ANALYSIS AND COMPARISON OF LINUX FILE SYSTEMS IN MULTIMEDIA ENVIRONMENTS

By

Peter Jahn

Jahn@LinuxCampus.net

A DISSERTATION

Submitted to

The University of Liverpool

in partial fulfilment of the requirements
for the degree of

MASTER OF SCIENCE

28/12/2010

ABSTRACT

ANALYSIS AND COMPARISON OF LINUX FILE SYSTEMS IN MULTIMEDIA ENVIRONMENTS

By

Peter Jahn

In recent years the usage of multimedia storage systems has dramatically increased and more and more vendors are selling Linux-based multimedia devices, such as satellite receivers/recorders, multimedia centers, music boxes, and other multimedia equipment. Since a current Linux kernel includes over 30 different file systems (including file systems for clusters and networks), the question is raised, what impact does a Linux file system and its characteristics have on the performance of a multimedia system?

Even though we can manipulate multimedia files like any other files, not every file system can cope with the different and complex requirements of all types of multimedia files. Each Linux distribution has its own preferred file system, but even if we could store all of our multimedia files on it, this would only be a stopgap. It is sometimes very hard to understand that every file system has its pros and cons, and that no single file system can fit every situation. All of the many file systems available were created based on experience with other file systems, and very often they were created to be the best for a special environment or situation. Storing and manipulating multimedia files is a very complex task, because multimedia libraries can contain anything from very small to quite large files. Also the amounts of files per partition can vary extremely.

This work provides an analysis to identify and evaluate existing Linux file systems and their behavior as multimedia file systems. Analyzing the requirements of multimedia files, such as different file sizes, typical file amount, and so on, was one part of this project. Also an up-to-date comparison of several Linux file systems available for multimedia files has been made.

Based on the outcome of this analysis, five different multimedia libraries were created and used for a series of experiments and performance tests in the lab. These experiments show how the default settings of file systems perform and which improvements can or should be made in multimedia environments.

DECLARATION

I hereby certify that this dissertation constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions, or writings of another.

I declare that the dissertation describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,

Peter Jahn

ACKNOWLEDGEMENTS

I would like to thank Taly Sharon, the dissertation advisor for her support, patience and constructive comments. I am also grateful to Martha McCormick, the general dissertation advisor for helping me to find a great dissertation advisor for my project. I would also like to thank my family for their understanding and great help during the last three years. And last but not least I would like to thank Joyce for being the most important friend during this study.

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

Page

Chapter 1. INTRODUCTION

Multimedia files are among the most commonly used files on computers today. Therefore storing and reading multimedia files from a hard disk are common tasks for every PC owner. Whether pictures, music, or movies, all of these are stored like other ordinary data in the file system. Since the common lifestyle of people today has changed to make use of more and more digital media devices, the need for digital storage space has also dramatically increased. Especially in multimedia environments one is are faced with a lot of new demanding requirements, such as a broad range of different file sizes, different amounts of files, real-time applications, and so on. No matter which kind of multimedia file one would like to handle, a common way to store such files is on general-purpose file systems. The problem is that although simply using the operating system's default file system could lead to a storage system where all multimedia files can be stored, such storage would only be a stopgap and a waste of resources.

1.1 Scope

The scope of this research project is to investigate the impact of multimedia files on a Linux file system and to make a deep comparative analysis of 5 popular journaling file systems currently available for multimedia files. The main research questions this dissertation is focused on are:

- What impact does a Linux file system and its characteristics have on the performance of a multimedia system?
- Which Linux file system fits better for which kind of multimedia files?

To be able to analyze and answer this question, first two objectives have to be defined:

- What is a multimedia system, what are its characteristics, and what is required of it?
- Where are the differences among file systems and what are their characteristics?

The research methods used to obtain the information were a combination of case studies and experiments. The analysis of the characteristics of multimedia files is based on a short theoretical research summary of pictures, music, movies and their properties. Several practical experiments with real libraries containing different multimedia content were performed to examine their behavior. The next objective was a deep theoretical analysis of case studies and documentation about Linux file systems and their possible impact on multimedia environments. Additionally an exploratory study of Linux file system features and tuning parameters was needed, because it was not clear which of these could have positive or negative impacts on multimedia files. The last part of this research was again based on experiments to test if some of the tuning parameters found in the previous part could be used to improve the performance of multimedia files on Linux file systems.

1.2 Problem Statement

Today there are several operating systems available and each of them offers several different file systems. Especially in multimedia environments the use of Linux operation system is growing very fast because it is open source and therefore cost-free to use. Another reason for Linux as the preferred multimedia platform is the broad offering of different file systems and the availability of the source code.

When one talks about multimedia content and its problems, one often hears that read performance is important to guarantee that the needed data is there within a useful time frame. However, even though this is true, there is much more to consider. The complete file handling, such as creating, deleting, moving, and reading files, is also important, and these are tasks where each file system not only has it pros and cons, but also each has its limits. Additionally each file system offers several parameters to manipulate the standard behavior, which again have a huge impact on the performance and the limits of the file system. The question is, how one can say that file system X fits best for multimedia files when praxis shows that there is a file size range from 20 KB files for small pictures, 4 MB files for MP3 files and large files, up to 50 GB for movies in HD quality. The amount of files per partition also varies extremely, because on a 600 GB partition one can store ~52 50GB files but ~600.000 1MB files. Managing such different file sizes and amounts of files

produces different requirements on the file system and therefore it is not believable that one file system can fit perfectly for all kinds of multimedia files.

Even though there are several Linux file systems available it is common practice to use the default file system of the Linux distribution. Not only is it more convenient to set up the system this way, but also very often Linux distributors try to convince customers that the default file system is the best choice for most environments. Going a step further, most of the Linux distributors do not use the unmodified kernel source to build their kernel (also called vanilla or mainline kernel). Instead they make a lot of modifications to the original source code to activate or implement missing kernel features. These modifications are created mainly by implementing kernel code patches before the kernel gets compiled (Love 2005). These patches lead to the problem that benchmarks of one file system cannot be accurately compared with the same file system on a different distribution. Even if the major and minor number of the kernel version were exactly the same, it would not mean that exactly the same file system source code was used to compile this kernel.
Another example is the I/O scheduler which has an important impact on the balance between performance and stability. Eckermann and Tobey (2010) mention in their article about server performance that the default I/O scheduler in the main stream kernel is the anticipatory scheduler, but they also explain that SUSE systems such as SUSE Linux Enterprise Server 11 use the completely different fair queuing scheduler. Using a different scheduler entirely changes the way data is ordered and written to the I/O device, and according to Moallem (2008) each I/O scheduler is optimized for a different kind of workload. So only changing the I/O scheduler on a multimedia system will change the performance seen on read and write requests.

These are reasons why all tests in this project were created on an Open SUSE 11.3 Linux system with the latest stable vanilla Linux kernel version 2.6.36 installed, which can be downloaded from http://www.kernel.org. This change to the mainline Kernel has the benefit of increasing the reproducibility of all tests but also has the drawback that a file sys-

tem such as Reiser4 could not be tested, because it is not mainstream and can only be implemented via kernel patches.

1.3 Approach

The main research methods used to obtain the needed information for this project were based on experiments. On a Linux System, there are many different file systems available, and it would be beyond the scope of this project to cover all available file systems. Therefore this research project is restricted to five of the newest and most common local Linux journaling file systems. The selection criteria used to decide which Linux file systems should be analyzed were: the file system must be a journaling file system, it should be the latest available version, it should be marked as "stable", it must be freely available in the Linux kernel without license restrictions or extra need of kernel patching, and it should be commonly used and accepted in the Linux community.

Based on these criteria, the project analyzes the following file system types: ext4, JFS, XFS, Reiser3 and Btrfs. Btrfs and Reiser3 are included in this research project for special reasons. Btrfs is marked as experimental in the Linux kernel 2.6.36 with the note that the disk format is unstable and not yet finalized. But according to a statement of the MeeGo developer Arjan van de Ven, the Btrfs file system is "the future of Linux file systems" and much more stable than most people believe, and therefore MeeGo, the new mobile Linux operation system, uses Btrfs as its default file-system (Ven 2010). Also Linux distribution developer manager J.R. Scott is thinking out loud about replacing the default file system ext4 with Btrfs in the next Ubuntu version (Scott 2010).

Reiser3, on the other hand, is not the latest Reiser version, because its successor, Reiser4, has been available since 2004. It is not that the successor was not a good file system, but for several reasons Reiser4 is still not yet available in the mainline Linux kernel, and therefore it will be mostly ignored in this project.

To gain a feeling of real and practical requirements of multimedia files, four libraries have been created with real multimedia content as a part of this project. Each library is approximately 600 GB and has been filled with one kind of multimedia or multimedia element

content. Depending on the kind of multimedia content stored, such as pictures, sound, small videos or large videos, the average file size varies extremely and therefore also the amount of files per library. In addition to dummy sample files created via Linux commands such as "dd" or "tar", four libraries were used as source files for a series of experiments and performance tests in the lab. The outcome of these experiments were used as an additional source for the exploratory study of Linux file system features and their positive and negative impacts on multimedia files. After this part, a list of parameters and file system features, such as block size, journaling settings, mount options, and structure options, which could lead to better test results, was available. The outcome of all tests and the exploratory study was used to repeat several experiments with optimized settings. A conclusive comparison at the end of the project shows the performance effects of using standard file system parameters compared to those when using tuned settings in multimedia environments.

1.4 Outcome

More and more multimedia devices, such as music boxes, satellite recorders and multimedia centers, are based on a Linux operating system. The outcome of this dissertation project is an up-to-date comparison of available Linux file systems which can be used for multimedia files. In detail this document contains the following information:

- Introduction to multimedia files and file systems
- Impact and requirements of multimedia files on file systems
- Structure and types of Linux file systems
- Analysis of available up-to-date Linux file systems
- Linux file system features and tuning parameters
- Practical tests of several Linux file systems and their features in multimedia environments
- Results and summary of research
- Results and summary of practical tests
- Recommendations

Chapter 2. BACKGROUND AND REVIEW OF LITERATURE

One of the fundamental requirements of multimedia files is that they be stored and read from a storage medium. Even though a normal user does not really think about the file system requirements needed for manipulating multimedia files, the industry started a long time ago to prepare for the future of multimedia storage technologies. This chapter shows that it has been predicted for several years that the requirements of multimedia applications will create extreme demands on file systems, and it also shows where general-purpose file systems can and cannot satisfactorily fulfil the requirements.

2.1 Related Work and Literature

A paper published by Gartner entitled "File System Innovations Growing in Importance" describes in detail that older versions of file systems cannot cope with the scalability which is needed for the characteristic workload of the future. Gartner's recommendation is that vendors of storage and servers should invest more in the development of file systems and techniques to prepare them for the expected growth in the near future (Gartner 2007).

The requirements on future storage systems which will be used for multimedia applications was described in detail in the survey by Halvorsen et al. (2003). This team came to the conclusion that future applications will be a mixture of time-dependent and time-independent data types, and that storage systems available today do not fulfill all the requirements to support continuous multimedia applications and systems (Halvorsen et al. 2003).

To design a multimedia storage server which performs well, one must consider several fundamental issues. In their tutorial about multimedia storage servers, Gemmell et al. (1996) present a deep overview of the architectures and algorithms needed for implementing digital multimedia storage servers.

The dissertation by Niranjan with the title "File System Support for Multimedia Applications" deals with the design, implementation, and a performance evaluation of a special multimedia file system called MMFS (Niranjan 1996). The author performed several experiments to show the benefits of using a special file system for multimedia files compared to using a general-purpose file system.

Park et al. (2000) also presented a paper that shows the benefits of a special multimedia file system called PMFS. In their research they designed and tested this parallel multimedia file system in comparison with the parallel virtual file system (PVFS) and came to the result that PMFS provides better performance in the processing of real time data.

In a paper entitled "Quality of Service Routing for Supporting Multimedia Applications" written by Wang and Crowcroft and published on the IEEE journal *Selected Areas in Communication*, the importance of quality of service for multimedia applications was examined.

The authors are convinced that in the last few years there have been a lot of technical improvements in multimedia environment, but that the requirements for the element quality of service routing for multimedia traffic are still missing (Wang & Crowcroft 1996).

Often multimedia content is streamed over the wire. In connection with a real-time application the control of the available bandwidth is an important factor. Busse et al. (1995) have performed several experiments on the LAN and over the Internet to test and tune mechanisms to dynamically control the bandwidth of multimedia applications such as a video conferencing system.

To know which file system is better for which kind of multimedia files, it is necessary to find out what the requirements of multimedia files are. Charts by Fulton Wayne (2010) and B&H (2010) provide some useful data about how large image files compare in image size and resolution. More general Information about all kinds of multimedia files can be found in the book *Digital Multimedia* by Chapman and Chapman (2009) and in the book *Modern Operating Systems* by Tanenbaum (2008).

The impact of multimedia files on operating systems is also explained very well in the *Operating System Concepts* book by Galvin (2005). The project also requires a lot of information about the growth of storage media and file systems. A very good history of the growth of hard disks is provided in the article by Klein (2008). The importance of growth in file systems and storage area networks is explained in Liao (2003) and Gartner (2007). The size limits in Linux file systems are compared in SuSE (2005).

The file system chosen for storing multimedia files has a huge impact on the end result. Performance tests by SUN (2004) and Stephan (2005) demonstrate that no Linux file system can behave and perform equally for all kinds of files. Even if one knows exactly which file system should be used for a special multimedia environment, there is still the problem that each file system has its own features and tuning parameters.

File system performance can decrease over time, and one reason for this is fragmentation. The master thesis by Loizides (2001) analyzes the impact of fragmentation on ReiserFS, and articles such as "ext4 online defragmentation" by Sato (2007) describe the negative impact of fragmentation. The chosen method of disk space allocation within the file system can also have a huge suboptimal impact. In his dissertation Kang (2007) describes allocation and space management strategies and their impact on the system. A different view of the allocation strategy problems can be seen in the dissertation by Leung (2009) where the main focus was on indexing and searching in large-scale file systems.

Several books and documents such as Carrier (2005), Leung (2009), and Loizides (2001) give a deep explanation of file system internals and help one to better understand the internal structure of Linux file systems. To understand a particular file system with all its functions, it is necessary to analyze its special capabilities. Articles such as "JFS Log" by Best (2000), "ext4 Features" by Mathur et al. (2007), *"File System Architecture and Terms"* by French (2008), and many more provide them.

Improving and analyzing the performance of a Linux system is a difficult task. Each file system operation depends on many factors and is influenced by many parameters. Books such as *Performance Tuning for Linux Servers* by Johnson et al. (2005) deliver a lot of

background information on the subject. The dissertation about the effect of Linux I/O schedulers on performance by Moallem (2008) and the tuning document by Eckermann and Tobey (2010) also prove that little changes on some parameters can have a huge impact on the overall performance.

2.2 Multimedia Terms and Definitions

Multimedia is a commonly used term today and therefore it is surprising that every book defines it slightly differently. For this reason this chapter will first clarify several terms used in this paper.

The term multimedia is a combination of the terms multi and media. The term multi means that something consists of various elements, and the term media refers to the software and hardware used for presenting or delivering these elements. According to Chapman and Chapman (2009, p.7) the various elements of multimedia are "text, still images, sound, video and animation". So, broadly spoken, multimedia is the presentation of something which is based on the combination of at least two of the five basic elements: text, still images, sound, video and animation. A simple example of multimedia is a web page which includes just text and still images. An example of more complex multimedia is a movie which includes sound, video, and sometimes text and animations. It is not necessary for this document to distinguish between the general term multimedia and the term multimedia elements. Therefore in this paper the term multimedia is used for both of these concepts.

The software used to present multimedia is usually called a multimedia application. The multimedia application for viewing a web page would be the browser, for listening to music a music player, and for watching videos a video player.

All types of multimedia content must be stored somewhere. Today, there is a rich variety of different storage areas available, such as the local hard disk of a computer, an externally connected storage device such as a USB stick, an optical medium such as a CD or a DVD, or a storage server connected over the wire. In this document a multimedia stor-

age system is a medium where data can be saved and read again afterwards, even if there were a power interruption. Some documents about multimedia, such as Halvorsen et al. (2003), use the term multimedia storage system slightly differently, because they include not only the storage device but also the software system which handles all the necessary tasks, such as I/O scheduling, data placement and so on. In this document such a combination is called a multimedia storage server.

