THE HIXOSFS MUSIC APPROACH VS COMMON MUSICAL FILE MANAGEMENT SOLUTIONS

Nicola Corriero, Vittoria Cozza and Fabrizio Fattibene
Department of Computer Science, University of Bari, Via Orabona 4, Bari, Italy

Keywords: Operating system, Linux, Filesystem, System calls, FUSE, Multimedia.

Abstract: *Hixosfs* music is an extension of ext2 Linux filesystem, with additional features to easy accessing and categorizing musical files. Specially it extends the inode struct inside the Virtual file system, considerating as file proprieties meta information such as album, author, title related to the content of a musical file. Comparition have been done respect to the Linux file system in user space *Musicmeshfs*, the Linux ext2 *xattr* feature, and ad hoc user space programs for efficiently retriving multimedia data. Since *Hixosfs music* manages the musical tags at kernel level, it offers higher performances then the other solutions but with less flexibility.

1 INTRODUCTION

In this work we present and compare different way to manage at file system (fs) level musical file collection of various format. We address both the problems of store inside a folder tree a musical collection in a configurable way with *MusicMeshFS*(Corriero, 2008) and *xattr* and the problem of access in a fast way the musical tag (e.g. author, album) values with *Hixosfs music*. Manage musical files in a Linux fs makes possible to use the shell tools as *find*, beside of ad hoc commands for specific musical fs that keep under consideration metadata. Generally speaking GNU/Linux fs are based on a common kernel layer called Virtual File System (VFS) then they keep the same logical structure and are all presented at the user in the same way. At this level structs and operations common to all fs types have to be defined. To implement a new fs means to know the kernel internals, learning kernel module programming and either to know the VFS architecture to modifies its structs. For example *Hixosfs music*, respect ext2 fs, extends the inode struct to include musical tags in the metainformations associated to a file.
Linux offers already a way to extend inode attributes but without physically modifying the inode struct with the *xattr* feature. This solution is currently used in most common Linux fs and it offers an easy interface to the final user that can choose and add runtime all the couples attribute-name attribute-value he likes to store with the command *attr*. Finally an other way to manage tags in fs is by writing a new fs in userspace, this can be done even from not kernel developers, thanks the the userspace library offered by Filesystem in Userspace (FUSE) project(Szeredi, 2008). In this way an user can ridefine the file operations keeping under consideration metainformation related to a file, and to have a different virtual view of his fs that anyway is not physically modified. MusicMeshFS (MMFS) is an example of fs based on fuse for multimedia data.

2 HIXOSFS MUSIC

Hixosfs music architecture(Corriero, 2009) generally recalls the widely used ext2 file system, but it extends the information stored in the inode with tags such as author, title, year and album for musical data. The *hixosfs* project required the linux headers to be modified, new system calls for read and write tags and the new hixosfs module to be implemented: *chtag* and *retag*, the final user recall the syscall by the tools *chmusic* and *statmusic* with analogous functionality of the well known chown and stat but obviously considerating musical tags(hix). The header "*linux/fs.h*" has been changed to extend the definition of iattr struct with musical tag. Specially the function `setattr` with the task to propagate at the inode the change related to iattr struct, has been extended specifying the new tags definition that will be stored inside the inode.
The inode is the kernel struct for all the file type man-

agement, *hixosfs music* extends the inode definition with a struct tag.

The struct tag has four fields for a total of about 100 byte of stored information, theoretically an i-node can be extended until 4 kb then it's possible to customize it with many tags for your purpose. It's convenient to choose tags that are most of the time used in the file search to di-scriminate the files depending their content. We chose here what was able to maximize the time of search musical files by most commonly used criteria as album or author name and so on.

A new module have been created, the *Hixosfs module* that includes *hixosfs* operations definition. Certain functions as *hixosfs_read_inode*, *hixosfs_update_inode* and *hixosfs_new_inode* don't differ too much respect to read and update for ext2 file system. In addition there is a new part for the management of the new content based file attribute in the struct *iattr* and *inode*. Finally two new system calls *Chtag* and *retag* have been implemented to allow to write and read the struct tag from hixos_inode.

All described modification have been done starting from 2.6.23 kernel, but can easily integrated in newer kernel versions. Beside standard user mode tools (stat, chmod, ls and so on), specific command are needed to directly handle with new file tags. There are two kind of user space programs hixosfs requires, to read/write tags from/to inodes and as well to interface with the tag based file management offered by hixosfs. At the first group belong programs as `statmusic` (think at shell command `stat`) and `chmusic` (think at `chown` and so on), at the second `orderby`, `ls`, `find`, with an intuitive meaning.

3 EXTENDED ATTRIBUTES

Currently Linux fs as ext2 (Remy Card), ext3, reiserfs allows to manage with metainformation related to a file with *xattr* feature. Patching the kernel with xattr you have a way to extend inode attributes that doesn't physically modify the inode struct. This is possible since in *xattr* the attributes are stored as a couple attribute-value out of the inode as a variable length string. Generally the basic command used to deal with extended attributes in *Xattr* is *`attr`* that allows to specifies different options to set and get attribute values, to remove attributes to list all of them and then to read or writes these values to standard output. The programs we implemented in our testing scenario are based on this user space tool.

