Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 5 (2011) 765–770

Procedia Computer Science

International Symposium on Frontiers in Ambient and Mobile Systems (FAMS)

IFHNFS: fast indexing information in mobile devices

Nicola Corriero and Giuseppe Gargiuolo and Giovanni Pani

University of Bari
Department of Computer Science
Via Orabona 4, 70125, Bari, Italy
ncorriero,pani@di.uniba.it; giuseppegargiuolo@gmail.com

Abstract

To track movements of objects and people we need to use expensive technologies and complex softwares. Usually these softwares need lots of memory. There are many examples of objects and people of which would be useful to know the precise locations. Something sent via pony express, position of buses in the city, friends' locations, and so on. Our purpose is to use connectivity of smartphone to share position via gps by using only a filesystem created ad hoc without databases or user space programs. IFHNFS is based on Hixosfs and NFS (Linux kernel). Indexing the most relevant data directly in the fs structure makes the exchange of data Faster and, via NFS, potentially available whenever and whenever. Ad-hoc solutions for Android have been developed in real contexts. The system have been tested on a network of Android OS devices that share data with a Linux server.

Keywords: Filesystem, Android, gps, wireless, ad-hoc network

1. Introduction

Android is an operating system for smartphones based on the Linux kernel. Smartphones often have a built-in GPS receiver signal. Indeed, there are many applications that provide informations on such sites or activities business in the vicinity of the device. The tracking of objects or persons arising from the use of expensive technology and complicated software that take up space and memory devices in an attempt to minimize power consumption (cellular phones). There are dozens of examples of people or things of which would be useful to know the real time precise location. Parcels sent via pony express, the position of the bus in the city, location of friends, ... GPS receivers do not allow you to send signals. Our proposal connectivity is to use smartphone to send GPS signals. The innovation lies in not using any database or user space program, but only a filesystem created ad hoc. Ifhnfs is a filesystem build to merge the advantages of Hixosfs [2] and Nfs [1]. Hixosfs allow us to improve the file content search by using hixos tag, a simple 128 byte C struct stored inside each inode. Nfs is a common network filesystem to share folders or device within a network. We tested our idea in a real testing scenario. We build two applications to send gps informations from devices throws ifhnfs file and to read from a simple web pages the last and historical location. Finally we tested our idea with usually similar scenario such as ext2 with xattr and common databases.

Selection and/or peer-review under responsibility of Prof. Elhadi Shakshuki and Prof. Muhammad Younas.
doi:10.1016/j.procs.2011.07.102

2. Database vs Filesystem

Normally these situations are handled using PCs with large primary and secondary memories that make possible the use of every operating system and every mean for saving and managing information.
Other approaches of the system require the use of complex databases over servers and/or embedded databases over embedded machines.

2.1. Oracle/Mysql

The tools used to handle large quantities of data are very efficient although they require large resources in hardware and software. Installing and executing of applications like Oracle or Mysql, in fact, require large quantities of memory and hard disk space.

2.2. Sqlite

In the embedded systems we have evident problems of memory that during the time have solicited light-weight and high-performance ad-hoc solutions. Sqlite is an application that implements all the functionalities of a database using simple text files. This enlighten the execution load of the system and facilitates the integration of the system inside an embedded system. However, the system installation produces a certain load to the mass memory.

2.3. Extended attributes

Currently Linux fs as ext2 [6], 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.

2.4. Standard NFS

L'ultimo caso da analizzare è usare l'approccio standard di un normale filesystem NFS senza aggiunta di alcuna patch. Nella sezione test saranno evidenziate le differenze rispetto a tale situazione. Il network filesystem permette di montare cartelle remote all'interno del proprio filesystem facilitando le operazioni sui file remoti.