Multimedia content is normally stored on a storage system just like other data, such as spreadsheets and word documents. Generally multimedia formats are very complex in comparison with other formats, because they contain a variety of different elements (Murray & Vanryper 1996, ch10_02). Over time, a lot of different formats have been introduced to improve the quality of the transmission time or to increase space savings. It is beyond the scope of this study to cover all of these formats in detail, since there are over 30 different multimedia formats listed on pages such as http://multimedia.cx/formats.html, for example, and each of these has different uses and characteristics.

2.3 Multimedia Characteristics and Requirements

Storing and manipulating multimedia objects on the hard disk is a very complex process and demands a lot from a file system. The main problem is that manipulating multimedia content on a storage device produces different requirements than the requirement for regular files.

Even though multimedia files are often stored like any other file on the same file systems regular files are stored, this is a problem, because especially older file systems were primarily invented for regular files and not for multimedia files. This is the reason why special file systems for multimedia files such as MMFS and PMFS were invented.

Even though experiments such as those by Niranjan (1996) and Park et al. (2000) have proved that special file systems such as MMFS and PMFS perform much better in multimedia environments, the situation remains that the common way for users to store multimedia files is on regular file systems which are included with their operating system. Of

course large and professional multimedia storage servers used for streaming video such as YouTube behave differently, but normal users do not have this equipment.

What are the complex requirements of multimedia files on general-purpose file systems?

2.3.1 File Size

The first problem is the varying file size of the different multimedia formats. Because of the wide variety of media combinations, the files can be from very small to quite large.

Digitized sound is stored in raw or compressed format as an audio file on the storage medium (Chapman & Chapman 2009). Depending on factors such as the length, sampling rate, and compression rate an audio file can be from a few kilobytes to several megabytes. An audio format which is perhaps used the most is MP3, and therefore it is not surprising that nearly every media player supports it.

Multimedia applications, such as iTunes by Apple, offer the functionality to convert a music CD to different formats such as AAC or MP3. Depending on the quality needed for MP3 files, a data rate of 16 kbit/s up to 320 kbit/s can be chosen, whereby the default setting in iTunes 10.1 is 160 kbit/s. The sound library created for this project was 600 MB and included 100,621 MP3 and AAC files from several iTunes libraries. Each file has a different size, but the average size of a file was 6.1 MB. The size of a picture depends on resolution factors, color depth, and compression, and, here again, one could have files from a few kilobytes to several megabytes. Even cheapest entry-level cameras are at least 5 mega-pixel digital cameras which produce pictures with a resolution of 2560 x 1920 (B&H 2010). Multiplied by 24 bits of color per pixel, such a resolution produces files with about 14 MB in raw format. Compressed using an algorithm such as JPEG, the files can be reduced to a file size of from 0.5 to 7 MB each, depending on the quality needed. The picture library created for this project was 600 MB and included 1,345,102 files, where the average file size of library was 0.46 MB.

Videos are mostly a combination of audio and moving images, and therefore it should be clear that their file sizes are much larger than the size of a single picture or of audio files.

Also the variety of different video files is broad, because one could have anything from a very simple video without audio up to a high quality video with a large resolution and perfect sound. As can be seen, the size depends on many factors, but, for example, a 100-minute long encoded MPEG-1 video file requires approximately 1.125 GB, whereas the same video for high-definition television (HDTV) requires approximately 15 GB of space on the hard disk (Galvin 2005, p.717). Worse, an uncompressed HDTV movie with a length of 2 hours fills a single 570 GB file (Tanenbaum 2008 p.470).

For this project two 600 GB video libraries were created, one with 11,617 video files with an average size of 52 MB and one with videos in HD quality, which contains 35 files with an average size of 17.5 GB per file.

2.3.2 File Amount

In the past it was common for a user to store text files, spreadsheets, and some pictures on a computer, which could lead to several hundred files. Nowadays it is increasingly common to store downloaded music, pictures, books, and movies electronically on the computer. Users have also started to transfer all their music CDs, pictures, and movies to their computer to save them electronically or share them on the Internet. This transfer saves space in their real book shelves and music storage cabinets, but it leads to several thousands of multimedia files and gigabytes up to terabytes of needed storage space.

Users no longer visually see the space needed in their storage systems, which also leads to the problem that everything is stored, even if it is useless. An example of this are pictures taken on a holiday trip. In the past it was common to use 24 or 36 picture cartridges in a camera. After the pictures were taken, these cartridges had to be taken afterwards to a shop for development. Due to the fact that the development was charged per picture, the worst pictures were usually rejected to reduce costs.

Today it is more common to use a digital camera, where the cost factor has changed completely. Instead of buying several cartridges, a user buys a memory disk where he or she can save several hundred pictures. Instead of bringing the complete memory disk to

a shop for development, the consumer usually stores all the pictures on a computer at home without making selections. Afterwards users look at their photos on their computer or upload some of them to websites. This change in behavior leads also to the problem that on a one-week trip users take several hundred pictures with their digital cameras, because it costs no more than taking 24 pictures. Also the selection process is not so important anymore, because the user has the feeling that storing a few more pictures does not hurt, because there are no immediate costs. But storing two hundred pictures with an estimated size of 2 MB leads to 400MB of needed storage space, and after a few trips or years, several terabytes are needed to store the picture collection.

Handling such an amount of files is not really a problem for current file systems, but it produces some negative consequences, such as longer file system checks, slower search results, longer file handling times, more space needed for metadata, backup problems, and much more. Of course with a tuned data placement policy (block size, block allocation strategies, defragmentation techniques, striping, and so on) the efficiency of the storage system can be improved (Park et al. 2000), but first one needs to know which settings would be helpful.

2.3.3 Media Delivery and Data Rate

The next challenge is that multimedia applications require more and more services from the operation system that are normally seen only in real-time environments. Chapman and Chapman (2009) mention that multimedia applications can be divided into time-based and static media. Static media, such as text files or pictures, are not time sensitive, which means it doesn't matter if the file opens in one or two seconds. Also, when the file is loaded into memory there are no continuous changes anymore, and therefore the file produces no heavy load on the system. In contrast is time-based multimedia content, such as videos and sound. It may not matter whether a file is opened in one or in two seconds, but when it has started within the multimedia application, it is not desirable for breaks to happen because the content cannot be delivered fast enough from the storage systems. So time-based media requires not only continuous transmission, they also re-

quire transmission at a fixed time rate. As a result of these requirements, multimedia applications are also called soft real-time applications (Halvorsen et al. 2003).

As an example, consider a single video stream which is displayed at a resolution of 800 x 600 and a color depth of 24 bits. It would take 800 x 600 x 24 = 11,520,000 bits of data for each frame and, at 30 frames per second, 345 Mbps to present it (Galvin 2005, p.717). During playback, the system must read the data from the hard disk at a speed which is fast enough to avoid breaks or delays which would be disturbing in the presentation. On the other hand, while a video is being recorded, the system must continuously store the recorded data on disk, and a buffer overrun would lead to loss of data. So ensuring continuous recording and retrieval of multimedia content at very high data rates are important requirements for every multimedia system.

Based on the video example above, one could easily come to the conclusion that it would be better to tune a system for writing speed to avoid problems in recording, but in reality the answer is not that easy. It depends extremely on the environment, but according to Halvorsen et al. (2003) most of the time "data is written once" and afterwards it is "read many times sequentially". While, on the one hand, it is important for a write process to finish as fast as possible no matter where the data is actually stored on the hard disk, a read process, on the other hand, relies on techniques such as well-implemented data placement policies. These are some reasons why for a long time the storage system was seen as the main bottleneck in high data rate, multimedia environments (Halvorsen et al. 2003).

2.3.4 Streaming

One extreme requirement on multimedia libraries is the possibility of streaming. More and more multimedia solutions are storing media content on a central media server where several clients can connect to download the content. Generally media servers offer two different streaming techniques, called real-time streaming and progressive download.

Progressive download means a client downloads the complete content to the hard disk before sending it to the multimedia application. Real-time streaming means the client sends the content immediately to the multimedia application without storing it on the hard disk (Galvin 2005 p.716). While both techniques have their pros and cons for the client, real-time streaming is the worse choice for the media server. If a client uses the progressive download method, it makes no real difference whether the download needs 5 minutes or 10, because the client starts the video only if the download has finished. On the other hand the real-time streaming variant produces the drawback that the media server must deliver the content in a special time frame to avoid breaks. Also, playing forwards and backwards and watching the video several times do not put weight on our media server if the client has downloaded the content to the hard disk. The data rate and the amount of data produced by several multimedia sources can be seen in the following Table 1.

Source	Mbps	GB/hr
MP3 music	0.14	0.06
Audio CD	1.40	0.62
MPEG-2 movie (640x480)	4	1.76
Digital camcorder (720x480)	25	11
Uncompressed TV (640x480)	221	97
Uncompressed HDTV (1280x720)	648	288

Table 1: Data Rate of Multimedia Sources, Tanenbaum (2008 p.471)

2.4 Multimedia Storage System

A multimedia storage system has to deal with many issues, and since on-demand streaming has become very popular, the requirements on a system have also multiplied. Even though the computer speed and memory of a personal computer is fast enough to load several multimedia applications, the possibly high number of concurrent users accessing the server at the same time is a generic problem (Gemmell et al. 1996).

In spite of quad-core processors and several gigabytes of memory, it is still very common today to have to wait for seconds or sometimes minutes to get multimedia data. One of the main reasons for this problem is the hard disk. While CPU performance has increased 16,800 times from 1988 to 2008, the hard disk has had a minimal performance increase of only 11 times (Klein 2008, p.13). Even though producers have managed to dramatically increase the capacity of hard disks and their sequential transfer rates, the average access time of most current hard disk models is not very much better than it was a few years ago.

File system programmers are also aware of this situation, and therefore they spend a lot of effort to improve their file system features. For this and other reasons, it is important to consider what is the best file system and the best file system settings for a particular multimedia storage environment, because this is the easiest way to increase general system performance and stability.

2.5 File Systems Overview

One great benefit Linux systems have compared to other operating systems is the flexible choice of file systems and their parameters. A system can easily be created in which pictures are stored on one partition, music files on a other partition, and videos in high-definition quality on a third partition. Each of these partitions can be formatted with a different file system, or with the same file system but with completely different tuning settings.

One main goal of the practical tests in this research project was to find out which file system and which parameters fit best for multimedia content. It is heard very often that modern Linux file systems are already tuned and that the performance difference between them is marginal and can be ignored. An example that this is not true and that there is a real need for the outcome of this project can be seen in a test result printed in the German *Linux Magazin* edition 09/10 p.40-41. In this test two 32 GB partitions were created, and one was formatted with Btrfs and the other with ext4. Afterwards the team created 99,999 very small files on each partition. The real payload of these files together was less

than 1 MB total, but Btrfs needed 53 MB on the partition for the payload and the corresponding metadata. This is a huge overhead and waste of space, but it is necessary from the internal viewpoint of the file system. The result of ext4 is even more frightening, because it needed 621 MB of space for the same files. The time needed to complete the task was also different. Whereas the Btrfs kernel module needed approximately 30 seconds in the kernel space, ext4 needed 9 minutes longer! This single test result shows that the same content can produce completely different results on current file systems, but again this doesn't mean that Btrfs is the best file system and that there is no need for other file systems. It only means that in exactly this environment Btrfs was the better choice.

Another interesting pair of aspects is the partition and file size. Where hard disk vendors have succeeded has been in increasing the hard disk capacity. Today, common 3.5" hard disks are available up to 2 TB in size (3 TB is rarely available), allowing the consumer to create a single 2 TB partition on a single hard disk. By combining this with several other hard disks of the same size within a RAID array, the consumer can create very large file systems of several TB. This is again is a challenge to Linux file systems because each of them has its own limitations. For example, ext3 supports a maximum file size of 2 TB and a maximum file system size (partition) of 8 TB, whereas Reiser 3.6 supports a maximum file size of 8 TB and a maximum file system size (partition) of 16 TB (Stephan 2005). Also the amount of files per hard disk or directory cannot be ignored, as will be discussed next.

Every Linux file system has its own metadata area in which it keeps information in its own structured way about the data stored on the partition. Within this metadata there is an area which is used to store inode information, which is again used to store information such as file name, owner, time stamps, block location, and so on about files and directories. Each file and each directory stored on a hard disk needs exactly one inode, no matter how small or large a file is. Depending on the file system used, inodes are allocated either statically or dynamically. If a file system with static allocation is being used, the decision must be made how many inodes should be created on this partition when it is formatted. Changing the inode table size afterwards is not possible without reformatting

the partition, and therefore several file systems offer dynamic allocation. Using dynamic allocation has the benefit that when the partition is formatted, a fixed area for the inodes is created, and if the system needs more inodes, an additional area is allocated to smoothly increase the inode area (Sweeney et al. 1996 p.5).

One possible approach to overcome the inode problem would be to create many more inodes on the hard disk from the beginning than theoretically needed. Whereas in ext2 the default inode count was in some cases too low, the developers have set the default number much higher in ext3/4, but this leads to the next problem. When a computer is shutdown without the partitions being correctly dismounted beforehand, perhaps because of a power outage or hardware fault, for example, a so-called "unclean" and possibly erroneous file system is the result. At the next boot Linux will check if the partition is marked as unclean, and if this is the case, the kernel will start a file system check (fsck) on this partition. Now it depends on the file system used what exactly happens next, but in any case the fsck will try to repair the file system to gain a "clean" state again, and during this time the file system is unavailable. The time needed to repair the partition depends on many facts, two of which are journaling and inodes. Whereas in a non-journaling file system the kernel has to examine all of the file system's metadata, in a journaling file system only file transactions which are not finished in the journal need to be checked. There are also several different implementations of journaling levels and settings available and some of them were analyzed and tested within this project.

The next impact on the file system check time is the amount of inodes. Normally an fsck is a very slow operation because every inode in the file system must be checked. Interesting to know is that the time needed to repair an ext2/3 file system depends on the amount of inodes created, and the file system check needs the same amount of time regardless whether the partition is filled with files or empty (Mathur et al. 2007). This is one of the reasons why it is a bad idea to create many more inodes than needed, because this increases the fsck time. The next important topic to investigate is journaling.

A journal is, broadly spoken, a hidden log file on the partition that keeps track of every transaction on this partition. In case of an unexpected shutdown, the fsck program can

dramatically reduce the repair time afterwards because of the information found in the journal. It depends on the journaling level and settings used, but usually the log is updated before and after a transaction is executed on the hard disk. This log contains only transactions limited to the metadata and the modification of inodes (Stephan 2005). Managing all transactions within a log file creates additional overhead to the file system, and therefore some file systems, such as Reiser3 and ext3, offer different journaling settings, such as ordered, write back, and journal. To improve performance, some file systems also support external journals which are saved on a different block device (Stephan 2005). In contrast, the latest ext4 version supports only one journaling mode and does not support external journals.

In a discussion of journaling, the Linux file system JFS must be mentioned. JFS, which stands for Journaled File System, is a 64-bit journaling file system created by IBM which was designed to provide a robust and full journaling file system for high-performance systems (Best 2000). One great thing about JFS is that its special journaling design allows the file system to be recovered within seconds after an unexpected shutdown, even for large partitions (Best 2000). In comparison, for its maximum partition size of 1 EB, ext4 would need over 119 years to finish one single e2fsck (Mathur et al. 2007). Of course at the moment this is only mathematically a possible partition size, but this shows that the file system chosen is especially important on large partitions where high definition videos will be stored.

Finally, tuning must be mentioned. Just as the standard file system of a Linux distribution may be used, standard parameters could also be selected. When a file system is created on the hard disk, some important parameters, such as block size, amount of inodes, inode size, index support, journaling settings, and many more, can be changed. Again it depends on the file system chosen, but many of these parameters have a huge impact on the behavior and performance of the file system. For example, let us examine the block size. Each file system is divided by formatting the disk into blocks of a fixed size, and the chosen block size determines the allowed file and file system size as can be seen in Table 2.

