Performance Analysis of Multimedia File System

박진연[1*]　　원유집[1**]　　Jaideep Srivastava[2**]
[1]한 양 대 학 교　전자전기컴퓨터공학부　{jypark,yjwon}@ece.hanyang.ac.kr
[2]Dept. of Computer Science , University of Minnesota, Minneapolis, MN, USA

Jinyoun Park[1*]　　Youjip Won[1**]　　Jaideep Srivastava[2**]
[1]Div. Of Electrical and Computer Engineering, Hanyang University
{jypark,yjwon}@ece.hanyang.ac.kr
[2]Dept. of Computer Science , University of Minnesota, Minneapolis, MN, USA

Abstract

Intensive I/O bandwidth demand of the multimedia streaming service puts significant burden on file system. Different from the legacy text based or image data, the semantics of the data in multimedia format can be significantly affected if the data block is not delivered by the predefined deadline. The legacy file system used in Unix or Unix like environment is designed to efficiently handle the files who sizes range from few hundreds of byte to several tens of gigabytes. This fundamental design philosophy results in the file system based on multi level skewed tree structure. Multi level i-node structure has significant drawback when the application performs sequential read operation. In this article, we present the result of the performance study of the file system which is specifically designed for handling multimedia streams. We implemented the file system on Linux Operating System environment and examines the performance behavior of the file system under streaming I/O workload. The result of the study shows that the propsed file system performs much more efficiently than the ext2 file system of Linux does.

1. Introduction

1.1 Motivation

The efficiency of the underlying file system plays a critical role in providing the streaming service in cost effective manner. If it takes less time to retrieve frames or a certain amount of data blocks for a streaming session, the multimedia server can service more number of streams from the disk subsystem with smaller amount of synchronization buffer. Dominant factor which constitutes the disk I/O latency is seek overhead. Thus, in designing the multimedia file system, special care needs to be taken to minimize the seek overhead in data retrieval. The Unix(or Unix like) file system is designed towards handling wide range of file size. It adopts skewed tree like file structure where the node in the tree corresponds to data blocks in a file. With maximum depth of 3(triple indirect) in the tree, the Unix file system can handle the files whose sizes differ in the order of magnitudes. To transmit the individual frames of the video file to the end user, the server scans the video file sequentially and the rate at which the individual block is retrieved should be compliant with the playback rate of the file. When the server performs sequential scan of a file, the internal nodes of the tree are visited repeatedly in the course of navigating the leaf nodes(data blocks). Navigating through the multi-level tree structured file entails non-trivial amount of disk head movement overhead in visiting the internal nodes of the tree. Even though repeated visit to the internal nodes keeps these nodes in the buffer cache, visiting the internal nodes raises significant overhead in scanning operation. Due to this reason, legacy unix-like file system does not *support the streaming workload efficiently. In this work, we focus our* effort in proposing the new file system opted for streaming operation and analyzing its performance behavior under streaming workload.

1.2 Related Work

There have been a number of prototype file system which is designed specifically to handle multimedia data. IBM Tiger Shark file [H98] system offers feature such as resource reservation, admission control, deadline-based disk scheduling, large disk blocks, wide striping etc. Also, [BFD97] proposed Tiger video file server . All content files are striped across all of the computers and disks in a Tiger system.

2. File System Requirement for Multimedia Data Retrieval

The requirement for multimedia file system can be summarized into two constraints: (i) Individual stream needs to be supplied sufficient amount of data blocks for one round's playback from the disk in each round; (ii) Total time to retrieve one round playback worth's data blocks for all streams should be less than the length of a round. Constraints (i) is to prohibit each stream suffering from the starvation.

