An Autonomic Defragmentation File System

Hyun-Chan Park[*] Jun-Seok Lee[†] Chuck Yoo[*]

[*]Department of Computer Science and Engineering
Korea University, Seoul, Republic of Korea
E-mail: {hcpark, hxy}@os.korea.ac.kr

[†]Department of Computer Science and Engineering
Korea Army Academy at Yeongcheon, Gyeongbuk, Republic of Korea
E-mail: junseogi44@gmail.com

Extended Abstract

1. Introduction

The fragmentation is a traditional problem of file system. Currently, it is much more important problem because of that the difference between CPU and I/O performance was increased. When a fragmentation occurs, file system cannot maintain its own block allocation policy and block layout on disk. For these reasons, the fragmentation degrades the file system performance in all respects. The main problem of file system fragmentation is that it cannot recover itself. Therefore, defragmentation is required not to degrade the performance of file system.

Existing techniques for defragmentation have two approaches. First, we can re-write all the files of file system at some specific time. There are so many tools, such as Microsoft Defragmentation Tool, which use this method. It almost perfectly removes a fragmentation on disk. However, it requires too many I/O operations for a short period. Meantime, user cannot fully utilize his(or her) computer. Second, we can re-write small files when the over-write operation is required on those files[1]. This method needs no additional I/O operations. However, over-writes on small files are not frequent operation. If the fragmented file has a read-only property, it will remain as fragmented file forever.

2. An Autonomic Defragmentation with Lazy-Copy Technique

To solve above problems, we propose the autonomic defragmentation file system without a degradation of system performance. At first, we design the Automatic Layout Scoring(ALS) system which is used for measuring the fragmentation ratio of the files. The ALS counts the number of contiguous blocks in a file when the blocks are allocated for the file. We can recognize the fragmented files with ALS. After the detection, we search the free and contiguous blocks for defragment the file. If searching succeeds, we copy the file at idle time. To reduce the effect of additional I/O operations, we chase the idle time of the hard disk drive which contains the file. When the idle time is found, we copy a small portion of file. Because our goal is minimizing the effect of additional I/O, we divide the copying process into several pieces. The size of a piece is same with a maximum size of one I/O request for the block device, for example, 64KB. When the file copy is completed, we edit the block indexes in the I-node of target file to pointing new block indexes which are contiguous. We call this method as lazy-copy technique because copying is performed only at idle time of I/O device.

There are two issues for design and implementation of these methods. First, we decide new location of defragmented file with considering of a relationship with other files and logical hierarchy of files. If we move the file far away from other files in same directory, overall performance of file system will be degraded because our operation will break the spatial locality of related files. For a prototype, we search the nearest blocks with an original location of first data block for new location. We plan to measure the effect of various policies about the block allocation. Second, we must minimize a move of disk head. If we move the disk head for lazy copy, some other process that occasionally use the disk will have disadvantages from breaking its locality of files. To minimize the move of disk head, we design the lazy-copy system efficiently. All the lazy-copy requests are maintained in B+ tree with an index of disk block number. When an idle time has come, we can easily search the nearest request with current disk head position.

We implement the prototype of Lazy-Copy Defragmentation File System(LCD-FS) based on EXT2 file system of Linux 2.6.18. We evaluate the performance of LCD-FS by comparison with EXT2. We execute the

series of file operation workload. This workload is consists of 800,000 read/write operations which was executed on network file system which runs on a 502 Mbytes hard disk partition[2]. It uses 10% of a partition at first, and the usage is increased for 90% at mid-time. At last, 70% of the blocks in the partitions are allocated. We generate the fragmented files on two file systems, EXT2 and LCD-FS by 10 steps. At each step, we execute the workload several times or create and delete several big files. And we measure the layout score for each step. As a result, layout scores are increased by 2.4%~10.4% for first and last step. Furthermore, we measure the read/write performance of fragmented file system at first and last step. For various files(16 KB~32,768 KB), write performance is increased by 1%~8.5% and read performance is increased by 1.2%~7.5%. This result shows that the fragmentation degrades the file system performance and the defragmentation of LCD-FS successfully works. LCD-FS will maintain the performance of file system continuously. Figure 1 shows the results of experiments for read/write performance.

Another goal of LCD-FS is minimizing the effect of additional I/O requests. To evaluate this goal, we execute the same I/O operations on EXT2 and LCD-FS. We search the specific text pattern in Linux source tree using Grep utility. As a result, the elapsed times of two experiments are almost same. It shows that LCD-FS do not generate the I/O requests while another I/O-intensive process is running. After the execution of Grep, LCD-FS starts the defragmentation of file system.

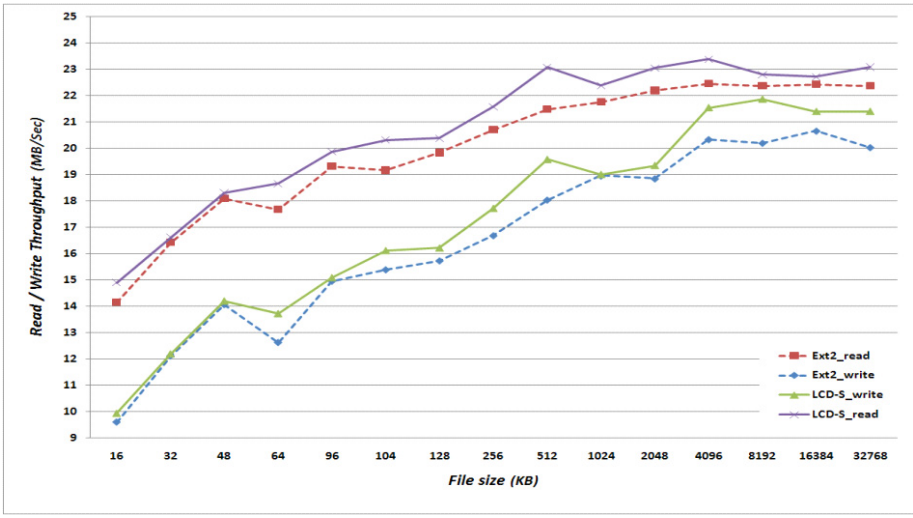

Figure 1. The performance of LCD-FS

3. Conclusion

Existing techniques for defragmentation of the file system need intensive disk operation for some periods at specific time such as disk defragmentation program. In this paper, for solving this problem, we design and implement the autonomic defragmentation file system without a degradation of system performance. We propose the Automatic Layout Scoring method for measuring fragmentation ratio of files and suggest the lazy-copy technique that copies the fragmented file at idle time. And we introduce two major issues of autonomic defragmentation system.

We implement these algorithms in Linux and evaluate them for small and defragmented file to get the layout scoring. We outperform the EXT2 file system by 2.4% ~ 10.4% in layout scoring evaluation. And the performance of read and write for various file size is better than the EXT2 by 1% ~ 8.5% for write performance and by 1.2% ~ 7.5% for read performance.

Keywords defragmentation , autonomic defragmentation, lazy-copy technique, autonomic layout scoring

References

[1] W.H. Ahn, et al., "DFS: a de-fragmented file system," Modeling, Analysis and Simulation of Computer and Telecommunications Systems, 2002. MASCOTS2002. Proceedings. 10th IEEE International Symposium on, 2002, pp.71-80.

[2] Keith A. Smith and Margo I. Seltzer, "File System Aging - Increasing the Relevance of File System Benchmarks," In Proceedings of the 1997 ACM SIGMETRICS Conference, pp.203-213, June 1997.