File System	File Size Limit	File System Size Limit
ext2/ext3 with 1 KiB blocksize	16448 MiB (~ 16 GiB)	2048 GiB (= 2 TiB)
ext2/3 with 2 KiB blocksize	256 GiB	8192 GiB (= 8 TiB)
ext2/3 with 4 KiB blocksize	2048 GiB (= 2 TiB)	8192 GiB (= 8 TiB)
ext2/3 with 8 KiB blocksize	65568 GiB (~ 64 TiB)	32768 GiB (= 32 TiB)
ReiserFS 3.6	1 EiB	16384 GiB (= 16 TiB)
XFS	8 EiB	8 EiB
JFS with 4KiB blocksize	8 EiB	4 PiB

Table 2: Block Size Limitations, SuSE (2005)

So choosing a larger block size leads to a better support of larger files, but it could also lead to a waste of storage space. The problem is that normally if one saves a file which is smaller than the block size, it is not possible to use the remaining space within the block anymore, and, depending on the file system content, this could lead to less free space. Several file system vendors have tried to implement various techniques, such as delayed block allocation, tail packing, and so on, to overcome this problem. The feature tail packing is enabled per default in ReiserFS 3.6. This is a technique by which multiple small files can be saved within a logical block (Jones 2008). The benefit of this technique is that one can save needed space, but the drawback of this solution is that it decreases the write performance. This also the reason why one can deactivate this feature when mounting the partition, which leads to the next factor, mounting options.

After the file system has been created, it must be mounted before the Linux system can use it. A lot of default parameters can be changed in the mounting command or configuration in order to change the behavior of the file system. Possible parameters include journaling parameters, tail function, deactivation of access time stamps and file rights, and many more. Some parameters are more general and valid for all file systems, but most of them are file system specific. If these parameters are changed, the configuration should be carefully tested, because not all changes lead to a better, securer and more stable system.

One parameter which should be carefully examined is the I/O scheduler which is used. Each compiled kernel is configured to use a special scheduler and this scheduler has a huge impact on the performance. Tannbaum (2008 p.467) mentions that the characteristics of multimedia files are so different from those of traditional text files that even playing a simple video creates many new and different demands on the I/O scheduler.

The I/O scheduler is responsible for bundling read and write requests and reordering data packages to minimize disk movements (Tanenbaum 2008, p.777-778). On a Linux system the noop scheduler, the deadline scheduler, the anticipatory scheduler and the complete fair queuing scheduler (CFQ) are available. Whereas the anticipatory scheduler is the default scheduler in the mainline kernel and fits well for desktop systems with single disks, the complete fair queuing scheduler fits better for multi-user environments (Eckermann & Tobey 2010).

Moallem (2008) examined the performance of different Linux I/O schedulers in detail and came to the conclusion that each of them works for a special kind of environment and he also found that different scheduler settings completely change how stable and how fast the system performs. Even though each kernel is compiled with a standard I/O scheduler, one can change the default system scheduler for the complete system with the kernel boot option "elevator=SCHEDULERTYPE". Additionally one can change the default I/O scheduler just for one (or more) special device(s) via the command:

*"echo SCHEDNAME > /sys/block/*DEV*/queue/scheduler"*

(Eckermann & Tobey 2010). These options give the user the possibility of choosing one I/O scheduler for the operating system and a different scheduler for a multimedia library.

2.6 Summary

Even though each Linux distribution has its own preferred file system, as discussed, there are many facts one should consider before being able to choose the best file system and file system setting for a multimedia library. All of the many file systems available were

created based on experience with other file systems, and very often they were created to be the best for a particular environment or situation. Essentially, the main task of a file system is to store, organize, and manipulate files in a fast and efficient way, no matter whether a few large files or several thousands of small files are being stored. To fulfill this task the file system must be precisely tuned. The problem is that each file system has its limitations, such as maximum volume size, maximum file size, maximum count of files in a subdirectory, and so on. At this time huge differences can be seen among the file systems which are generally available. If a file system was created mainly to handle thousands of small files in a very efficient way, it may have huge problems handling extremely large files.

On the other hand, the requirements of multimedia content vary extremely from small files to large files, from a few files to many files, and from time-sensitive to time-insensitive data transfer. Therefore it can be said that one single file system cannot fit perfect for every kind of multimedia library. Of course, one could use almost any file system for multimedia data, but this would only be a stopgap, as the next practical section of this document will show.

Chapter 3. METHODS AND REALIZATION

The project evaluation objective was to identify and evaluate existing Linux file systems and their behavior in multimedia file systems. Analyzing the requirements of multimedia files, such as different file sizes, typical file amount, and so on, was the first step in this project. Multimedia libraries can contain anything from very small to quite large files. To obtain more practically oriented results in the lab, it was decided to create several multimedia libraries containing real multimedia content as part of this project. Each library was filled with one kind of multimedia content, as can be seen in the next table, Table 3.

Lib #	Library Name	Content Information
1	pictures	Pictures of all sizes and formats
2	sound	MP3 and ACC files from several iTunes libraries
3	video	Video files of movies
4	video-hd	Full HD Video files

Table 3: Multimedia Library Content

Of course it would have been much easier to create libraries with simulated files of an exact size, but this would not have simulated a realistic environment. To analyze special situations and strange test results with the multimedia libraries it was still necessary to do tests with simulated files, but the main focus for all tests was using the real libraries. The outcome of these experiments was used as an additional source for the exploratory study of Linux file system features and their positive and negative impacts on multimedia files. After this part of the testing, a list of parameters and file system features, such as block size, journaling settings, mount options, and structure options, was available. The outcome of all tests and the exploratory study was used to repeat several experiments with optimized settings. A conclusive comparison at the end of the project shows the performance effects of using standard file system parameters compared to tuned settings in multimedia environments.

3.1 Multimedia Libraries

Each library which was created as part of this project was stored on a 750 GB hard disk and was filled to approximately 600 GB with one kind of multimedia content (pictures,

sound, video-normal or video-hd files). The exact content of each library was only filtered based on the kind of multimedia type and not on a special file size. So, for example, the library pictures were created by merging several different libraries containing picture files without concern about how small or large they were. The directories (and the number of directories) were also incidentally created by the copy and move process by creating the libraries. Depending on the kind of multimedia content stored, the average file size and the amount of files per library differ greatly, as can be seen in the following table, Table 4.

Library name	Used Size Bytes	Directory count	File count	Smallest File Size	Largest File Size	Average File Size
picture	628,073,484	415,766	1,345,102	0.03 KB	3.2 MB	0.46 MB
sound	628,794,564	28,626	100,621	1.3 MB	9.1 MB	6.10 MB
video	627,756,276	1,317	11,617	5.3 MB	110 MB	52.77 MB
video-hd	628,153,840	243	35	987 MB	36,840 MB	17,526.61MB

Table 4: Multimedia Library Details

3.2 Testing Rules and Environment

The tests were performed in two different system environments:

Environment 1: Default Settings

The first phase of tests was based on a system with kernel and file system default settings and no special tuning was done. All tests were based on the following order and were repeated for all five file system and for all four libraries:

1. Format the hard disk with the test file system and default parameters
2. Mount the test file system with default parameters
3. Collect some file system parameters of the test file system (inode count, size, ...)
4. Copy library X to the mounted test file system
5. Collect free and used space parameters
6. Dismount the test file system
7. Accomplish a forced file system check on the test file system
8. Mount the test file system with default parameters
9. Delete all files from the test file system
10. Dismount the test file system

The complete testing scenario were repeated three times, to ensure accuracy. The numbers shown in the raw data in the diagrams are the average result of these three tests. The exact raw data of each tests can be found at the end of this document in the appendix section.

Environment 2: Optimized File System Settings

Based on the research result the file system settings, such as format options and mount options, were tuned to best fit this kind of multimedia files. Afterward some tests from the first phase were repeated to examine differences to the default settings.

3.2.1 General Testing Rules

- All tests were run at least twice
- Between each test, the system was rebooted to clean up the system cache
- Kernel version and software environment remained unchanged during tests
- All nonessential services were stopped before running the tests

Software environment

All testing were performed with OpenSUSE 11.3 and the latest stable vanilla kernel 2.6.36

The following file systems were tested: ext4, ReiserFS, XFS, JFS and Btrfs

Hardware environment

For the lab environment 4 PCs with the following hardware specification were available:

- Motherboard: ASUS S775 P5Q Deluxe
- Processor: Intel Core 2 Quad Q9400 (4x 2.67GHz)
- Memory: 4GB DDR2 PC-800 (to reduce memory cache effects the available memory was limited by most tests by using the Grub kernel parameter "mem=1000M")

- Hard disk: 2x Samsung SATA2 750GB, 32MB Cache, 8.9ms, 7200rpm (one disk for the operating system, log files, and traces and the other disk for running the test workloads).
- Several other hard disks of the same type were used as library source media

Benchmark Tools

To carry out performance tests the following freely available benchmarks tools and Linux commands were used:

- Bonnie++ v.1.03, which the Linux mainstream benchmark suite used to perform a number of simple tests of hard drive and file system performance. Available from: http://sourceforge.net/projects/bonnie
- Linux tools dd and tar were used to create test files in various sizes as needed
- Linux command line utilities such as: time, stat, free, iostat and so on were used for system diagnostic during the tests

Chapter 4. EVALUATION AND RESULTS

This chapter describes the outcome of some practical experiments with default and tuned system settings. Conceptually, there is always a gap between theory and practice and therefore was it important in this project to include experiments with before unknown test results. Such test results are necessary not only to prove already known facts, but also, and even more importantly, to find unexpected behavior.

4.1 Outcome Test Scenario I: Default Settings

4.1.1 Format Time

Formatting a hard drive partition with a file system is the initial process of creating a new file system. Part of this process is that the file system creates the metadata area which it needs and the block area is prepared. Whereas in the past this process was also known as the process which deletes or sweeps everything from the hard disk, this is not true anymore. The test results in Figure 1 show that formatting a 750 GB hard disk takes from 0.042 to 155.836 seconds.

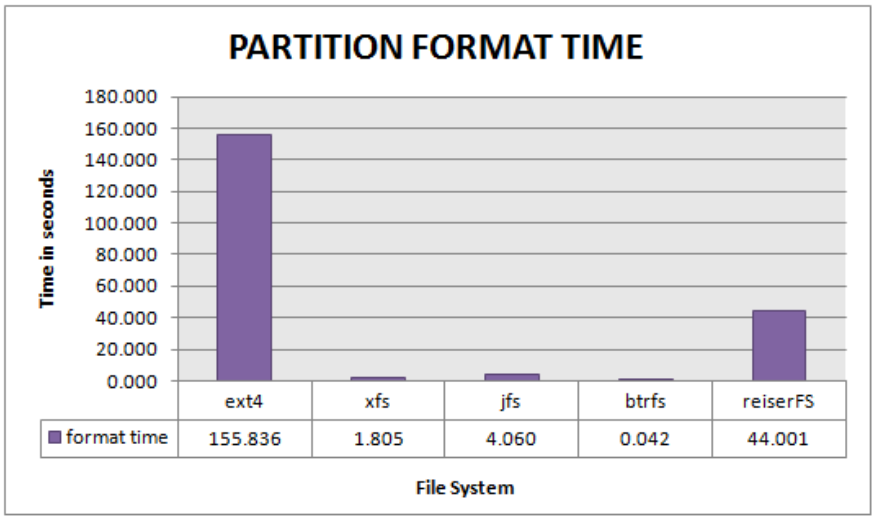


Figure 1: Partition Format Time

It is absolutely impossible to sweep a 750 GB hard disk in less than a second, and it is also not really possible to do it in 155 seconds. Aside from sweeping the block area, a longer format time can also be caused by a different metadata handling. Ext4 is known as a file system with static inode allocation, which means that during the formatting process the exact inode amount is determined and allocated. On the other hand, file systems such as btrfs use a dynamic allocation, which means the inode tables are prepared if they are needed. In general it is not a bad thing if the formatting time takes a little bit longer, because this could also mean that other processes are afterwards faster because the file system was already prepared for it. If a faster formatting time of ext4 is needed one can use the fsck.ext4 option *"-E lazy_itable_init=1"* which has the effect that the inode table is initialized in the background when the file system is the first time mounted (Ubuntu 2010,c). The next fact which can be improved is the default amount of inodes created from the ext4 fsck program. Table 5 shows that the largest amount of inodes needed can be found on the partition of the picture library, because it includes 1,345,102 files and 415,766 directories.

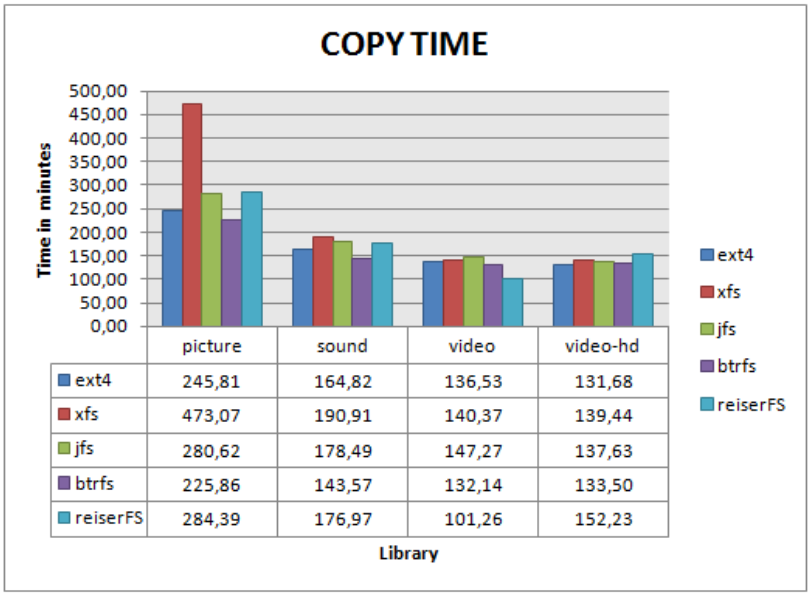
Library name	Inodes needed for library	Inodes created by ext4 fsck
picture	1,760,868	45,793,280
sound	129,247	45,793,280
video	12,934	45,793,280
Video-hd	278	45,793,280

Table 5: Needed Inodes

Together this makes 1,760,868 inodes needed, but the fsck program created 45,793,280 inodes for this partition. This is not only a huge waste of unused inodes, but it also causes an unnecessary burden on our partition. Even worse is the inode result on partitions of the other libraries, because they store much less files and directories but the same default amount of inodes were created. It is possible to reduce the amount of inodes created by formatting the disk via the fsck.ext4 option "-N exact-number-of-inodes" or "-i bytes-per-inode" (Ubuntu 2010, c)

4.1.2 Copy Time

One of the main factors in working with a multimedia library is the time it takes to fill it with data. The diagram in Figure 2 shows that the chosen file system with its default parameters has a huge impact on the time needed.



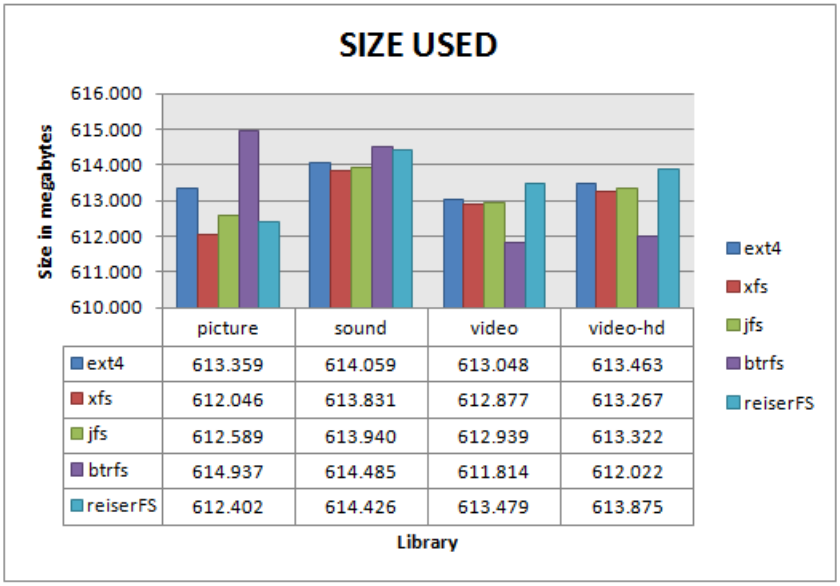
	picture	sound	video	video-hd
ext4	245,81	164,82	136,53	131,68
xfs	473,07	190,91	140,37	139,44
jfs	280,62	178,49	147,27	137,63
btrfs	225,86	143,57	132,14	133,50
reiserFS	284,39	176,97	101,26	152,23

Figure 2: Library Copy Time

Interesting to find out that was, Btrfs was the fastest in nearly every library. This is important information, because it seems that in comparison to other file systems Btrfs behaves very quickly with its default parameters no matter whether it has to handle small or large files. Sweeney et al. (1996) mentions that Xfs was created to perform well with very large files. This is perhaps the reason why Xfs shows better results with larger files than with smaller files. Based on the very bad test results when creating the picture library, it must also be said that Xfs would be a bad choice for this kind of data. The data in the table shows in detail that Xfs needed 247 minutes longer than Btrfs to copy the same content. Which is was nearly twice as long. In general it seems also that ext4, which is the default file system in most of the newest Linux operating systems, is also a good choice for these libraries. Important to note is that the partition was formatted with ext4 and it was not converted from ext3, because 100% ext4 functionality is only available when the partition is originally formatted as ext4.