4 FILE SYSTEM IN USERSPACE

FUSE is an open source project with GPL and LGPL license that offers a Linux kernel module to allow to not privileged users to create their own fs without write kernel level code, in fact customizing fuse you can create your own fully functional fs directly in user space. This approach is used to write virtual fs, that don't care about store data on the disk but of arranging them to offer a virtual view of such data to the final user. For the MMFS design please refers to (Corriero, 2008).

5 PERFORMANCE MEASURES

In this section performance measurements made on hixosfs are presented. The monitored operation are about reading e writing tags from and to a file, reading from disk the folders content, the operation of ordering a group of files with user passed parameters. The time required to perform this operations by hixosfs has been measured and compared with the time needed in the some situation by:

- a fuse based file system approach, specially musicmeshfs (MMFS)
- xattr Linux fs feature
- an ad hoc user space program

All analyzed case have in common a musical files processing phase application based on TagLib C(Wheeler, 2008) library to extract musical tags such as author, track, title year, album. The time itself has been measured by the unix command *`time command1`* that gives as output the execution times of input command command1. As you can expect the study shows that performing such operations with hixosfs gives an huge optimization in term of time since it works in kernel mode with operating system privileges. Every action done for the discovery of information involves a system call and then a software interrupt for each system call.

To work with *hixosfs music* and related user mode tools you have to to patch and recompile the kernel source, to compile user space programs, to create a hixosfs fs (ext2-modified) with a 256 byte inode by, e.g, `mkfs.ext2 -I 256 ...` and finally to mount hixosfs fs in your tree To test his way of working and then do performance comparisons, after preparing the system, we populated the fs with a collection of 80 Gb of musical data. The main command used to add files inside the *hixosfs music* is addmusic and the sintax is:

```
addmusic FILENAME | FOLDERNAME
```

By using this command you can find the *hixosfs music* already mounted and copy music file inside it. Otherwise it's possible to use `cp` to physically copy files in the partition and then `filltag` to populate the hixosfs_inode tags struct. The usage is:

```
filltag FILENAME
```

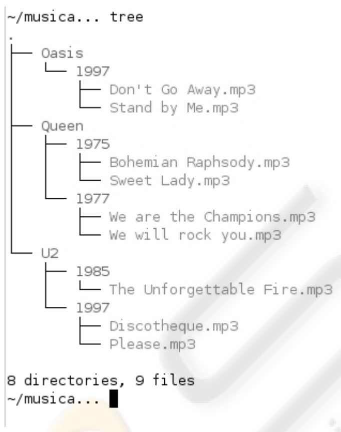
The command open file with taglib function to find metainformation about music and uses a system call to insert this information in the inode. Since it uses a system call the operation is a high priority kernel mode task. Once fs contains musical files, they can be ordered by an ad hoc program called orderby. The syntax is:

```
orderby [-a tag1 | -b tag2]
```

For example with the command:

```
#orderby author year
```

we have all file in fs in a tree with the first node as a folder with as name each track author, the second node as a folder with as name the track year and in the third node, finally, the music file itself. The trick of using hard links for implementing `orderby` allows coexisting in safe way more then one order.



```
~/musica... tree
.
├── Oasis
│   └── 1997
│       ├── Don't Go Away.mp3
│       └── Stand by Me.mp3
├── Queen
│   ├── 1975
│   │   ├── Bohemian Raphsody.mp3
│   │   └── Sweet Lady.mp3
│   └── 1977
│       ├── We are the Champions.mp3
│       └── We will rock you.mp3
└── U2
    ├── 1985
    │   └── The Unforgettable Fire.mp3
    └── 1997
        ├── Discotheque.mp3
        └── Please.mp3

8 directories, 9 files
~/musica... █
```

Figure 1: `orderby`.

5.1 Musicmeshfs vs Hixosfs

Musicmeshfs is an example of file system based on the fuse library. The concept of orders is common at both hixosfs and MMFS. In the case of MMS the string used at mounting fs time, includes a reference to a particular required orders. For example mounting with the option Author#Year creates a three levels tree: first level with author of the tracks, second year of the tracks in the database for each author, then the track files themself. The MMFS implementation includes a database that stores the metainformation and the data organization according user choose. In fact in general a virtual fs never changes the real files, but it just shows them at the user according a chose order. Since MMFS is a kind of read-only fs, the only access you can do to write in fact is only to change a file tag, a command that seems interesting to perform to do a comparison with *hixosfs* is the common command that shows the files according predefined order such `ls`. In the simple case of a fs, let's say with folder arranged by author (that implies one folder for one author), by executing `ls`, MMFS executes the command as a run time query (select ...) to the db of the information related to the author and show them in a file system fashion. MMFS is high customizable in fact it's possible to query the database in a configurable and easy way thanks MMFS language, but the access to a db is not costless. From one other side *Hixosfs* doesn't query a db to search for metadata but keeps them in the inode itself can move inside the folders with ls command faster as in table 1.