3. IFHNFS

The system we propose is based on NFS. Network File System is a Linux filesystem that makes possible file sharing within a network. The filesystem uses a server where is possible to configure the hosts that can work over the single folders exported. A network file system allows users to mount partitions on remote machines available on the same network on even in other networks making the user work on that file system in the same way as he usually works locally. More versions of NFS are available to users. The earliest work on TCP protocols and support no autenthication protocols, dispiting the newer version (NFSv4) which works on Kerberos making connection fast and secure. Interesting is the way more clients can, via NFS, connect to a server to store data, centralizing large quantities of information that can be available from anybody for further uses. Each folder of our system is dedicated to an antenna-client for sharing detected data. In booting phase the embedded system mounts the remote folder and uses it for data backup every thirty minutes. A script runs on the server to verify periodically the presence of backup data and to import data in the database.

IFHNFS takes the power of treating remote data locally even securing connections and combines it with the speed of retrieving informations in a fast way.

Tests have shown centralizing data in a IFHNFS filesystem is faster than using a common database like mysql or SqlLite. From the application side, IFHNFS has been tested in postal environments implementing a real-time tracking system for packs all over the world.

Here is a structural description about how our new filesystem works. Properly IFHNFS is a NFS filesystem extension which adds 4 tags to the inode structure of a resource (file, directory, etc.) to make data retrieval available to user who don't need to open the entire file to get particular information.

To make kernel read these new tags, HIXOS group worked on some utilities who write data in the new structure and read data from it, respectively called *chtag* and *retag*. These user space applications refer to some system calls suitably added in the Virtual File System located in kernel space.

4. System architecture

An ad-hoc filesystem created inside the kernel will be used to handle the data provided by the network. Each networks node has a ifhnfs client.

4.1. Android

Android[13] is a software stack for mobile devices that includes an operating system, middleware and key applications. The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language.

Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.

4.2. Hixosfs

Hixosfs is as an ext2 Linux filesystem (fs) extension able to classify and to manage metadata files collections in a fast and high performant way. This is allowed since high priority is given to the storing and retrieving of metadata respect tags because they are managed at fs level in kernel mode.
The *hixosfs* core idea is that information regarding the content of a metadata of a file belong to the fs structure itself. Linux fs in general stores common information for a type of file, such as permissions or records of creation and modification times, then the struct to represent a file inside the VFS called inode keeps information about such data. To represent tags physically inside the inode, *hixosfs* reserves a greater amount of memory for the inode to store extra information needed to label a file respect its content such as *album, author, title, year* for music file.
In this paper we explain how the fundamental concepts implemented in *hixosfs* can help to solve problems in embedded systems.

We had to implement two system calls to write and read the new information stored inside the inode: *chtag* and *retag*, the final user recall the syscall by the tools *chtag* and *stattag* with analogous functionality of the well known chown and stats but obviously considering generics tags. Finally we present others user space programs, for example you need an ad hoc command to populate the fs starting from a collection of metadata, the command that we call *addtag* can work to extract author and so on from mp3 file to fill the inode.

Hixosfs has been used to tag gps files and mobile devices. In this way all the load has been transfered to the kernel that handles and organizes the hixosfs files as occurrs. The servers and the clients contain partitions that can read and set the hixosfs tags so to manage the database.

The kernel struct for all the file type management is the **inode**.

The struct *tag* has four fields for a total of about 100 byte of stored information, theoretically an inode 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 discriminate the files depending their content. We choose here what was able to maximize the time of search gps files by most commonly used criteria as courier, tracker or data.

For such a reason we decided to use a generic version of hixosfs with a generic structure in which is possible to insert file representative tags case by case.

```
struct tag {
#ifdef CONFIG_HIXOSFS_TAG
char tag1[28];
char tag2[28];
char tag3[28];
char tag4[28;
unsigned int tag_valid;
#endif
}
```

In our partition there was only a hixosfs filesystem mounted in RAM with folders and files containing files provided by process.

Two user space programs have been written to read and write hixos tag. `retag` is used to read hixos tag stored inside ifhnfs file, while `chtag` is used to write a hixos tag field of ifhnfs file.