4.1.3 Statistics for Nearly Full Library

The used and free spaced results of the libraries should not really be surprising. In all cases the same partition size and the same amount of source data was used, and therefore the results in Figure 3 should be equal.

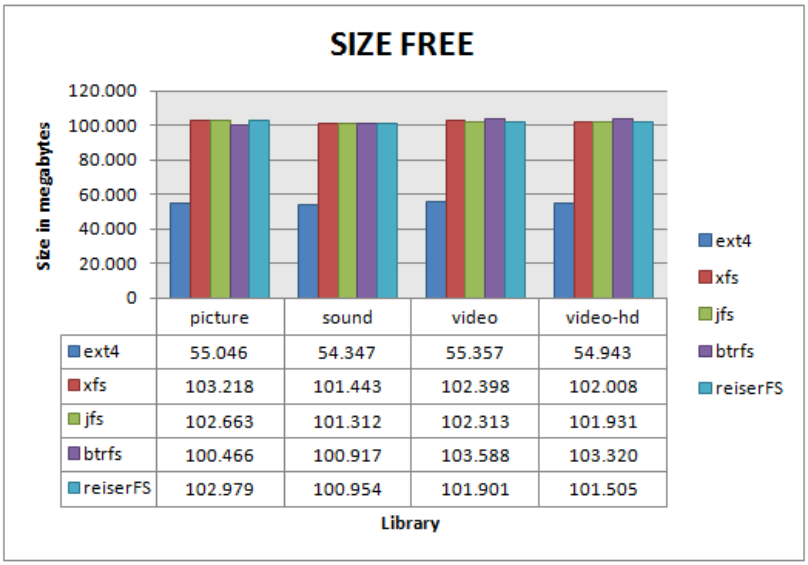


	picture	sound	video	video-hd
ext4	613.359	614.059	613.048	613.463
xfs	612.046	613.831	612.877	613.267
jfs	612.589	613.940	612.939	613.322
btrfs	614.937	614.485	611.814	612.022
reiserFS	612.402	614.426	613.479	613.875

Figure 3: Size Used

XFS is the winner in the picture library because it needed 2,891 MB less than Btrfs for the library content. The picture library is also the library which contains the most and the smallest files. Not every file system can handle such small files in the same way, and for the size used in the test a difference of over 2 GB can be seen. It is interesting that the size used is inversely proportional to the copy time. Whereas Btrfs showed the best copy time in the picture library, it also needed the most space. The reverse was true for the Xfs file system: it needed the least space but also the longest time for the copy process. Among the other three libraries there were mixed results, but the difference between the best and the worst result was never more than 2 GB. Interestingly ReiserFS only shows good results with smaller files. Of course because of tail packing it should work great in environments with smaller files, but this would not necessarily mean that it should work worse with larger files.

Next the free space on the hard disk after the copy job was examined. Although all other file systems in Figure 4 show only a small difference of around 2 GB, ext4 is 48 GB behind!



	picture	sound	video	video-hd
ext4	55.046	54.347	55.357	54.943
xfs	103.218	101.443	102.398	102.008
jfs	102.663	101.312	102.313	101.931
btrfs	100.466	100.917	103.588	103.320
reiserFS	102.979	100.954	101.901	101.505

Figure 4: Size Free

This strange result prompted a lot of additional tests, because it was not at all clear why ext4 showed nearly the same size used, but ~48 GB less free space. A lot of time was spent trying to find the reason for this loss, and whether it was a result of data placement strategies or metadata handling, but the reason was much simpler and could finally be found in the task "formatting the file system". As the next figure, Figure 5, shows this huge gap is already there after the file system is formatted. It does not occur when the data is copied.

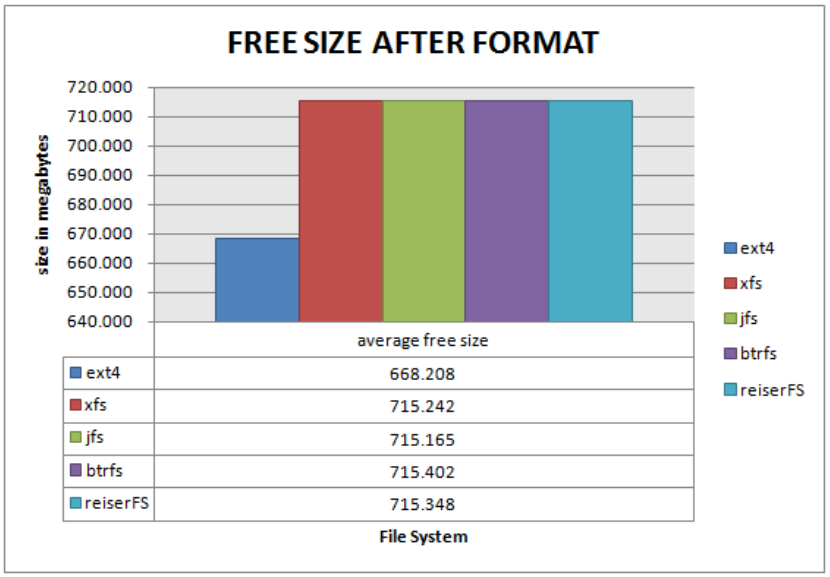


Figure 5: Free Size after Format

Tests with different partitions sizes showed that this gap is always there and that it grows with the partition's size. At last it was learned that this gap is a standard feature of ext4 and it is called "reserved space for root". According to the man page for ext4, 5% of an ext4 file system is reserved for the super user called root, and this percentage should avoid fragmentation and-- the most important fact-- it should prevent that a non-privileged process fill up the complete partition (Ubuntu 2010,c).

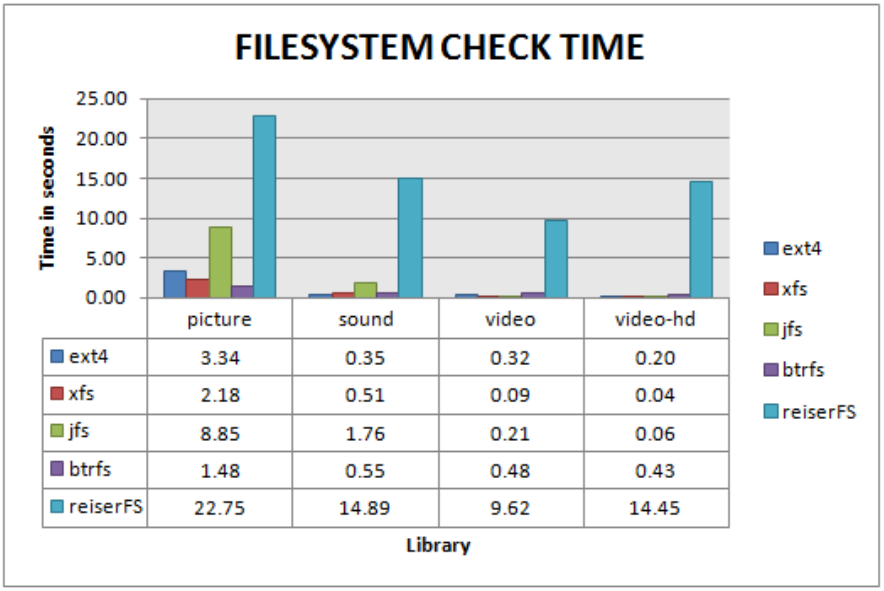
This feature alone can lead to a lot of discussion, but from the viewpoint of this project the following conclusions were drawn:

- Five file systems were tested and only one of them offers this feature
- Filling up a partition is a huge problem on the root file system because it prevents the system from being able to work correctly and the root user from being able to solve this problem
- Filling up a data partition also leads to problems, but the Linux system should still be working as usual and the root user should not have problems to fix this
- File system quotas can be used to restrict the file system space for non-root users and their processes
- And last but not least, 5 percentage as a general rule is not ideal. On a smaller partition this value could be good, but on larger partitions this could be a huge waste of space

Another thing which is important to mention is the fact that all tests were performed as root, and also all results were presented using commands as the root user. So even though the root user was used to show the free and used space, the Linux system did not show the reserved space for the root user when the Linux command "df" was used. This means that normal Linux commands do not show the root user this reserved space, and therefore it could easily be forgotten. On a partition of 2 TB this standard feature would reserve 100 GB!

4.1.4 File System Check

In general the low file system check times of all five file systems in Figure 6 were wonderful news. Only ReiserFS needed a longer time for checking the file system in all libraries, but even this time was not really bad, since the difference is just a few minutes and not a few hours.



	picture	sound	video	video-hd
ext4	3.34	0.35	0.32	0.20
xfs	2.18	0.51	0.09	0.04
jfs	8.85	1.76	0.21	0.06
btrfs	1.48	0.55	0.48	0.43
reiserFS	22.75	14.89	9.62	14.45

Figure 6: File system Check Time

However, such fast file system check times were not always normal, and the following diagram in Figure 7 demonstrates the difference and improvements of, for example, ext4 compared to its older versions.

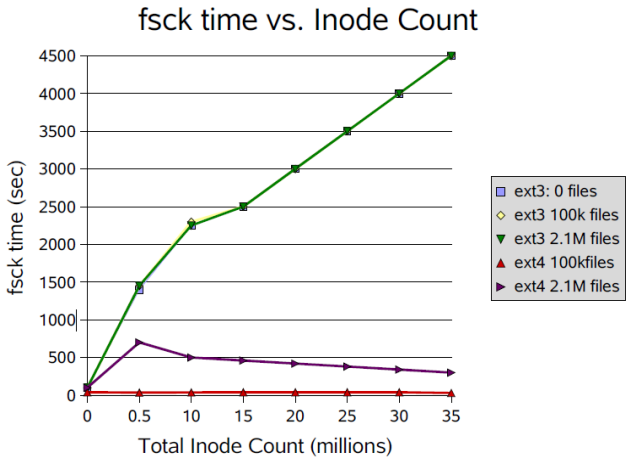


Figure 7: fsck Time vs. Inode Count, Mathur A. et al. (2007)

This diagram above shows that the e2fsck time of ext3 increases "linearly with the total number of inodes in file system, regardless of how many are used," which means that e2fsck takes the same amount of time to repair a file system without a single file as with several millions of files, because only the number of inodes count (Mathur et al. 2007)!

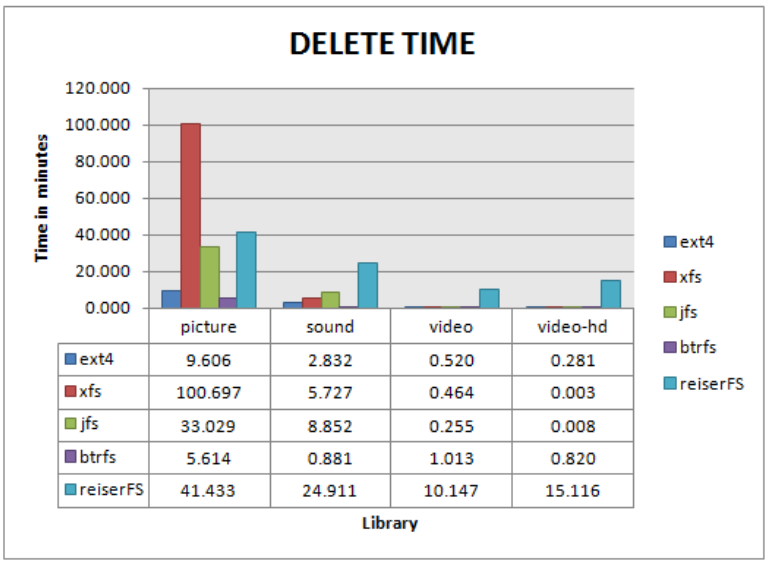
To speed up fsck time, the ext4 developers implemented a mechanism to store all unused inodes within a checksum-secured table, and therefore the fsck time can be 2 to 20 times faster than before (Mathur et al. 2007). The drawback of this solution is that the table with the unused inodes is created by the fsck program, and therefore the first check is always slower.

As can be seen, there are several solutions available to reduce the fsck time, but the easiest task a user could perform is to tune the inode amount for the expected file and directory amount of the multimedia library. Especially if one wants to store larger video files on the file system and knows that the 600 GB library can be filled up with 50 large video files, there is no need to let the ext4 file system create over 45,000,000 inodes for such a partition!

4.1.5 Delete Time

The next main factor in working with a multimedia library is the time for deleting data. In contrast with read requests for files this is a task which seldom happens. Normally a file is deleted if it is not needed anymore, and one might think that time should be not that im-

portant, but this is wrong. Deleting a file has many impacts on a system and some of these impacts will be discussed here first. Deleting a file happens not only when the user wants to get rid of one. Instead very often deleting is a part of a move process. If a file is moved within a partition, only entries in the metadata must be changed, and therefore this is not a strenuous process. On the other hand, if data is moved from one partition or hard disk to another, this move process could create a large load on the system, because in this case there are both create and delete processes for the block of data. The next diagram in Figure 8 illustrates this point:



	picture	sound	video	video-hd
ext4	9.606	2.832	0.520	0.281
xfs	100.697	5.727	0.464	0.003
jfs	33.029	8.852	0.255	0.008
btrfs	5.614	0.881	1.013	0.820
reiserFS	41.433	24.911	10.147	15.116

Figure 8: Library Content Delete Time

Even though 600 GB was used in each library, there are extremely large delete time differences! This time the analysis of the result much more difficult because there is no straight forward rule such as "file system X needs always more time than file system Y". But, why are there even differences within a library, when each file system contains the same amount of files and directories?

The first reason could be the file amount. From left to right each library contains less files. For example, the pictures library includes 13 times more files than the sound library and 38,431 times more files than the video-hd library. One trend which can seen is that the amount of files has an impact on the delete time but not exponentially. The differences can be seen better in the raw delete time data. Xfs showed that it cannot handle a large amount of small files as efficiently as the other file systems, and the delete time of the

picture library demonstrates exactly the same problem. It can also be seen that XFS can handle large files better than its competitor because it produced the best results in the video-hd library. Another question which should also be discussed is, what happens exactly if a file is deleted? The short answer is, it depends entirely on the file system used. A lazy approach is that the files within a directory inode are only marked as available, and an excessive approach is that additional a real cleanup in the block area had happened. Which approach is better depends mainly on the three factors: time, security, and system load.

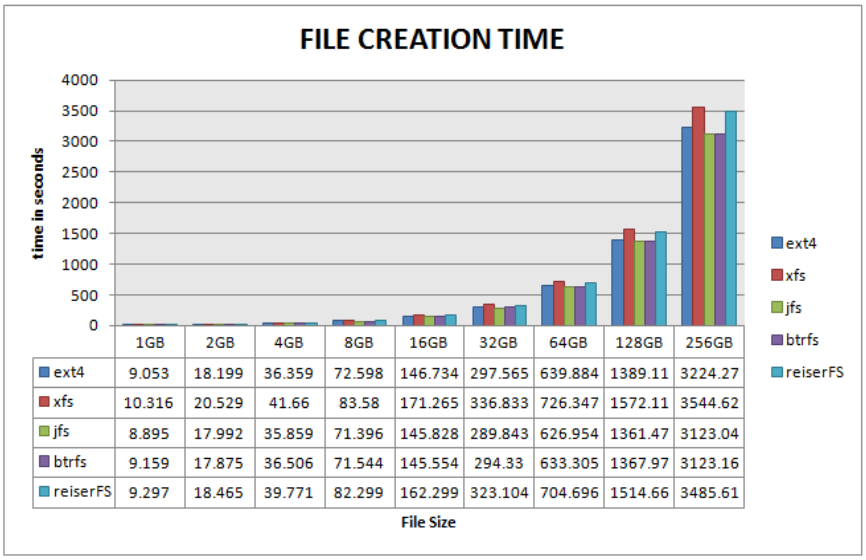
The first factor is time. If a library is filled with content and the user want to save new material in it, he or she must first delete some or all older content. It depends on the user, but in this case one could say that the user would prefer a delete time of seconds instead of minutes or hours. Even if the user knows that the data blocks will not be deleted and only the metadata area is cleaned up, he or she can start in a short time with the initial task of copying new material to the partition. On the other hand there is the factor security. Everyone who has seen a police movie, such as CSI, knows that deleted data can be restored from a hard disk. Here again it depends on the needs of the user, but even though fast deleting time is always nice, it may be that it is more important that the deleted content cannot be recovered anymore. Not only people with questionable intentions try to securely delete data, but also companies and governments are interested in preventing other people from recovering deleted data.