Table 1: Hixosfs vs mmfs.

`ls` in MMFS	`ls` in hixosfs
13.10 sec	0.20 sec

5.2 Xattr vs Hixosfs

As introduced earlier, Xattr is a Linux feature to support extended file attributes. To use Xattr means to compile the kernel with the support of such tags, to create the fs with extended attributes and mount the fs with *mount -o remount,user_xattr what where*, then to use ad hoc programs to read and write the extended attributes. To deal with extended attributes we used `attr` that allows to specifies different options to read and write attribute values; The way we used this command is like in the following example:

- `attr -s` *autore* `-V` *Valore /mnt/Musica/mia_musica.ogg*: creates the couple (autore, Valore) for the musical file musica.ogg;
- `attr -g` *autore /mnt/Musica/mia_musica.ogg*: allows to read the couple of values associated to the attribute autore;
- `attr -l` */mnt/Musica/mia_musica.ogg*: allows to read all the couple attribute values;

To populate the fs with extended attributes we created a read_Xattr script that recall `attr -s` command to associate to each file of our testset the following attributes: author, album, title, year. Then `getXattr` script to read the attributes.

It seems relevant to compare now the operation of extraction extended attributes thanks getXattr with the *hixosfs statmusic*. This command is similar to the

Table 2: Hixosfs vs xattr.

`getXattr *`	`statmusic *`
29.725 sec	2.987 sec

the standard Unix user mode tool `stat`, specially the output of `stat` is the list of tags content for each input file, while in the case of `statmusic` musical tags are showed too:

```
# statmusic Bohemian Rhapsody.mp3
File Name: Bohemian Rhapsody.mp3
Author: Queen
Title: Bohemian Rhapsody
Album: A Night at the Opera
Year: 1975
```

The time comparison result are in the table 2. As you expect since the way hixosfs and xAttr physically store and access tags, `statmusic` gives better performances.

5.3 Open_Read vs Hixosfs

`Open_Read` is an ad hoc program that open a music file, extract tags and then it shows them on the screen:

```
TagLib_File* tlf = taglib_file_new(argv[i]);
TagLib_Tag* tlt = taglib_file_tag(tlf);
printf("\nAlbum: %s", taglib_tag_album(tlt));
```

`Open_read` uses the taglib library that works with the standard system call *fopen or/and lseek* and print on the screen with printf C standard function. We compared the use of Open_read with the use of `statmusic` in hixosfs that recalls the system call *retag* and read the tags directly from the inode and shows them on the screen. Comparing the time spent by both approach we had less then 1 second (0.03 sec) needed from `statmusic`, respect 4.19 minutes required in the case of the ad hoc program.

6 CONCLUSIONS

This work comparison analysis shows that the faster way to handle musical metadata, in term of time spent to deal with, is when the tags management happens at kernel level: the case of *hixosfs* and *xattr* feature.
Hixosfs music appears to be an innovative approach to manage musical file tags at fs kernel level and it gives higher performances when dealing with an huge number of files. If we think of a partion formatted hixosfs that collect only musical data, hixosfs appears the best performant solution. The huger advantage in time in the case of *hixosfs* is because tags are stored inside the inode and they are accessed by high priority kernel space calls: the system calls. From the operating system efficiency side a further disavantage is an intuitive memory lost. In fact *hixosfs music* allocates for each file memorization in the *hixosfs* formatted partition, a 256 byte inode even if the file has to be tagged with no attributes so better if we use hixosfs only for a whole partition for musical data or as the fs for embedded devices as ipod or mp3reader.// From user side the problem is that the tags are fixed at fs design time even if here we chose the most common tags for indexing musical data such as author, album, year, title, that we extracted from musical file since the use of taglib. As further development we imagine to extend the inode with different kind of tags chosen from the final user and to implement user space commands that can allow to the user to copy the tag in the inode to the tag in the file and the opposite easily. In the case of working with a fs in userspace as MMFS or using xattr feature they had lower performances in term of time but they offer high flexibility for tag selection and management. A further work can be to extend fuse library to make it to offer an abstraction layer over *hixosfs music*. *Hixosfs* design also can be enriched to became a network fs well suited for the musical files sharing and to simplify the indexing of files inside the network.

REFERENCES

Hixosfs file system. http://www.di.uniba.it/hixos/hixosfs/index.html. home page.

Corriero, C. Hixosfs_music: a filesystem in linux kernel space for musical files. MMEDIA 2009. home page.

Corriero, Cozza, D. t. Z. (SIGMAP 2008). A configurable linux file system for multimedia data.

Remy Card, Theodore Ts'o, S. T. http://e2fsprogs.sourceforge.net/.

Szeredi, M. (2008). Fuse. http://fuse.sourceforge.net. Home page.

Wheeler, S. (2008). Taglib. http://developer.kde.org/wheeler/taglib.html. Home page.