXT3 file system. During the process of file compression, compression technique is applied to the uncompressed files, after identifying the compressed file and uncompressed files of the user level.

User level compression file types are such as mp3, mpeg, jpeg, pdf. The Fig. 3. is a complete system architecture for implementation system.

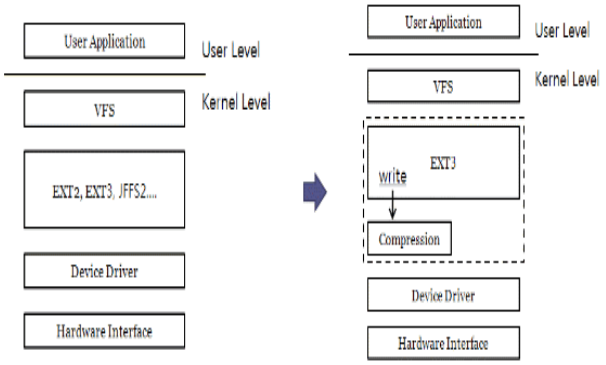

Fig.3 Proposed System Structure

This paper proposes a new compression techniques for a EXT3 journaling file system to improve the writing speed and to use storage space efficiently. Journaling file system actually requires two write operations for a single write operation, besides additional disk space is required.

During the process of file compression, compression technique is applied to the uncompressed files, after identifying the compressed file and uncompressed files of the user level. It can reduce the waste of processor cycles which comes from unnecessary duplication of files compression. When user-level re-compression of the compressed file, original size and compressed files are created in a similar size. There will be unnecessary waste of resources and increase of disk recording time because of the compressed files as large as original files in the middle of user level re-compression. The decision of whether or not the user file compression is performed in such a process. EXT3 journaling file system through selective compression algorithm prevents the performance degradation and uses the available storage space efficiently. The data files are compression selectively. The selective compression algorithm is as follows. Users files are separated compression or non-compression files. The proposed algorithm of this paper just applied for the non-compression files.

Fig. 4 shows the flow of the action of the selective compression algorithm which is proposed in this paper.

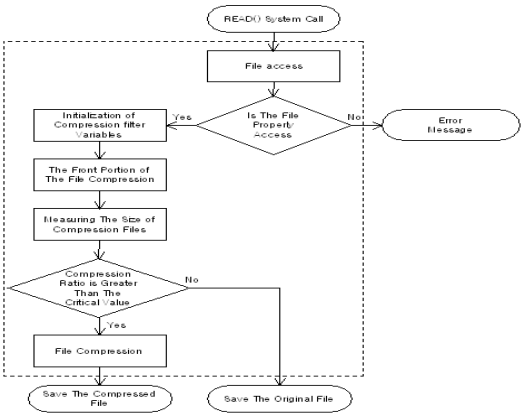

Fig. 4. Selective Compression Algorithm

Fig. 4, READ system call makes files access. If it doesn't access file normally, program exits and prints out error messages. If it accesses file normally, compression procedure including variable initialization for compression filter is performed. For the destination file, it compresses the front portion of file and measures the size of compressed files. Compression ratio for a predetermined threshold can be compared with the size. If the compression ratio of the files is higher than a predetermined compression threshold, it proceeds to compress files and save them. If the compression ratio of the files is lower than a predetermined compression threshold, the original file will be saved.

In this paper, Zlib compression algorithm is used. Zlib is a kind of data compression library written in C. Zlib is general purpose lossless compression algorithm(DEFLATE), so it has been widely used because it was not affected by patent. DEFLATE algorithm has been published as RFC 1951. Internally, DEFLATE algorithm is applied by the LZ77 compression algorithm, and Huffman coding. Zlib is proved to be reliable. Therefore, it guarantees decompression for compression files. When trying compressing them in the same way again, this algorithm is not be compressed any more. Zlib provides several functions in C, we can simply compress the files using those functions.

4. Experiments and Evaluation

This study carried out the experiments in the physical system. Table 1 is the experimental environment.

Table 1. Experimental Environment

HW Environment	SW Environment
CPU : Intel(R) Core(TM)2 Duo CPU 6400 @2.13GHz RAM : 2GB	Fedora Release 9/kernel Linux 2.6.33.6 file system : EXT3FS ordered mode(EXT3FS default-mode, zlib ver1.2.5

Table 1. shows the implementation system environments to test. The implementation O.S used in the physical system is Fedora Linux O.S. The experimental results are like followings. Therefore, I use the Linux EXT3 file system for the purpose of experimental environments. The proposed idea gives high compression rate for frequent using character strings.