The last factor is the work and system load. Cleaning up metadata and block data costs not only time, but it also costs resources. It depends on the overall system power, but cleaning up a larger partition could produce such a load on the system that in the meantime it cannot be used for other tasks. As can be seen in the test results above, this concerns a time difference of 1 seconds to 100 minutes during which the system is busy. Especially in multimedia environments it is important that the system can deliver content in a fast and dedicated time frame. Take, for example, in a situation where several video streams send data to media players at the same time and the administrator must delete some older files to have room for newer video files. Would it be useful in this situation for the delete process to be secure, but the video stream is not fluent anymore? Especially

in smaller or cheaper multimedia environments this has a larger impact. It would not be an unusual situation for the boy to be sitting in the living room watching a recorded football match while his sister is sitting in her room watching a recorded soap opera which she loves. This would not be the best moment for the father to start to securely cleaning up some older files on the multimedia library because he wants to make space for new content.

4.1.6 File Creation Time

The libraries include files with a lot of different file sizes, and therefore additional tests with simulated files which all had the same size were performed. Based on the results of these libraries it was interesting to see how much the creation time had an impact on the copy time. In the following test files from 1 GB up to 256 GB were created using the Linux dd command. Figure 9 shows that the file creation time continuously grew as expected.



	1GB	2GB	4GB	8GB	16GB	32GB	64GB	128GB	256GB
ext4	9.053	18.199	36.359	72.598	146.734	297.565	639.884	1389.11	3224.27
xfs	10.316	20.529	41.66	83.58	171.265	336.833	726.347	1572.11	3544.62
jfs	8.895	17.992	35.859	71.396	145.828	289.843	626.954	1361.47	3123.04
btrfs	9.159	17.875	36.506	71.544	145.554	294.33	633.305	1367.97	3123.16
reiserFS	9.297	18.465	39.771	82.299	162.299	323.104	704.696	1514.66	3485.61

Figure 9: File Creation Time Test 1

For all file sizes, xfs had the worst (longest) creation time, followed by reiserFS. The fastest result was produced by jfs and btrfs, both of which always needed nearly the same amount of time.

The next test was to fill a library with files of a particular size. The first task was based on creating 500,000 1MB files, and the second task on creating 20 files of 25 GB each. Both tasks were performed using the dd command with the file system cache deactivated. In summary, both tasks in Figure 10 created the same amount of data, 500 GB, and the only difference was in the amount of files.

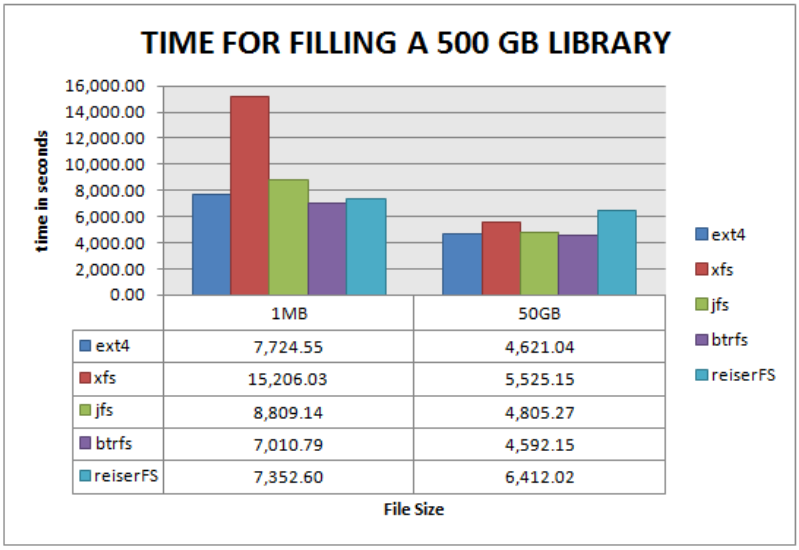


Figure 10: File Creation Time Test 2

This simulated test also produced the result that xfs is not the best file system for such so many small files and that jfs is again a little behind btrfs. The results of ext4 are very good and comparable to library copy tests. It seems also that all the file systems which were tested behave much better in environments with larger files. This is again an interesting outcome, because based on the test results one could say as a rule of thumb that environments with smaller files must be more carefully planned than those with larger files.

4.2 Outcome Test Scenario II: Optimized Settings

This chapter will clarify some of the effects of changing default file system parameters and their impact on the multimedia libraries. The last chapter showed that the same file system parameters produce different test results for different kinds of libraries. Understanding this is important, because this means that changing some settings also has its

pros and cons. On the one hand, different settings can lead to a much more efficiently tuned system, but, on the other hand, they can also lead to a much more unstable or risky system. Also it may be that the default settings fit better for one kind of environment and changing them could lead to poorer results because the default settings were already the best. Finding the right balance between better performance and a more risky system is very difficult and depends on many factors, such as the environment and the availability needed from the system. To avoid discussions about too risky parameters this project only tested parameters which could improve the system without increasing the instability of the system too much. Several tests in this project led to either the same or, very often, even much worse results. A discussion of all negative results would be very interesting, but is not within the scope of the project. Therefore this chapter provides only a small subset of the results.

4.2.1 Mount Options

Many documents such as (Brindley 2010, p.9) discuss several mount options to improve performance, and two options called noatime and nodirtime are mentioned very often. Both mount options are used to prevent the update of the last access timestamp each time a file or directory is accessed. Not only does this feature reduce unnecessary journal activity and useless disk writes, but it can also help to reduce the power consumption (Brindley 2010, p.9). This feature can be activated via the mount option "-o no-atime,nodirtime" or by using a similar entry in /etc/fstab. Because the access time is not really relevant for most of our multimedia environments, this feature can be activated to improve performance. Especially in environments with a lot of read and search actions in the file system this feature could be a benefit. In all the following tests the partitions were mounted with the option "-o noatime,nodirtime".

4.2.2 Ext4

Earlier tests in this document have already shown the impact and importance of the for-matting process parameters. Among all the file systems tested, ext4 needed the longest

time for the formatting process. ext4's reservation of 5% of the space for root and the too large amount of default inodes can only be manipulated by using different ext4 format options. Several ext4 formatting tests were performed to find better parameters. To show the impact of different parameters four ext4 formatting tests are compared in Figure 11.

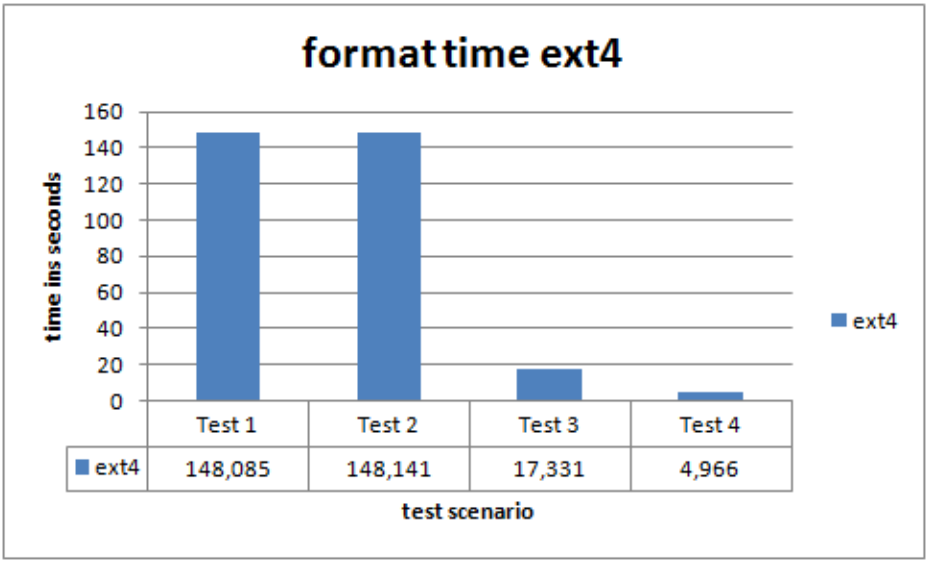


Figure 11: Format Time ext4

The first Test time was obtained using the default format options. The second Test time was obtained by additionally deactivating the reserved root space with the option "-m 0". In addition to the option from Test 2, Test 3 included option "-N 3000000" to create a much smaller amount of inodes. In Test 4, in addition to the other changes, the inode allocation delay was also deactivated by using the option "-E lazy_itable_init=1". The result of these tests are very helpful, first because they show that the tuned format time is 29 times faster than the default time. Second, with the option "-m 0" the 5% reserved space for root has been deactivated, which leads to ~46 GB of more free space for the libraries. The next thing which should be checked in the impact of these parameters is one of the main factors in this project, the copy time. Figure 12 shows the copy time of the picture and video-hd library with the optimized format options. The copy time of the picture library increased by 9.6 minutes and by 0.45 minutes when copying the video-hd library.

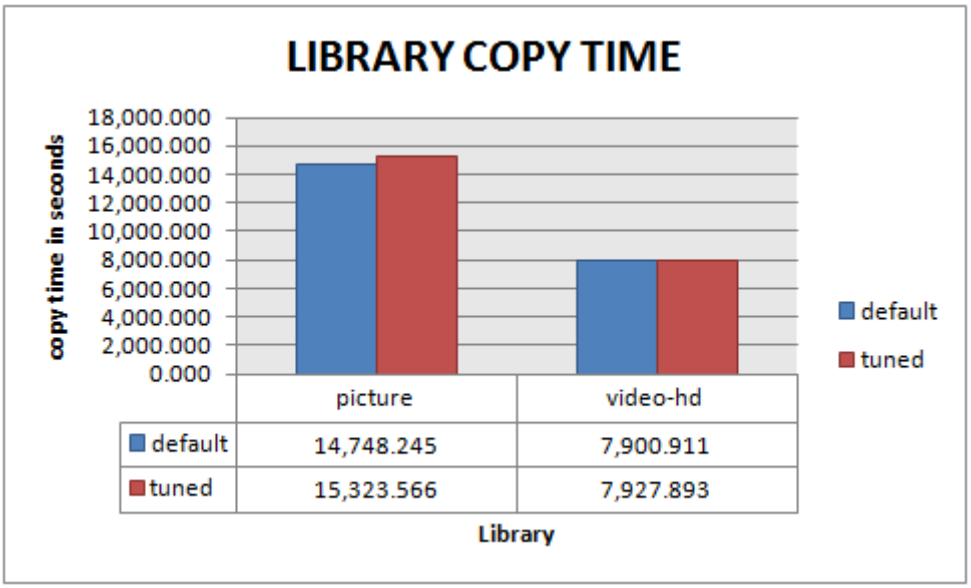


Figure 12: Tuned Library Copy Time

This slightly increased copy time was produced by the delayed inode allocation features which saved 12.4 seconds of formatting time. Of course there are situations where this feature would be a benefit, but based on these test results it would be better to decrease only the amount of inodes. This alone saved 130.8 seconds of formatting time.

4.2.3 Xfs

The next file system which had shown room for improvement was xfs. The format time of xfs was already very good, so there was no need to improve it, but the time for creating and deleting small files was not so good. In general it seems that xfs could not handle a very large number of metadata operations as fast as the other file systems. The best option the man page of xfs lists to improve metadata operations is to manipulate the size of the xfs log by using the option "-l size=XXm". Additionally it would also be possible to move the log to a different partition to increase the performance, but because all of the test scenarios in this project are based on a single multimedia disk environment, this option was ignored. The next parameter the Linux man page mentions to improve performance is the parameter "-l lazy-count=1", which basically reduces superblock modifications (Ubuntu 2010,a). The benchmark tool bonnie++ was used to create test results using the xfs default settings and the xfs tuned settings. Figure 13 shows the results of the

default format parameters compared to Figure 14 with the optimized format settings "-l size=64m lazy-count=1".

```
Version 1.03d       ------Sequential Output------ --Sequential Input- --Random-
                    -Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
Machine        Size K/sec %CP K/sec %CP K/sec %CP K/sec %CP K/sec %CP  /sec %CP
msc           40000M           87193  10 37752   5           92917  10  57.1   0
                    ------Sequential Create------ --------Random Create--------
                    -Create-- --Read--- -Delete-- -Create-- --Read--- -Delete--
                  files  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP
                      5    38   0 +++++ +++    38   0    37   0 +++++ +++    37   0
```

Figure 13: Bonnie++ Output with Default Settings

```
Version 1.03d       ------Sequential Output------ --Sequential Input- --Random-
                    -Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
Machine        Size K/sec %CP K/sec %CP K/sec %CP K/sec %CP K/sec %CP  /sec %CP
msc           40000M           87098  10 37866   5           93120  10  56.2   0
                    ------Sequential Create------ --------Random Create--------
                    -Create-- --Read--- -Delete-- -Create-- --Read--- -Delete--
                  files  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP
                      5    37   0 +++++ +++    37   0    37   0 +++++ +++    37   0
```

Figure 14: Bonnie++ Output with Tuned Settings

Both tests results were obtained using the command "time bonnie++ -d /media/cpcheck/ -u root -s 40000 -f -b -n 5" and therefore produced the same amount of benchmark data. Most of the resulting numbers are the same and one can see very small differences only in areas such as sequential input and sequential delete, but these are also the areas where improvements are needed. Even though the differences found are very small they do have a measurable impact on situations where, for example, a copy job needs two or three hundred minutes. The real impact of the above mentioned parameters on the libraries can be seen in Figure 15 and Figure 16.

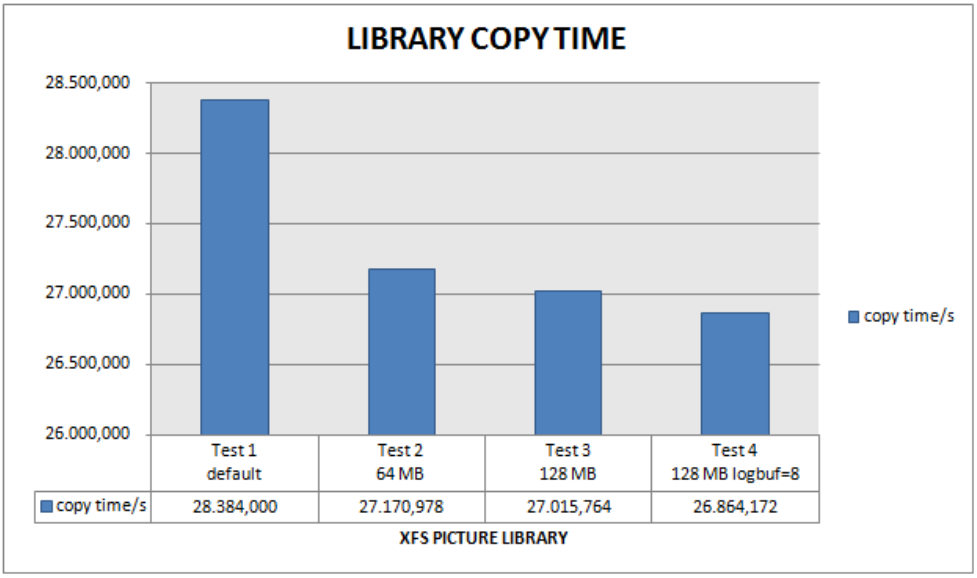


Figure 15: Xfs Picture Library Copy Time

Figure 15 shows that the copy time needed decreased from left to right where 30.4 minutes less were needed for the library copy job! The largest gap in this diagram can be seen between Test 1 with default settings and Test 2 where the main difference was an increased xfs log of 64 MB. Increasing the log from 64 MB to 128 MB in Test 3 did not change the copy times dramatically any more. The picture library is the library with the smallest and largest number of files and thus benefitted greatly from these tuning settings. Therefore the next question was what effect would these parameters have on the video-hd library which contains large files. Figure 16 shows the copy time of the tuned Linux environment compared to the default settings, and, as can be seen, there is no huge difference.