3. Synopsis : Unix File System

Typically in unix file system, the information required for management is kept strictly apart from the data and collected in a separate inode structure for each file. This structure is called "*i-node*". The inode stores the metadata information of each file and *has data references for actual data location.*

3.1 Structure of inode in unix file system

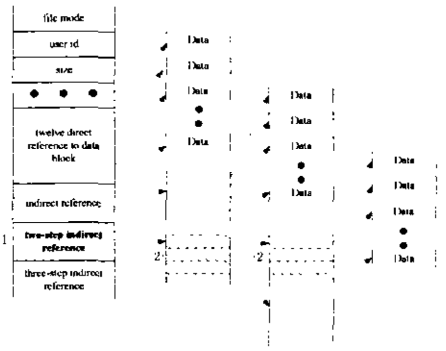
file mode
user id
size
twelve direct reference to data block
indirect reference
two-step indirect reference
three-step indirect reference
Data
1
2

Figure 1. Structure of ext2 file system

Ext2 inode structure contains the information of *file mode (e.g. rwxrw-r--)* , *user id of owner* , *file size* , etc for management and *data reference*. Data reference is composed of twelve direct references , an indirect reference, an two-step indirect reference and three-step indirect reference. Reference pointer size is 4 bytes.

If we assume that data block size is 4Kbyte in ext2 file system and direct reference is used, maximum size of stored data is 48Kbyte, i.e 12 direct references * 4Kbyte blocks. Single 4Kbyte block can hold 1024 block pointers. Thus with indirect reference, up to 4Mbyte can be stored. If with two-step indirect reference, up to 4Gbyte of data can be stored in a file. For the files larger than 48Kbyte , the retrieval of a block can cause retrieval of pointer blocks to locate the data blocks. While this tree structure based file organization gives greater flexibility in handling wide variety of file sizes, retrieval of pointer blocks gives substantial overhead in the streaming operation.

3.2 Characteristics of Multimedia file

Even with various compression method (e.g. MPEG-1/2/4 [MPEG1], [MPEG2],[MPEG4]), single one hour movie requires from hundreds of Mbytes up to several Gbytes capacity. Storing the file whose size ranges in the file organization from hundreds of Mbytes up to several Gbytes of file requires two-step indirect block or three-step indirect block. For example, to access the last data block of 1Gbytes file, three blocks one accessed in the course of seaching the data block. The three blocks consists of indo table block ,① in Figure 1, and two indirect blocks , two ② in Figure 1 . Ext2 file system for multimedia has following disadvantages.

- Increased I/O latency of startup access : Because one inode block and two indirect blocks needs to be retrieved prior to access the data block , non trivial amount of startup latency entails. Even though these blocks are in the buffer cache, memory access time consumes significant fraction of CPU cycle.
- Disk seek time overhead : Because it is very unlikely that pointer blocks and data blocks are stored consecutively , disk head need to travel to retrieve three blocks. Disk seek time is the major factor which constitutes major fraction of disk latency. The average disk seek time wastes 67% of the total time spent to access one 512 bytes sector in a random access pattern [R95] . This significantly reduces I/O performance of disk.
- Overhead of fragmentation : While the process that creates and deletes file , file fragmentation happens. Likewise, this fragmentation increases disk seek time.

Above disadvantages give motivation to make a new file system on Linux.

4. Designing Multimedia File System on Linux

4.1 Single Level Organization

To overcome the disadvantages of legacy Unix file system mentioned in Section 3, we propose simple but efficient file system designed for multimedia streaming purpose. Multi-level reference to data blocks which can cause file fragmentation is avoided and instead file organization is designed to be as simple as possible while reasonable degree of flexibility in supporting the various size files. Accordingly, simple file structure which does not use multi-level indirect reference reduces disk seek time. All data is placed on the disk in the unit of logical unit rather than the physical data unit, e.g. data blocks. Logical data unit can corresponds to a single frame of inter-frame coded video file or sequence of audio samples. By storing the data based on LDU, i.e. single frame or sequence of audio samples which takes up the consecutive location of the disk platter, it is possible to avoid accessing multiple data blocks scattered over the disk platter. This approach can greatly enhance the performance of retrieving the data blocks. When the video file is compressed with average playback rate with 9Mbits/sec under MPEG2 coding scheme, the size of I frame can be as large as 150Kbyte. Thus, the single I frame can consist of several tens of 4Kbyte data block in legacy unix file system, which can be scattered over the disk platter.