For a high compression ratio file, the time to spent in data compression and time to record the compressed data to disk will be shorter than the time to record the original files on disk. On the other hand, user-level files and low compression files which create the original size compression files waste resources and increase disk recording time because of unnecessary file compression. Therefore, the technique of the selective compression algorithm is required for a low compression ratio files and high compression files.

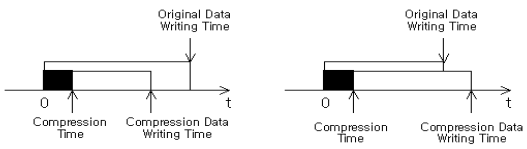

Fig. 5 Write Execution Time Comparison for High Compression Ratio File and Low Compression Files

Fig. 5. shows comparison of execution time for writing file according to compression ratio. If you save a user-level compression file without the selective compression algorithm, you may waste unnecessary resources and decrease writing speed due to redundant compression process.

During file compression, compression program is to determine compression or not by checking compression ratio of a certain portion of the file. By setting the threshold of compression ratio, if the file compression is lower than the threshold, the proposed algorithm determine it as a user-level file and does not compress the rest of file. If the compression ratio is higher than the threshold, the rest of file is compressed and written on the hard disk.

File compression is performed using the Zlib library gzib format. Zlib compression library is a freely available license unlimited. This library provides integrity checking function of compression and decompression functions in memory. Zlib is a proven and reliable. It ensures extracting compressed files normally. In the same way re-compressing does not work anymore.

In this paper, I used kernel version 2.6.33 based on Linux. Kernel 2.6.33 version is supplied by Fedora core 9, and supports EXT3 file system. When you check var/log message file after rebuilding the kernel which applies printk() function to ext3_journal_start() of /fs/ext3/inode.c, you can see that the journal function starts at ext3_write_begin() function. As a user program calls write() system call, write operation starts generic_file_buffered_write() function of /mm/filemap.c. Using Zlib library, generic_file_buffered_write() function is applied to the compression scheme.

If you encounter file system stability problems, the work of the file system is down, modified data can be corrupted or recovery may be impossible. The proposed enhancement of the EXT3 file system is maintained stability, because it performs journaling ordered mode of existing EXT3 file system.

Table 2. File Types used in the Experiment

	hwp	xls	ppt	doc	All Files
Number of test file	20	20	20	20	80
The number of files which are compressed by a selective compression algorithm	16	19	3	19	57
Existing file size (byte)	4152832	5604352	32083963	15469420	57310572
Size of file after adopting an selective compression algorithm	3795655	2775084	30801845	12915301	50287885

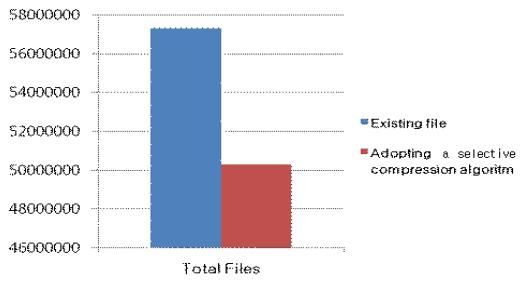

Fig. 6 Comparison of Overall Performance

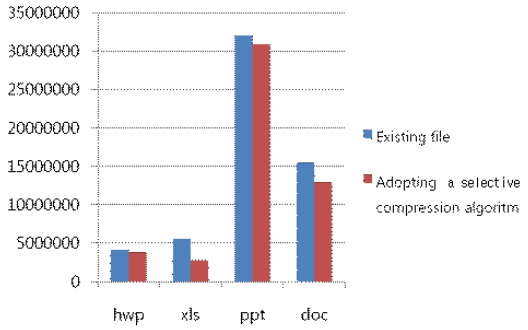

Fig. 7 Comparison of Individual File Performance for a Particular File

I experimented the proposed idea's implementation. Each of 20 randomly selected Hangul, Excel, and Power Point files, was tested. Temporary compressed files are the beginning 10240 bytes of each file and specifying a threshold of compression. The selective compression algorithm determines to store compression file or un-compression file by comparing between compression rate(threshold) and compression rate of temporary compression file's compression rate.

Compared to the original source file size is 100, the size of the compressed file was expressed as percentage. A lower percentage number will be compressed as much. The final compression efficiency ratio significantly dropped in the case of Hangul file. Meanwhile, the compression efficiency ratio in Excel file was much higher.

The efficiency ratio for the temporary compressed files of Hangul was good. However, the final compression ratio of Hangul was revealed bad. Therefore it seems to be better idea to specify the availability of this algorithm after compressing a little more by increasing the threshold.