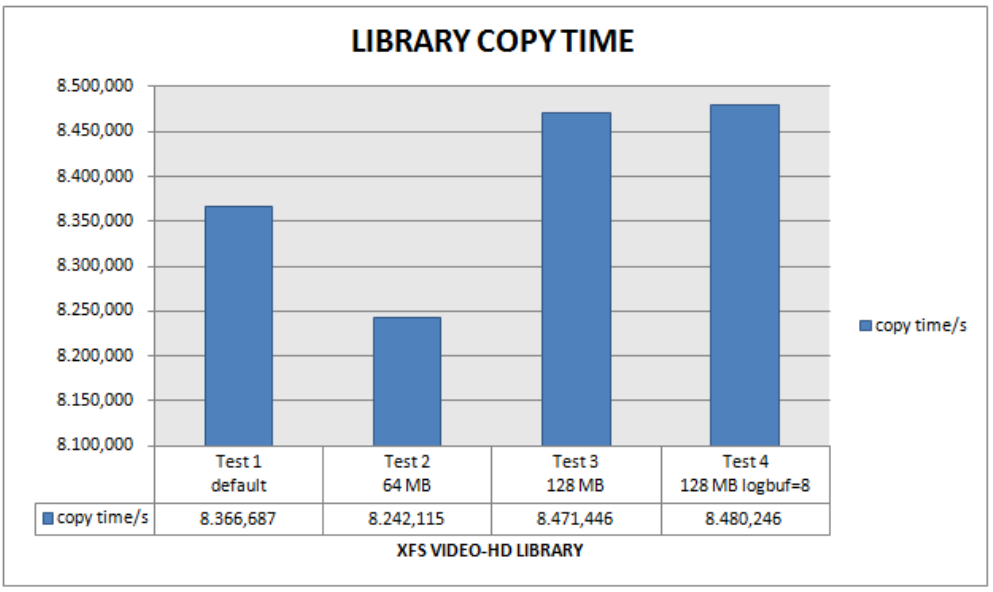


Figure 16: Xfs Video Library Copy Time

Even though the copy time in the second Test was 2.06 minutes faster than in Test 1 with the default settings, this is not comparable with the 30.4 minutes faster copy job result in the picture library. Increasing the log size and the log buffer in Test 3 and Test 4 leads in this library to an approximately 2.01 minutes longer copy time than with the default parameters in test 1. In summary, it can be said at this point that the increased log size of 64 MB brought the biggest improvement for the two libraries and that the other settings which were tested produced even a little bit worse results.

Now it is time to discuss the impact of these parameters on the deletion time of xfs. The last chapter showed that the deletion time of xfs was much worse in environments with many small files. The next diagram, Figure 17, shows that the tuning parameters tested above also have a positive effect on the deletion time. The largest time difference is 5.35 minutes and can be seen between Test 1 with the default setting and Test 4 with increased log settings.

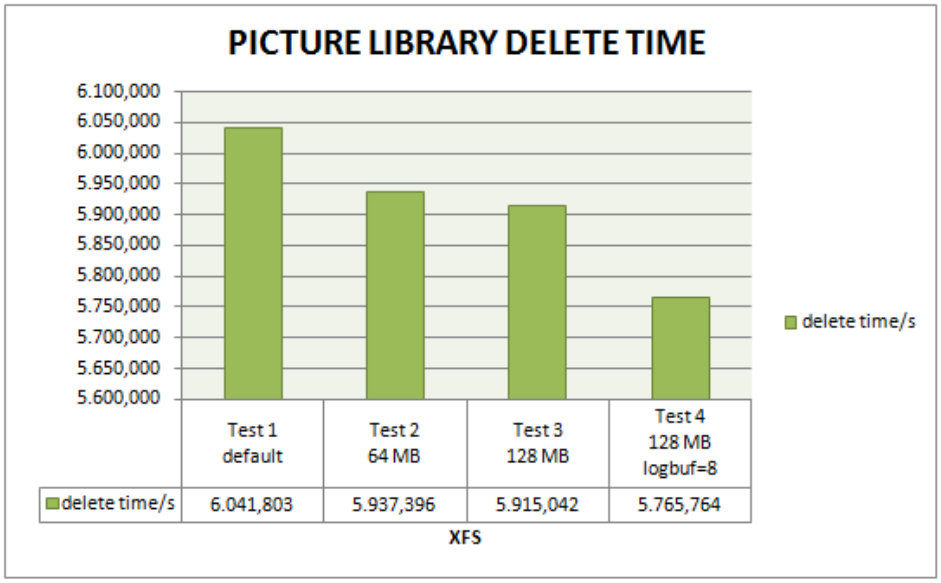


Figure 17: Xfs Picture Library Delete Time

This was the next proof that changing the xfs log size to 64 MB produces a better results in environments with many small files than one gets using the default settings.

4.2.4 ReiserFS

Several tests with ReiserFS format options and Bonnie++ and some theoretical research led to the idea that changing the ReiserFS journaling mode to "writeback" or the Linux I/O scheduler to "deadline" could have a good impact on the performance. To check this out these settings were used to repeat the sound library test where ReiserFS had not shown very good results. Figure 18 and Figure 19 show the impact of these two parameters on the copy and delete tasks using the sound library.

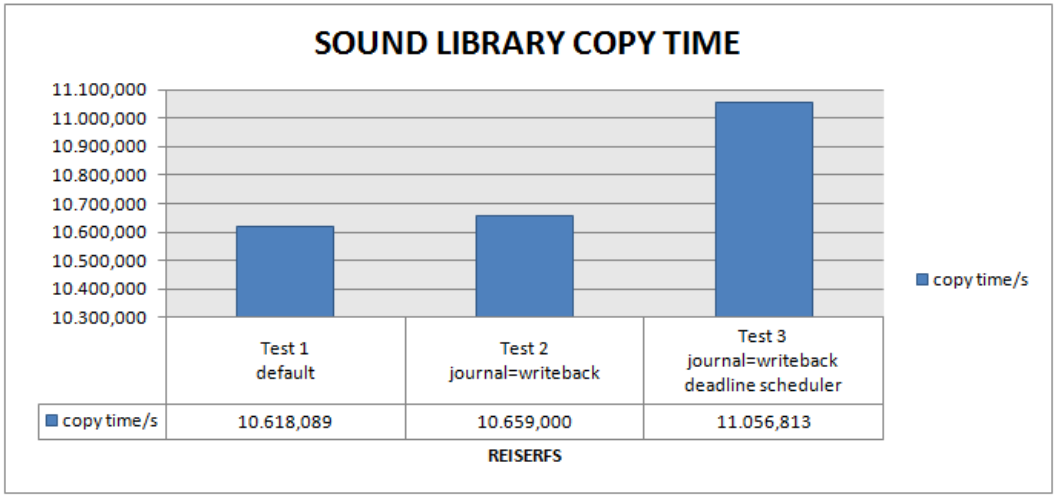


Figure 18: ReiserFS Sound Library Copy Time

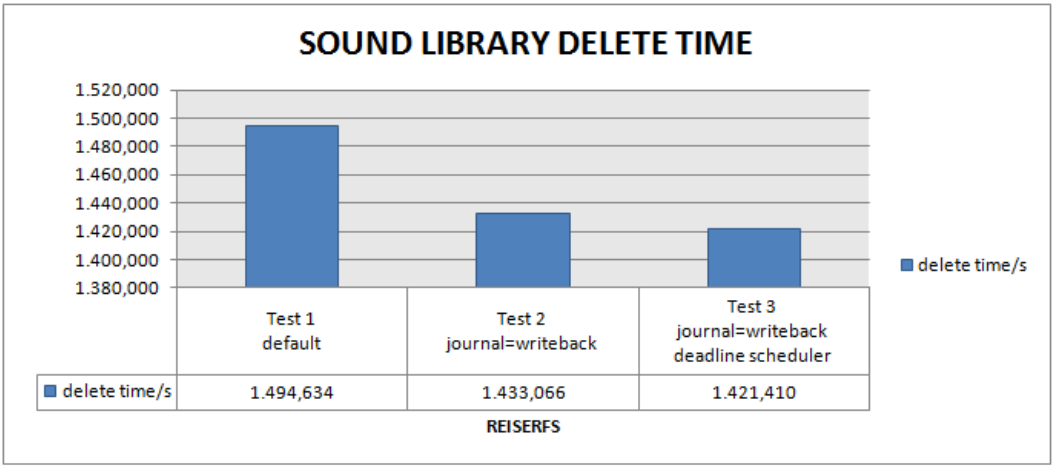


Figure 19: ReiserFS Sound Library Delete Time

The problem is that this time the results were not as clear as in the other tests discussed above. Using the journal mode "writeback" led to a longer copy time of 0.68 minutes but to a shorter deletion time of 1.02 minutes. In the copy results of test 3 with the additional change to the deadline scheduler there was a much worse result in the copy time, but again a better result in the deletion time. Simply said, using these two tuning parameters leads to the situation where one could choose between a better copy or deletion time, but not both.

4.2.5 JFS

JFS showed good results in every respect. No matter whether JFS has to handle small or large files, the default settings of this file system seem to fit well for every environment. Therefore it was decided to test different I/O scheduler settings to see if JFS could benefit from a different scheduler. This time the test results were clear because, as Figure 20 and Figure 21 show, JFS produced better copy and deletion times with the cfq-scheduler.

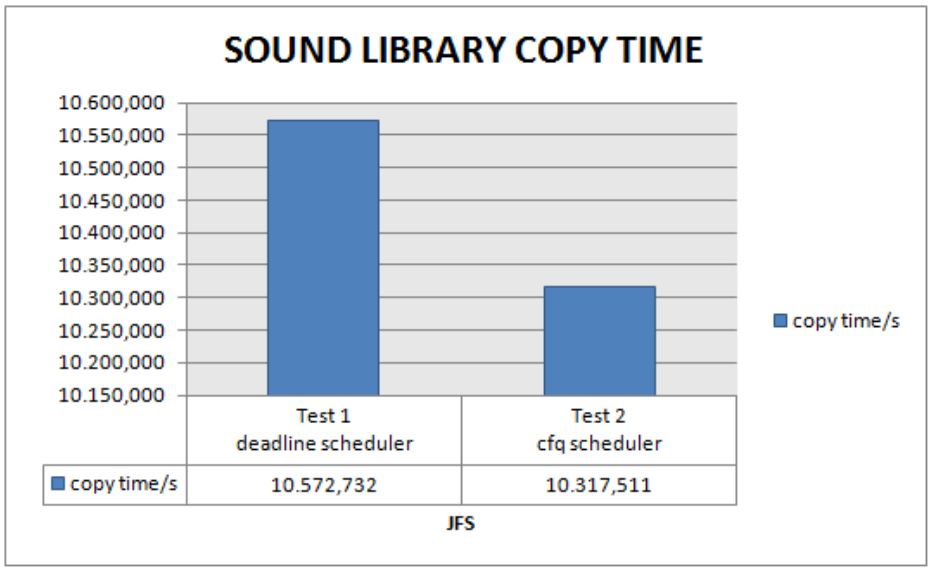


Figure 20: JFS Sound Library Copy Time

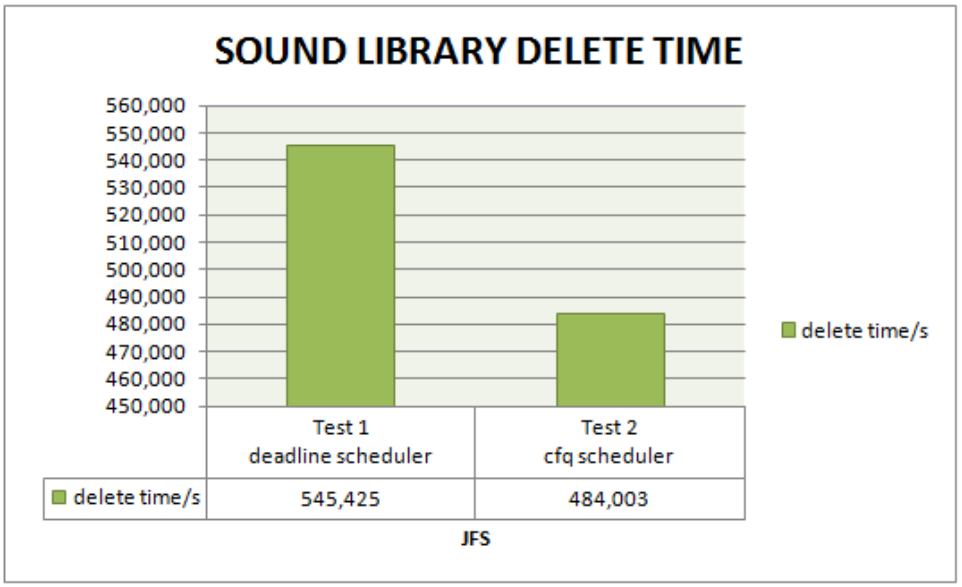


Figure 21: JFS Sound Library Copy Time

Whereas in the ReiserFS performance tests the deadline scheduler produced a better deletion outcome, the results in Figure 20 and Figure 21 show that in both tests JFS worked faster with the cfq scheduler. This is again an interesting outcome because it shows that a different scheduler can,for example, lead to a better deletion performance on one file system while the same scheduler leads to a worse deletion performance on a different file system.

Chapter 5. CONCLUSIONS

Based on the results which were achieved in this project due to the different experiments, it is proven that one kind of file system using only the default parameters, cannot fit perfectly for every type of multimedia. These experiments also show that there is large variability in the results which are summarized in this chapter.

5.1 Lessons Learned

In summary, one can see that there were differences of up to 155 seconds in the formatting time, and up to 247 minutes in the copy time, which is nearly twice as long as the best file system needed in this test. There were 48 GB of difference in free space on a ~700 GB partition, 22 seconds difference in a standard file system check, and, last but not least, 95 minutes of difference when deleting the same library. The research experiments in this project also showed that these time differences vary extremely depending on the size and amount of files stored. Even though some file systems, such as Btrfs and JFS, showed great results in all tests, there are still differences within each library. A very interesting outcome of this project is that the largest problems shown in these test results could be easily manipulated by using some simple parameters. For example, the format time of ext4 could be decreased by 29 times and at the same time 48 GB of space could be freed for user and daemon processes. Also, the very poor copy performance of xfs using the picture library could be decreased 30.4 minutes just by using a larger xfs log parameter. The test results of ReiserFS 3.6 were a little disappointing, because for years ReiserFS was the default file system on every SUSE Linux system, but the results are bad compared to the other file systems tested. One reason for this could be that the Reiser developers have perhaps mainly been working on the next version, version 4, of this file system and have not spent that much time on improvements on the older version any more. In any case, according to the test results of this project, it would not be a good decision to use ReiserFS 3.6 for these kinds of multimedia libraries without using better

tuning options. The results of Btrfs in this project were very interesting. Of course this file system is still under development and is not marked as stable, but in nearly every test it was in first or second place. Compared to the other file systems, the test results for Btrfs in this project show that in the future an excellent and well-performing file system could be available which can handle both small and large files very well. The impact of I/O schedulers on the performance should not be underestimated and, as tests in this project have shown, the same I/O scheduler can produce different results on a different file system. A general overview of the test results with default parameters can be seen in Table 6.

Task	Libary	Best Result	Worst Result
Partition format time	---	btrfs	ext4
Library copy time	Picture	btrfs	xfs
	Sound	btrfs	xfs
	Video	reiserfs	jfs
	Video-hd	ext4	reiserfs
Used space after Library copy job	Picture	xfs	btrfs
	Sound	xfs	btrfs
	Video	btrfs	reiserfs
	Video-hd	btrfs	reiserfs
Free space after Library copy job	Picture	btrfs	ext4
	Sound	btrfs	ext4
	Video	reiserfs	ext4
	Video-hd	reiserfs	ext4
File system check time	Picture	btrfs	reiserfs
	Sound	ext4	reiserfs
	Video	xfs	reiserfs
	Video-hd	xfs	reiserfs
Delete time	Picture	xfs	btrfs
	Sound	btrfs	reiserfs
	Video	jfs	reiserfs
	Video-hd	xfs	reiserfs

Table 6: Overview of Experiments with Default Settings

The strengths and weaknesses of this research should be mentioned. The theoretical part is based on many well-known books, theses and dissertations about multimedia, Linux operating systems and Linux file systems which is one of the project strengths. Advisedly this project has not referenced many benchmarks from other authors, because they may

not be accurate anymore. The Linux kernel changes so quickly, and therefore it is possible that some benchmarks cannot reproduced with a newer kernel. Therefore it was better to produce new benchmarks which are based on the newest kernel version. The main weakness this project has is that the practical experiments were mainly based on self-created libraries. If someone would create a new library with the same kind of content, it would not be very surprising to receive see some different test results. The reason for this is that all of the libraries were created by copying several libraries together, and this has led to a wild mixture of file sizes and directory structures. This method has led to a more natural situation, but also to a environment which cannot be recreated 100% again. In retrospect this method was still a good decision, because the goal of the experiments of this project was to examine how one special multimedia library behaves on several different chosen file systems.