An example of file handling.
`chtag -c` sets the courier of the device.
`chtag -n` sets the tracking of the device.
`chtag -l` sets latitude of the device.
`chtag -o` sets longitude of the device.

```
$ chtag -c XDA124D test.gps
$ chtag -t KZ000729695IT test.gps
$ chtag -l 12/12/09
$ retag test.gps
courier: XDA124D
tracking: KZ000729695IT
latitude: 45.46369
longitude: 9.18814
$
```

Hixosfs use also other ad-hoc user space programs to order data or to looking for a tag value. `orderby`, in fact is used to create a tree of folders to order data in each folder/tag. The idea is to associate a folder to a tag value. In this way all file with a particular tag value “x“ are stored inside a folder ”x“. `findtag` is a user space program write to search all file with a given input value.

So the idea is that periodically a demon process read data from gps antenna and puts informations inside hixos tag by `chtag`. Before changing data in hixos tag, our process append last position to the file content. In this way we can read last position by `retag` (and so from hixos tag) and all historical positions from the content of each file.

5. Testing Scenario: Hlocator

The system was implemented on Android. Two applications have been implemented: *Hlocator* for Android, *Hlocator_web* (the other) on the server for monitoring.

The idea is to create an application for Android mobile devices to keep track of the current GPS position of one carrier. Hlocator at any instant of time and items, by calling the *chtag*, the coordinates of the last two tags of the *my_tags* (latitude and longitude) that identify the last position recorded by the mobile device.

The application makes use of *orderby*, C Hixosfs utility, that works even on IFHNFS [1] to organize the data already present, where each file represents a parcel post, by courier. In this way it will be possible through the utility *retag* retrieve tracking number of parcels from a given carrier. The application performs in the background sending the GPS coordinates to the server.

In particular, this Web-Application(2) shows the final position of the parcels on their way by reading the GPS coordinates in the inode or the journey by reading the contents of the file where you will find all the points sent by the GPS Android. In the first mode access to information of longitude and latitude will be faster (as demostration tests

[1] orderby working on local filesystem partitions as visible through the power of NFS

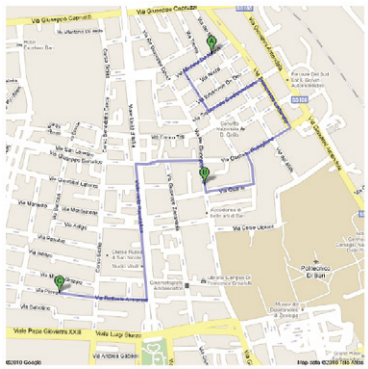

Fig. 1: *Maps*

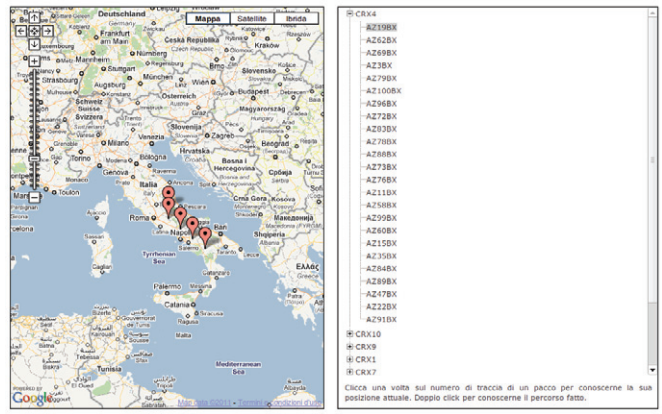

Fig. 2: *Hlocator*

of the following section). Here too, the organization's structure is made using filtered couriers utility *orderby* used for Android. The display of new packages require that the utility is invoked periodically so that the Web application should also provide for the new parcels submitted by couriers on the road.

Hlocator software is being tested and will be released shortly.

6. Comparison test

The testing phase focuses on analysis of the time required to read and modify discrete amounts of data. The comparison was conducted using the times recorded by using IFHNFS operations:

- Create 20,000 files
- Editing tags on 20,000 files created in previous test
- Reading tags of all 20,000 files
- Searching for a particular value in the tag between the 20,000 files on the partition located on the Server

compared with the time recorded by equivalent operations on two types of databases: Sqlite and Mysql. To perform these tests but you have created a simple database containing a single table with 5 fields (4 tags and 1 ID).