Multimedia file system consists of super block, inodes, index region, and data region. Data unit group corresponds to the notion of file in legacy unix file system. A data unit group can store video and audio samples of the video separately or can store the information in interleaved fashion. The size of data unit group is determined when the file system partition is formatted.

4.2 Superblock

As it currently stands, the file system does not have boot block. Superblock is located in the first block of the file system partition. It contains the information about (1) how many data unit group is in-use , (2) how much size of data unit group is used and (3) the number of free data unit groups. Data unit group is allocated free space based on first fit algorithm.

4.3 Inode

Since this file system has dedicated purpose of handling multimedia data, various redundant file system meta information is not maintained. This results in simple but yet elaborate inode . Inode maintains the information about (1) the name of data unit group, (2) id of data unit group, (3) number of LDUs, (4) creation time of data unit group and (5) encoding scheme.

4.4 Data Unix Group

Data unit group is partitioned into two regions: index region for logical data units and data region for logical data unit. Index region consists of array of index for referencing logical data units. Index for logical data unit contains the size and the location of the respective logical data unit.

5. Performance Evaluation

In this section, we examine the performance behavior of the proposed file system. The file system is implemented on the Linux operating system platform and the performance of the file system is measured via experiments under streaming workload.

The experiment is performed on Dell PowerEdge 2300 with dual Pentium III Xeon(Katmai) processor with 512 Kbyte cache. The box houses six Ultra-Wide SCSI hard disk each of whose capacity is 9.1 Gbyte . Six disks are configured in hardware RAID of level 5 and thus the operating system recognize the group of disks as single disk subsystem.

We first compared the performance of file system about streaming environment. This show that new file system on Linux is more efficient than ext2 file system. Next, we demonstrated that standard deviation about reading entire multimedia file in new file system is smaller than in ext2 file system. Which standard deviation is smaller show that jitter doesn't happen frequently.

5.1 Comparison of new file system and ext2 file system in streaming environment

This experiment is designed to determine the effect of concurrent multimedia streaming. It is important to ensure that files don't read from buffer cache. So, each stream must read each file only once. We created sixty MPEG-1 file of 30Mbytes in ext2 file system and in new file system on Linux. Then concurrent multimedia stream read each MPEG-1 file. We measured average, minimum, maximum read time of multimedia stream. And we repeated in the same way as changing the size of LDU

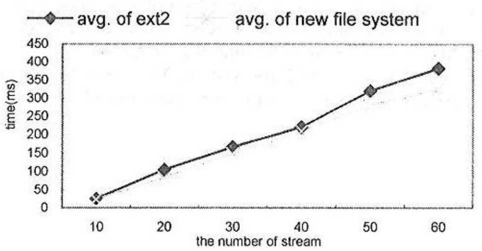

Figure 2. 32Kbytes LDU, average of stream read operation time.

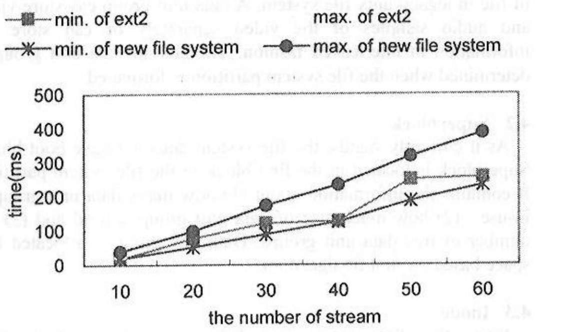

Figure 3. 32Kbytes LDU, min. and max. of stream read operation time.

In Figure 2,3, the average of stream read operation time is similar when ten streams are concurrently reading. But as the number of streams increase, new file system on Linux has smaller read operation time than ext2 file system.