5.2 Research Question Revisited

The first main research question in this project was "What impact does a file system and its characteristics have on the performance of a multimedia system?" The answer to this question is not easy, because it depends on many factors. This document has already shown that one would receive different test results and behavior if the underlying file system changes, but this document has also shown that the behavior changes if the underlying file system does not change, but the kind of multimedia content changes. The experiments with tuned settings show that even one parameter, for example can increase the copy time, while at the same time the delete time decreases.

How large the impact on the multimedia system is depends on many facts such as amount of files, type of content, amount of data and the streaming functionality needed. One interesting fact this research project has shown is that there is not only one kind of multimedia system available. Each system is different and each of them produces different results. Whereas a small personal multimedia library server can perhaps deliver two streams at the same time without problems, it could break down in an environment

with 10 or more streams. So in conclusion, in answer to the research question one could say that the file system usually has a large impact on a multimedia library, but it depends on expectations of the multimedia library owner if this impact is important or can be ignored.

The second main research question in this project was "Which Linux file system fits better for which kind of multimedia files?". Based on the outcome of the experiments, this again is difficult to answer. The main problem is that the experiments have shown very large variability in the results and there is no straight winner in all disciplines. The next problem is that most of the best results in Table 6 were produced by the unstable file system btrfs. Although btrfs produced such great results, due to the instability, it should be carefully considered! The libraries picture and sound, which contained smaller and more files than the other libraries were most affected by the underlying file system. The Libraries video and video-hd contained the largest files and in these environments most of the results were marginal. Based on all experiments in this project, one could come to the conclusion that especially environments with more and smaller files should be carefully examined. As illustrated in the following table, Table 7; for every kind of multimedia library three recommended options are listed. Due to the fact that Option 1 produced the best result it should be considered as first choice. However, if Option 1 for any reason is not available or undesirable then Option 2 or Option 3 are recommended.

Library	Average file size	Amount of Files	Option 1	Option 2	Option 3
Picture	0.46 MB	1,345,102	btrfs	ext4	jfs
Sound	6.10 MB	100,621	btrfs	ext4	reiserfs
Video	52.77 MB	11,617	reiserfs	btrfs	ext4
Video-hd	17,526.61MB	35	ext4	btrfs	jfs

Table 7: File System Recommendation for Specific Library

5.3 Future Activity

Every few months a new Linux kernel version will be released, and with each new kernel version there is a chance that the code of the file systems or of some built-in functions, such as the I/O scheduler, could have changed. Especially btrfs, which is currently not marked as stable in the kernel source code, could experience code changes to achieve a stable status in the near future. Whereas such changes or improvements are normally good for the industry, they also lead to the problem that older benchmarks or experiments need to be redone, because they may not be valid anymore. This is one of the reasons why experiments and benchmarks need to be repeated regularly.

5.4 Prospects for Further Work

This research project has shown that every Linux file system behaves differently with different kinds of multimedia files. On Linux there are several file systems available, and each of them offers different possibilities for improvements. Based on the experience of this project, it would be advisable to create a separate partition for each kind of multimedia file and to tune each partition exactly for this kind of file. Helpful for this task would be an exact list of formatting and mount parameters which exactly fit the current Linux kernel, the file system used, and the kind of multimedia files. Differences of 247 minutes in the copy time should not be ignored, especially if improvements can be made by changing only a few parameters. This tuning is even more important when the multimedia server has to handle several streams at the same time. This project has even found large performance differences between different Linux file systems when nearly all tests were based on one single user request. This leads to the question whether one would see the same test results in more stressed situations with several user requests at the same time.

REFRENCES CITED

Best, S. 2000. "JFS Log" [Online] Available from:
http://jfs.sourceforge.net/project/pub/jfslog/jfslog.pdf [accessed 23 August 2010].

Brindley, L. 2010. "Red Hat Enterprise MRG 1.3 Realtime Tuning Guide" [Online]
Available from: http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_MRG/1.3/pdf/
Realtime_Tuning_Guide/Red_Hat_Enterprise_MRG-1.3-Realtime_Tuning_Guide-en-
US.pdf [accessed 10 December 2010].

Busse, I. & Deffner, B. & Schulzrinne, H. 1995. "Dynamic QoS Control of Multimedia Applications based on RTP" [Online] Available from:
ftp://gaia.cs.umass.edu/pub/Buss9601:Dynamic.ps.gz [accessed 21 August 2010].

B&H 2010. "Digital Camera Resolution Chart" [Online] Available from:
http://www.bhphotovideo.com/FrameWork/charts/resolutionChartPopup.html
[accessed 22 August 2010].

Carrier, B. 2005. *File system forensic Analysis*
Available from: Addison Wesley, ISBN 0-321-26817-2.

Fulton, W. 2010. "Memory cost of images" [Online]
Available from: http://www.scantips.com/basics1d.html [accessed 22 August 2010].

Chapman, N. & Chapman, J. 2009. *Digital Multimedia* (third edition)
John Wiley & Son, Ltd, ISBN 978-0-470-51216-6.

Eckermann, M. & Tobey, B. 2010. "Two Paths to Server Performance" (Magazine)
Novell Connection Magazine JUL_2010.

French, S. 2008. "Around the Linux File System World in 45 minutes" [Online]
from:http://ols.fedoraproject.org/OLS/Reprints-2008/french-reprint.pdf.

Gartner 2007. "File System Innovations Growing in Importance"
ID Number G00145638.

Galvin, P. B. 2005. *Operating System Concepts*
Wiley, ISBN 0-471-69466-5.

Gemmell, J. et al. 1996. "Multimedia Storage Servers: A Tutorial" [Online] Available from:
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=384117
[accessed 18 December 2010].

Halvorsen, P. et al. 2003. "Storage Systems Support for Multimedia Applications" [Online]
Available from: http://www.duo.uio.no/sok/work.html?WORKID=70852
[accessed 18 December 2010].

Helba, S. 2009. *Investigating Hard Disks, File and Operating Systems*
Course Technology, ISBN 978-1435483507.

Johnson, S. & Huizenga, G. & Pulavarty, B. 2005. *Performance Tuning for Linux Servers*
IBM Press, ISBN 0-13-144753-X.

Jones, T. 2008. "Anatomy of Linux journaling file systems" [Online]
Available from: http://download.boulder.ibm.com/ibmdl/pub/software/dw/linux/l-journaling-
filesystems/l-journaling-filesystems-pdf.pdf [accessed 20 November 2010].

Kang, S. 2007. A dissertation about "Flexible allocation and space management in storage systems" [Online] Available from: http://repository.tamu.edu/handle/1969.1/5968 [accessed 18 December 2010].

Kernelnewbies 2010. "Ext4" [Online]
Available from: http://kernelnewbies.org/Ext4 [accessed 22 August 2010].

Klein, D. 2008. "History of Digital Storage" [Online]
Available from:
http://download.micron.com/pdf/whitepapers/history_of_digital_storage_wp.pdf
[accessed 15 August 2010].

Leung, A. 2009. A dissertation about "Organizing, indexing, and searching large-scale file systems" [Online] Available from http://www.ssrc.ucsc.edu/pub/leung09-ssrctr0909.html [accessed 18 December 2010].

Liao, H. 2003. "Storage Area Networks Architectures" [Online]
Available from: http://www.pmc-sierra.com/cgi-bin/document.pl?docnum=2022178
[accessed 15 August 2010].

Linux Magazin (2010) "In der Baumschule"
Linux Magazin edition 09/10 (German).

Loizides, C. 2001. "Analyse und Simulation von Fragmentierungseffekten beim ”ReiserFS” Dateisystem" [Online]
Available from: http://www.informatik.uni-frankfurt.de/~loizides/reiserfs/reiserfs-diplom-loizides.pdf [accessed 13 April 2010].

Love, R. 2005. *Linux Kernel Development*
Novell Press, ISBN 0-672-32720-1.

Mathur, A. & Cao, M. & Bhattacharya, S. 2007. "The new ext4 file system current status and future plans" [Online] Available from: http://www.kernel.org/doc/ols/2007/ols2007v2-pages-21-34.pdf [accessed 22 August 2010].

Moallem, M. 2008. A dissertation about the "performance evaluation of Linux I/O schedulers" [Online] Available from: http://gradworks.umi.com/MR/38/MR38892.html [accessed 18 December 2010].

Murray, J. & Vanryper, W. 1996. "Encyclopedia of Graphics File Formats"
2nd Edition, O'Reilly Media, ISBN-10: 1565921615
Online available at: http://www.fileformat.info/mirror/egff/index.htm
[accessed 7 November 2010].

Niranjan, N. 1996. A doctoral dissertation about the "File System Support for Multimedia Applications" [Online] Available from: http://portal.acm.org/citation.cfm?id=924911
[accessed 7 April 2010].

Ubuntu 2010,a. "Ubuntu Manpage: mkfs.xfs" [Online]
Available from: http://manpages.ubuntu.com/manpages/lucid/man8/mkfs.xfs.8.html
[accessed 20 November 2010].

Ubuntu 2010,b. "Ubuntu Manpage: mkfs.btrfs" [Online]
Available from: http://manpages.ubuntu.com/manpages/lucid/man8/mkfs.btrfs.8.html
[accessed 20 November 2010].

Ubuntu 2010,c. "Man Page for mke2fs - create an ext2/ext3/ext4" [Online]
Available from: http://manpages.ubuntu.com/manpages/karmic/man8/mkfs.ext2.8.html
[accessed 20 November 2010].

Park, S. et al. 2000. "Design and Implementation of the Parallel Multimedia File System Based on Message Distribution" [Online] Published as proceedings of the eighth ACM international conference on Multimedia, Available from: http://portal.acm.org/citation.cfm?id=376325 [accessed 21 August 2010].

Sato, T. 2007. "ext4 online defragmentation" [Online] Published as proceedings of the Linux Symposium 2007, Available from: http://www.linuxsymposium.org/archives/OLS/Reprints-2007/sato-Reprint.pdf [accessed 21 August 2010].

Scott, J. R. 2010. "Btrfs by default in Maverick" [Online] Available from: http://www.netsplit.com/2010/05/14/btrfs-by-default-in-maverick/ [accessed 21 August 2010].

Stephan, J. 2005. "Linux File Systems Comparative Performance" [Online] Available from: http://www.unisys.com/products/enterprise__servers/insights/insights__compendium/linux_file_systems_comparative_performance_white_paper_1-6-06.pdf [accessed 11 April 2010].

SUN 2004. "File System Performance: The Solaris OS, UFS, Linux ext3, and ReiserFS" [Online] Available from: http://www.sun.com/software/whitepapers/solaris10/fs_performance.pdf [accessed 13 April 2010].

SuSE 2005. "Large File Support in Linux" [Online] Available: http://www.suse.de/~aj/linux_lfs.html [accessed 22 August 2010].

Sweeney, A. et al. 1996. "Scalability in the XFS File System" [Online] Available from: http://www.usenix.org/publications/library/proceedings/sd96/full_papers/sweeney.txt [accessed 22 August 2010].

Tanenbaum, A. 2008. *Modern Operating Systems* (third edition) Person, ISBN-10: 0-13-600663-9.

Ven, A. 2010. "Btrfs as default file system" [Online] Available: http://lists.meego.com/pipermail/meego-dev/2010-May/002133.html [accessed 17 August 2010].

Wang, Z. & Crowcroft, J. 1996. "Quality of Service Routing for Supporting Multimedia Applications" [Online] Published on the IEEE journal Selected Areas in Communication 1996 Volume 14, Issue 7, page 1228 - 1234, Available from: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?reload=true&arnumber=536364 [accessed 21 August 2010].

APPENDICES

Appendix A. Raw data of Experiments with Default Settings

The following tables show the raw data results of the experiments. The average time was used in the diagrams in this document.

free size in bytes after format				
Filesystem	**Test 1**	**Test 2**	**Test 3**	**Average**
ext4	684,245,464	684,245,464	684,245,464	684,245,464
xfs	732,408,000	732,408,000	732,408,000	732,408,000
jfs	732,328,744	732,328,744	732,328,744	732,328,744
btrfs	732,571,944	732,571,944	732,571,944	732,571,944
reiserFS	732,516,796	732,516,764	732,516,764	732,516,775

Table 8: Free size on the Partition after Formatting

partition format time in seconds				
Filesystem	**Test 1**	**Test 2**	**Test 3**	**Average**
ext4	155.727	155.149	156.631	155.836
xfs	1.800	1.804	1.810	1.805
jfs	4.036	4.076	4.068	4.060
btrfs	0.037	0.038	0.051	0.042
reiserFS	41.877	45.119	45.006	44.001

Table 9: Partition Format Time

used size in bytes after copy job - picture library				
Filesystem	**Test 1**	**Test 2**	**Test 3**	**Average**
ext4	628,073,112	628,083,896	628,082,496	628,079,835
xfs	626,735,390	626,729,288	626,739,260	626,734,646
jfs	627,291,392	627,291,392	627,291,388	627,291,391
btrfs	630,023,720	629,530,644	629,531,484	629,695,283
reiserFS	627,099,332	627,099,332	627,099,332	627,099,332

Table 10: Used Size on Picture Library after Copy Job

used size in bytes after copy job - sound library				
Filesystem	**Test 1**	**Test 2**	**Test 3**	**Average**
ext4	628,794,376	628,796,756	628,797,132	628,796,088
xfs	628,561,020	628,564,212	628,564,460	628,563,231
jfs	628,674,916	628,674,912	628,674,912	628,674,913
btrfs	630,171,448	628,763,788	628,763,536	629,232,924
reiserFS	629,172,268	629,172,268	629,172,268	629,172,268

Table 11: Used Size on Sound Library after Copy Job

used size in bytes after copy job - video library				
Filesystem	**Test 1**	**Test 2**	**Test 3**	**Average**
ext4	627,756,216	627,762,424	627,765,512	627,761,384
xfs	627,584,684	627,585,768	627,586,360	627,585,604
jfs	627,649,404	627,649,404	627,649,404	627,649,404
btrfs	626,709,472	626,769,520	626,013,656	626,497,549
reiserFS	628,202,924	628,202,924	628,202,924	628,202,924

Table 12: Used Size on Video Library after Copy Job

used size in bytes after copy job - video-hd library				
Filesystem	**Test 1**	**Test 2**	**Test 3**	**Average**
ext4	628,185,936	628,185,936	628,185,936	628,185,936
xfs	627,984,592	627,984,632	627,985,683	627,984,969
jfs	628,041,392	628,041,392	628,041,392	628,041,392
btrfs	626,624,816	626,967,584	626,540,457	626,710,952
reiserFS	628,608,184	628,608,184	628,608,184	628,608,184

Table 13: Used Size on Video-hd Library after Copy Job

free size in bytes after copy job - picture library				
Filesystem	**Test 1**	**Test 2**	**Test 3**	**Average**
ext4	56,374,008	56,363,224	56,364,560	56,367,264
xfs	105,672,610	105,711,640	105,701,668	105,695,306
jfs	105,126,976	105,126,976	105,126,980	105,126,977
btrfs	102,548,280	103,041,356	103,040,516	102,876,717
reiserFS	105,450,272	105,450,272	105,450,272	105,450,272

Table 14: Free Size on Picture Library after Copy Job

free size in bytes after copy job - sound library				
Filesystem	**Test 1**	**Test 2**	**Test 3**	**Average**
ext4	55,652,744	55,650,364	55,649,988	55,651,032
xfs	103,879,908	103,876,716	103,876,468	103,877,697
jfs	103,743,452	103,743,456	103,743,456	103,743,455
btrfs	102,400,552	103,808,212	103,808,464	103,339,076
reiserFS	103,377,336	103,377,336	103,377,336	103,377,336

Table 15: Free Size on Sound Library after Copy Job

free size in bytes after copy job - video library				
Filesystem	**Test 1**	**Test 2**	**Test 3**	**Average**
ext4	56,690,904	56,684,696	56,681,608	56,685,736
xfs	104,856,244	104,855,160	104,854,568	104,855,324
jfs	104,768,964	104,768,964	104,768,964	104,768,964
btrfs	105,862,528	105,802,480	106,558,344	106,074,451
reiserFS	104,346,680	104,346,680	104,346,680	104,346,680

Table 16: Free Size on Video Library after Copy Job

free size after copy job - video-hd library				
Filesystem	**Test 1**	**Test 2**	**Test 3**	**Average**
ext4	56,261,184	56,261,184	56,261,184	56,261,184
xfs	104,456,336	104,456,296	104,454,960	104,455,864
jfs	104,376,976	104,376,976	104,376,976	104,376,976
btrfs	105,947,184	105,604,416	105,847,973	105,799,858
reiserFS	103,941,420	103,941,420	103,941,420	103,941,420