In Word files and Power Point files, the compression efficiency is different according to each file. To determine compression, it was found out that the proposed algorithm is good by filtering. Overall, the time consuming for compression does not take so long. It shows that the selective compression algorithm is better on space-efficient vs. time-consuming.

5. Conclusions

This paper proposed to adapt the selective compression algorithm in Linux EXT3 journaling file system. This feature was developed to save file processing space as well as ensure data reliability in embedded systems. It gives the effect to reduce the storage space through the selective compression technique when you have a writings file.

By doing so, this algorithm saves storage space on small embedded systems and maximizes the storage capacity. This paper was implemented in the Linux Operating System using the EXT3 file system. We implemented the selective compression algorithm in the EXT3 file system. The proposed algorithm was implemented and tested. The files used on the test are ordinary files such as Power Point, Excel and MS-Word files. The result of the experiment shows that the highly compressed file in the enhancement EXT3 file system can save the storage space

Whether to compress or not should be previously determined for all the files. The performance of the file system may be degraded because compression is performed ever in a portion of the low compression file. However, the burden is negligible compared to the benefits of saving the storage space. The proposed selective compression algorithm could be confirmed to be superior to the general compression algorithm.

Acknowledgments

This work was supported by Dong-eui University Grant.(2010AA177)

References

[1] RedHat, “Red Hat’s New Journaling File System ext3”, 2001
[2] http://www.zlib.net/manual.html
[3] TAKAHASI HIROKAZU, “Linux Kernel 2.6 Structure and Principles”, Hanbitmedia, 2007
[4] Daniel P.Bovet, Marco Cesati, “Understanding the LINUX KERNEL”, O’Relly, January 2001.
[5] C.S. Han,, “Design of the Meta-data Journaling Mechanism to Enhancement and Stabability in FAT File System", The KIISS Journal, Vol.36, No. 3, 2009.
[6] RedHat, “Red Hat’s New Journaling File System : ext3”, 2001
[7] J.S.Suk, Log-ordered Mode Journaling for the EXT3 File System", KIISE 06 Conference A, 2006.
[8] S.W.Kim, "Design of the Meta-data Journaling Structure for Shared Disk File System", Korea Multimedia Conference, 2000.
[9] T.H.Kim, "Journaling File System for Enhancement Reliability in the NAND Flash Memory", KIISE 2006 Conference A, 2006.
[10] D.H.Yok, "The Journaling File System for NAND Flash Memory", Korea Foreign Language Univ. M.S thesis, 2009.
[11] Prabhakaran, V. Arpaci-Dusseau, A. C. Arpaci-Dusseau, R. H. “Analysis and Evolution of Journaling File Systems”, PROCEEDINGS OF THE GENERAL TRACK, 2005.
[12] Preslan, K. W. Barry, A. Brassow, J. Cattelan, R. Manthei, A. Nygaard, E. Van Oort, S. Teigland, D. Tilstra, M. O'Keefe, M. “Implementing Journaling in a Linux Shared Disk File System”, NASA CONFERENCE PUBLICATION, 2000.
[13] Daniel P.Bovet, Marco Cesati, “Understanding the LINUX KERNEL”, O’Relly, January 2001.
[14] H.J.Choi, "The Efficiency Log Control Mechanism for Log Reduction, Fragement Data Management of Journaling File System", KAIST PH.D thesis, 2008.
[15] E.Y.Jung, "Design and Implementation of Journaling File System for Mobile Computing Environment", Vol.8.No.1, 2001.
[16] H.C.Lee, "Journaling Mechanism and Indexing Structure of Nand Flash Memory File System for Mobile Multimedia Device", Technology Science Journal Vol.37-1, pp.41-49, 2007.
[17] T.H. Kim, "Journaling Recovery Mechanism for the RFFS", Busan Nat'l Univ. M.S thesis, 2007.

Seung-Ju, Jang received a B.Sc. degree in Computer Science and Statistics, and M.Sc. degree, and his Ph.D. in Computer Engineering, all from Busan National University, in 1985, 1991, and 1996, respectively. He is a member of IEEE and ACM. He has been an associate Professor in the Department of Computer Engineering at Dongeui University since 1996. He was a member of ETRI(Electronic and Telecommunication Research Institute) in Daejon, Korea, from 1987 to 1996, and developed the National Administration Multiprocessor Minicomputer during those years. His current research interests include fault-tolerant computing systems, distributed systems in the UNIX Operating Systems, multimedia operating systems, security system, and parallel algorithms.