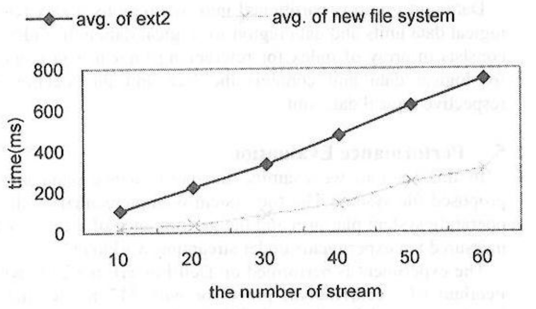

Figure 4. 64Kbytes LDU, average of stream read operation time.

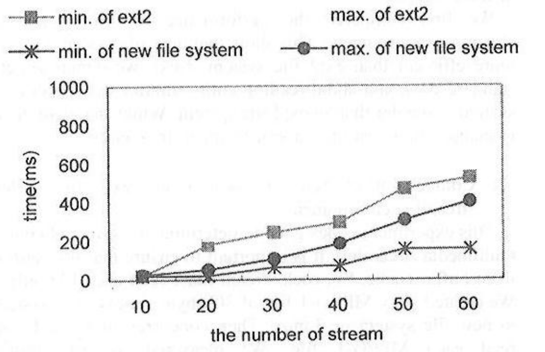

Figure 5. 64Kbytes LDU, min. and max. of stream read operation time

This is because new file system removed disk seek time overhead because of two-step indirect and more-step indirect reference and block groups. Maximum time of read time for stream on new file system is always smaller than on ext2 file system. This show that though many stream services, retrieval operation is performed stably.

5.2 Standard deviation about file read

This experiment is designed to measure read operation time when a stream read one MPEG-1 file. We repeated 30 times and measured average of read operation time. As read operation time and standard deviation about read operation time is smaller, jitter doesn't happen frequently

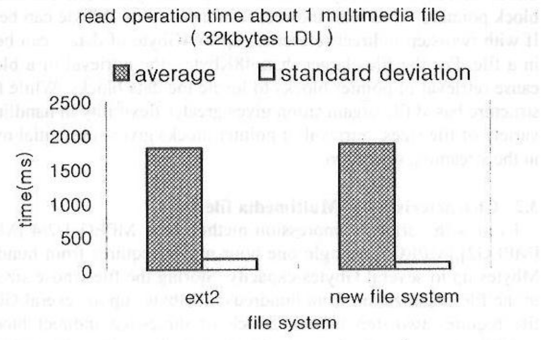

Figure 6. average and standard deviation of file retrieval time

In Figure 6, ext2 and new file system is similar about read operation time ,but standard deviation of ext2 file system is about 171 and standard deviation of new file system is about 112. The percentage improvement is about 34% . This show that new file system has better point about jitter

6. Conclusion

The main motivation of multimedia file system on Linux is that in legacy unix file system multimedia streaming is not suitable. we focus our effort in proposing the new file system opted for streaming operation and analyzing its performance behavior under streaming workload.

We fount that :

- Structure of Ext2 file system is not suitable multimedia streaming.
- New multimedia file system has less service cost / stream .
- New multimedia file system has better point about jitter.

We expect that with multimedia file system , multimedia stream server can increase the number of concurrent streams and reduce disk overhead and thus, improve multimedia stream server performance.

Reference

[BFD97] William J.Bolosky, Robert P.Fitzgerald , John R.Douceur "Distributed schedule management in the Tiger video fileserver" *Operating Systems Review (ACM)* 1997

[H98] Roger L.Haskin "Tiger Shark - a scalable file system for multimedia" . *IBM Journal of Research and Development* v 42 n 2 p185-197 , 1998

[R95] Rosenblum, M., "The Design and Implementation of a Log-Structured File System", *Kluwer Academic Publishers*, 1995.