Table 17: Free Size on Video-hd Library after Copy Job

picture library copy time in seconds				
Filesystem	**Test 1**	**Test 2**	**Test 3**	**Average**
ext4	14,738.000	14,759.861	14,748.245	14,748.702
xfs	28,336.000	28,429.000	28,387.000	28,384.000
jfs	16,786.038	16,836.545	16,888.589	16,837.057
btrfs	13,570.000	13,509.952	13,575.214	13,551.722
reiserFS	16,988.206	16,931.714	17,269.571	17,063.164

Table 18: Copy Time for Picture Library

sound library copy time in seconds				
Filesystem	**Test 1**	**Test 2**	**Test 3**	**Average**
ext4	9,949.978	9,844.346	9,873.449	9,889.258
xfs	11,428.923	11,402.618	11,532.025	11,454.522
jfs	10,707.936	10,652.439	10,767.000	10,709.125
btrfs	8,684.166	8,531.000	8,627.173	8,614.113
reiserFS	10,612.000	10,617.960	10,624.308	10,618.089

Table 19: Copy Time for Sound Library

video library copy time in seconds				
Filesystem	**Test 1**	**Test 2**	**Test 3**	**Average**
ext4	8,180.843	8,267.000	8,126.973	8,191.605
xfs	8,514.744	8,393.000	8,359.354	8,422.366
jfs	8,836.663	8,855.346	8,817.229	8,836.413
btrfs	7,922.740	7,941.000	7,921.000	7,928.247
reiserFS	9,120.280	9,106.000	9,140.234	9,122.171

Table 20: Copy Time for Video Library

video-hd library copy time in seconds				
Filesystem	**Test 1**	**Test 2**	**Test 3**	**Average**
ext4	7,911.000	7,868.000	7,923.733	7,900.911
xfs	8,401.000	8,425.381	8,273.680	8,366.687
jfs	8,253.638	8,252.704	8,266.999	8,257.780
btrfs	8,019.130	8,000.013	8,010.890	8,010.011
reiserFS	9,129.000	9,139.824	9,132.755	9,133.860

Table 21: Copy Time for Video-HD Library

file system check time in seconds - picture library				
Filesystem	**Test 1**	**Test 2**	**Test 3**	**Average**
ext4	196.727	203.356	201.570	200.551
xfs	131.798	130.063	130.649	130.837
jfs	533.007	530.535	529.530	531.024
btrfs	88.146	89.125	89.123	88.798
reiserFS	1,365.256	1,364.144	1,366.000	1,365.133

Table 22: File System Check Time for Partition with Picture Library

file system check time in seconds- sound library				
Filesystem	**Test 1**	**Test 2**	**Test 3**	**Average**
ext4	20.890	21.621	20.643	21.051
xfs	29.227	29.830	32.016	30.358
jfs	110.141	103.385	103.584	105.703
btrfs	32.799	32.172	33.133	32.701
reiserFS	889.000	896.239	894.545	893.261

Table 23: File System Check Time for Partition with Sound Library

file system check time in seconds - video library				
Filesystem	**Test 1**	**Test 2**	**Test 3**	**Average**
ext4	19.647	18.321	20.288	19.419
xfs	5.454	5.542	5.557	5.518
jfs	12.410	12.400	12.447	12.419
btrfs	28.937	29.174	28.950	29.020
reiserFS	866.514	864.779	905.650	878.981

Table 24: File System Check Time for Partition with Video Library

file system check time in seconds - video-hd library				
Filesystem	**Test 1**	**Test 2**	**Test 3**	**Average**
ext4	11.701	11.801	11.802	11.768
xfs	2.610	2.174	2.629	2.471
jfs	3.680	3.704	3.705	3.696
btrfs	25.591	25.647	25.620	25.619
reiserFS	886.000	885.356	885.880	885.745

Table 25: File System Check Time for Partition with Video-HD Library

picture library delete time in seconds				
Filesystem	**Test 1**	**Test 2**	**Test 3**	**Average**
ext4	585.880	563.356	579.870	576.369
xfs	6,044.325	6,053.073	6,028.012	6,041.803
jfs	2,002.065	1,964.384	1,978.798	1,981.749
btrfs	338.726	336.111	335.645	336.827
reiserFS	2,521.723	2,452.766	2,483.480	2,485.990

Table 26: Delete Time for Picture Library Content

sound library delete time in seconds				
Filesystem	Test 1	Test 2	Test 3	Average
ext4	170.340	173.621	165.726	169.896
xfs	344.468	344.331	342.072	343.624
jfs	531.478	532.385	529.584	531.149
btrfs	53.110	52.248	53.137	52.832
reiserFS	1,473.680	1,508.676	1,501.545	1,494.634

Table 27: Delete Time for Sound Library Content

video library delete time in seconds				
Filesystem	Test 1	Test 2	Test 3	Average
ext4	29.377	31.528	32.740	31.215
xfs	27.779	27.790	27.879	27.816
jfs	15.017	14.951	15.880	15.283
btrfs	61.394	60.130	60.844	60.789
reiserFS	918.025	908.429	905.650	910.701

Table 28: Delete Time for Video Library Content

video-hd library delete time in seconds				
Filesystem	Test 1	Test 2	Test 3	Average
ext4	16.876	16.719	17.050	16.882
xfs	0.163	0.174	0.201	0.179
jfs	0.505	0.549	0.348	0.467
btrfs	48.908	49.326	49.450	49.228
reiserFS	903.714	909.086	908.120	906.973

Table 29: Delete Time for Video-HD Library Content

file creation time - test 1									
Filesystem	**1GB**	**2GB**	**4GB**	**8GB**	**16GB**	**32GB**	**64GB**	**128GB**	**256GB**
ext4	9.053	18.199	36.359	72.598	146.734	297.565	639.884	1389.11	3224.27
xfs	10.316	20.529	41.66	83.58	171.265	336.833	726.347	1572.11	3544.62
jfs	8.895	17.992	35.859	71.396	145.828	289.843	626.954	1361.47	3123.04
btrfs	9.159	17.875	36.506	71.544	145.554	294.33	633.305	1367.97	3123.16
reiserFS	9.297	18.465	39.771	82.299	162.299	323.104	704.696	1514.66	3485.61

Table 30: Time for Creating File with Specific File Size

file creation time - test 2		
Filesystem	1MB	50GB
ext4	7,724.55	4,621.04
xfs	15,206.03	5,525.15
jfs	8,809.14	4,805.27
btrfs	7,010.79	4,592.15
reiserFS	7,352.60	6,412.02

Table 31: Time for Filling a Library with Files of a Specific Size

Appendix B. Raw Data of Experiments with Tuned Settings

The following tables show the raw data results of the experiments with tuned settings.

tuned format time ext4			
formatting parameters	Test 1	Test 2	Average
default	148.024	148.146	148.085
-m 0	148.321	147.961	148.141
-m 0 -N 3000000	17.311	17.351	17.331
-m 0 -N 3000000 -E lazy_itable_init=1	4.866	5.066	4.966

Table 32: ext4 Format Time with Tuned Settings

tuned xfs video-hd library tasks						
xfs	format time/s	copy time/s	used size	free size	file system check/s	delete time/s
Test 1 default	1.805	8,366.687	627,984,969	104,455,864	2.471	0.179
Test 2 64 MB	0.983	8,242.115	627,985,132	104,521,332	2.627	0.169
Test 3 128 MB	1.852	8,471.446	627,986,348	104,454,580	2.661	0.157
Test 4 128 MB logbuf=8	1.818	8,480.246	627,985,100	104,455,828	2.637	0.178

Table 33: Video-hd Library Tasks on Tuned XFS Partition

tuned xfs picture library tasks						
xfs	format time/s	copy time/s	used size	free size	file system check/s	delete time/s
Test 1 default	1.805	28,384.000	626,734,646	105,696,306	130.837	6,041.803
Test 2 64 MB	1.036	27,170.978	626,729,116	105,777,348	129.396	5,937.396
Test 3 128 MB	1.713	27,015.764	626,737,796	105,703,132	128.420	5,915.042
Test 4 128 MB logbuf=8	1.899	26,864.172	626,763,692	105,677,236	128.796	5,765.764

Table 34: Picture Library Tasks on Tuned XFS Partition

tuned jfs sound library tasks						
xfs	format time/s	copy time/s	used size	free size	file system check/s	delete time/s
Test 1 deadline scheduler	3.768	10,572.732	628,674,912	103,743,456	103.052	545.425
Test 2 cfq scheduler	4.172	10,317.511	628,674,916	103,743,452	103.255	484.003

Table 35: Sound Library Tasks on Tuned JFS Partition

tuned ext4 picture library tasks						
ext4	format time/s	copy time/s	used size	free size	file system check/s	delete time/s
Test 1 default	155.836	14,748.245	628,079,835	56,367,264	201.570	576.369
Test 2 optimized	5.836	15,323.566	628,082,276	103,681,524	221.406	2,452.406

Table 36: Picture Library Tasks on Tuned ext4 Partition

tuned ext4 video-hd library tasks						
ext4	format time/s	copy time/s	used size	free size	file system check/s	delete time/s
Test 1 default	155.836	7,900.911	628,185,936	56,261,184	11.786	17.051
Test 2 optimized	5.203	7,927.893	628,185,936	103,577,864	11.162	17.312

Table 37: Video-hd Library Tasks on Tuned ext4 Partition

tuned reiserfs sound library tasks						
reiserfs	format time/s	copy time/s	used size	free size	file system check/s	delete time/s
Test 1 default	44.010	10,618.089	629,172,268	103,377	893.261	1,494.634
Test 2 journal=writeback	40.696	10,659.000	629,172,268	103,377,336	1560.921	1,433.066
Test 3 journal=writeback deadline scheduler	39.090	11,056.813	629,172,268	103,377,336	1617.437	1,421.410

Table 38: Sound Library Tasks on Tuned ReiserFS Partition

Appendix C. HARD DISK PERFORMANCE

	1988	2008	Increase
CPU Performance	1 MIPS	16,800 MIPS	16,800 x
Memory Device Density	128K	2GB	16,000 x
Disk Drive Performance	60ms	5.3ms	11 x

Table 2: HDD Performance Has Not Kept Pace with Other System Components[4]

Figure 22: Hard Disk Performance, Figure Source: Helba (2009)

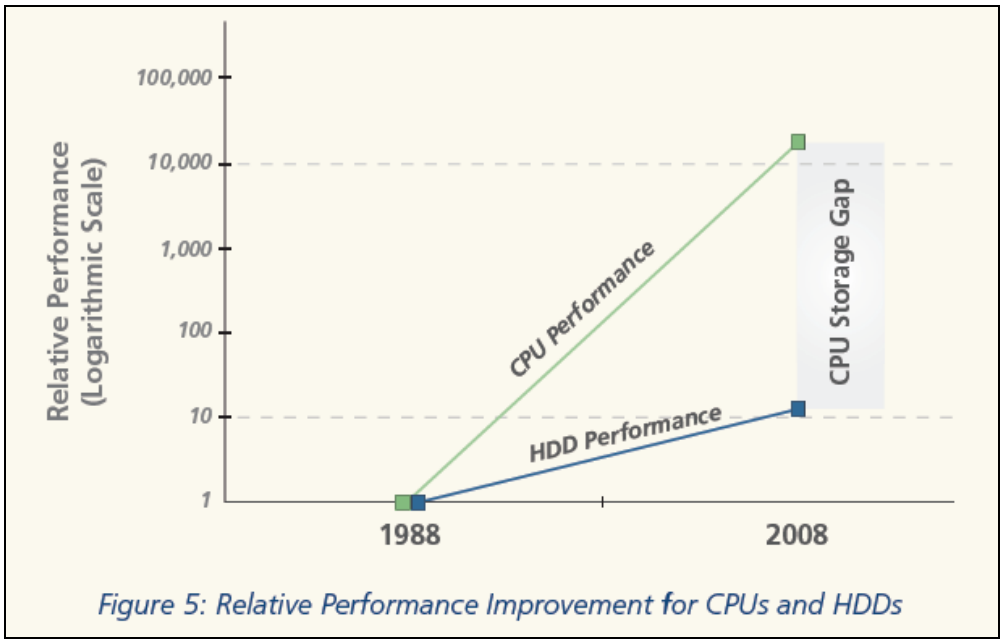


Figure 23: Relative Performance Improvements, Helba (2009)

Appendix D. MULTIMEDIA STORAGE SPACE REQUIREMENTS

Storage space requirements for uncompressed digital multimedia data.

Media Type (specifications)	Data Rate
Voice quality audio (1 channel, 8 bit samples at 8kHz)	64 Kbits/sec
MPEG encoded audio (equivalent to CD quality)	384 Kbits/sec
CD quality audio (2 channels, 16 bit samples at 44.1 kHz)	1.4 Mbits/sec
MPEG-2 encoded video	0.42 MBytes/sec
NTSC quality video (640 X 480 pixels, 24 bits/pixel)	27 MBytes/sec
HDTV quality video (1280 X 720 pixels/frame, 24 bits/pixel)	81 MBytes/sec

Figure 24: Data Rate for Uncompressed Multimedia Data, Gemmell et al. (1996)

Impact of scan resolution on the amount of memory needed to save the file.

Scan Resolution	6x4 inch Image Size (pixels)	Pixel Count	Memory size in bytes		
			24 bit RGB Color	8 bit Grayscale	Line art
75 dpi	450x300	135,000	405,000	135,000	16,875
150 dpi	900x600	540,000	1,620,000	540,000	67,500
300 dpi	1800x1200	2,160,000	6,480,000	2,160,000	270,000
600 dpi	3600x2400	8,640,000	25,920,000	8,640,000	1,080,000
1200 dpi	7200x4800	34,560,000	103,680,000	34,560,000	4,320,000
2400 dpi	14400x9600	138,240,000	414,720,000	138,240,000	17,280,000
4800 dpi	28800x19200	552,960,000	1,658,880,000	552,960,000	69,120,000
9600 dpi	57600x38400	2,211,840,000	6,635,520,000	2,211,840,000	276,480,000

Figure 25: Scan Resolution, Fulton (2010)

This chart gives an overview of digital camera resolutions

Digital Camera Resolution Chart								
Capture Resolution	Video Display*	Print Size***						
		2x3"	4x5"/4x6"	5x7"	8x10"	11x14"	16x20"	20x30"
320x240	Acceptable	Good	Acceptable	Poor	Poor	Poor	Poor	Poor
640x480 - 0.3 Megapixel	Good	Excellent	Good	Poor	Poor	Poor	Poor	Poor
800x600	Excellent	Photo Quality	Very Good	Acceptable	Poor	Poor	Poor	Poor
1024x768	Excellent	Photo Quality	Excellent	Good	Acceptable	Poor	Poor	Poor
1280x960 - 1 Megapixel	Excellent	Photo Quality	Photo Quality	Very Good	Good	Poor	Poor	Poor
1536x1180	Excellent**	Photo Quality	Photo Quality	Excellent	Very Good	Acceptable	poor	poor
1600x1200 - 2 Megapixel	Excellent**	Photo Quality	Photo Quality	Photo Quality	Very Good	Acceptable	Acceptable	Poor
2048x1536 - 3 Megapixel	Excellent**	Photo Quality	Photo Quality	Photo Quality	Excellent	Good	Acceptable	Acceptable
2240x1680 - 4 Megapixel	Excellent**	Photo Quality	Photo Quality	Photo Quality	Photo Quality	Very Good	Good	Acceptable
2560x1920 - 5 Megapixel	Excellent**	Photo Quality	Photo Quality	Photo Quality	Photo Quality	Excellent	Very Good	Very Good
3032x2008 - 6 Megapixel	Excellent**	Photo Quality	Photo Quality	Photo Quality	Photo Quality	Photo Quality	Excellent	Very Good
3072x2304 - 7 Megapixel	Excellent**	Photo Quality	Photo Quality	Photo Quality	Photo Quality	Photo Quality	Excellent	Excellent
3264x2448 - 8 Megapixel	Excellent**	Photo Quality	Photo Quality	Photo Quality	Photo Quality	Photo Quality	Photo Quality	Excellent
10 Megapixel +	Excellent**	Photo Quality	Photo Quality	Photo Quality	Photo Quality	Photo Quality	Photo Quality	Photo Quality

Figure 26: Digital Camera Resolution Chart, B&H (2010)