The ID field is an integer and is the primary key of the table. Simulates the name of a file in an operating system. The next 4 fields named TAG1, TAG2, TAG3, TAG4 are vectors of 28 characters. Simulate four additional tags, input from IFHNFS within the inode of each file.

The database tests were carried out operations similar to those that run on IFHNFS, so you can compare the execution time

CREATION	EDIT	READ	FIND
219,441	121,558	24,996	60,428
214,432	131,629	24,122	63,121
216,366	123,894	25,027	65,796
218,435	124,59	26,321	60,774
217,632	122,942	27,368	61,347
219,132	125,189	27,228	60,111
218,762	121,382	27,206	65,135
216,643	122,673	24,963	62,52
218,081	126,782	26,672	60,973
214,998	122,268	25,621	63,239

We repeted 10 times our tests to avoid any problems.

Average of observed values:

	CREATION	EDIT	READ	FIND
IFHNFS	217,3922	124,2907	25,9524	62,3444
Mysql	184,3335	188,9434	186,6458	188,9494
SQLite	513,110	69,451	495,746	69,349

It gives an idea of what data management via a file system operating on IFHNFS Hixosfs filesystem is more performant because firstly, the operations are carried out in Kernel Space.

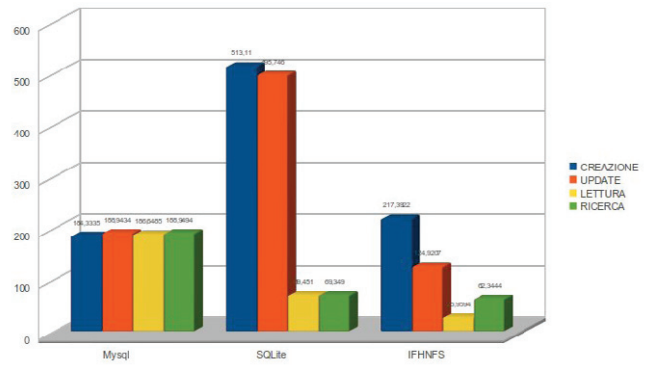

Fig. 3: Chart search data

This means that processes have a higher priority and are executed first. Secondly, the management of inode data structures speeds up the availability of information to the organizational structure of the filesystem that directory and file allocation is done through the use of facilities inode.

7. Conclusion

We presented ifhnfs as a new solution to share the content of a file within a network. Our purpose is to combines the advantages of hixosfs file system for the search of file content and the advantages of network file system to share data within a network. IFHNFS has been tested in postal environments implementing a real-time tracking system for packs which are sent potentially all over the world. We compared the idea with the standard approach and we noticed how the approach is more performant.

References

[1] Network File System. http://nfs.sourceforge.net/
[2] Hixosfs file systems. http://www.di.uniba.it/hixos/hixosfs/index.html. Home Page.
[3] Corriero, Cozza. *The hixosfs music approach vs common musical file management solutions*, SIGMAP 2009.
[4] Openmoko. http://www.openmoko.org, 2010. Home Page.
[5] *Wifi Mesh for HandHelds in Linux*, Corriero, Cozza, Pistillo, Zhupa. ICWN 2008
[6] Ext2. Remy Card, Theodore Ts'o, S. T. http:// e2fsprogs.sourceforge.net/.
[7] Sqlite. http://www.sqlite.org/. Home Page
[8] Kernel. L. Torvalds. www.kernel.org.
[9] Understanding the linux kernel . Bovet & Cesati. O' Reilly.
[10] The "virtual filesystem" in Linux . Alessandro Rubini. Kernel Korner. http://www.linux.it/ rubini/docs/vfs/vfs.html.
[11] An embedded solution for managing an ad-hoc wireless network,2010,ISBN: 978-0-7695-4278-2, Corriero, Ruci
[12] Bluetooth and filesystem to manage a ubiquitous mesh network,2010,ISBN: 978-1-61208-000-0, Corriero , Covino, Pani , Zhupa
[13] Android. http://developer.android.com/guide/basics/what-is-android.html, Home Page