A

SYNOPSIS REPORT

ON

KWEST
A Semantically Tagged Virtual File System

Submitted by

Sr.No.	Name of student	Roll No.
1	Gogte Aseem	4220
2	Gupta Sahil	4222
3	Pandit Harshvardhan	4247
4	Sharma Rohit	4260

Under the Guidance of

Prof. V. V. Phansalkar

JSPM's
Rajarshi Shahu College of Engineering
Department of Computer Engineering

[2012-2013]

INDEX

ABSTRACT

The limitation of data representation in today's file systems is that data representation is bound only in a single way of hierarchically organizing files. A semantic file system provides addressing and querying based on the content rather than storage location. Semantic tagging is a new way to organize files by using tags in place of directories. In traditional file systems, symbolic links becomes non-existent when file paths are changed. Assigning multiple tags to each file ensures that the file is linked to several virtual directories based on its content. By providing semantic access to information, users can organize files in a more intuitive way. In this way, the same file can be accessed through more than one virtual directory. The metadata and linkages for tagging are stored in a relational database which is invisible to the user. This allows efficient searching based on context rather than keywords. The classification of files into various ontologies can be done by the user manually or through automated rules. For certain files types, tags can be suggested by analyzing the contents of files. The system would be modular in design to allow customization while retaining a flexible and stable structure.

Keywords: virtual file system, semantics, indexing, classification, database, tagging, information access, metadata.

INTRODUCTION

Traditional file systems are mono-hierarchical and implement directory trees to categorize and store files. In such systems, directories are the only means to access particular files. The path of a file contains directories, which refer to its context and categorization.

As an example “c:\photos\college\trip\museum*.jpg” refers to all photos of a museum from a college trip. In this case, it is not possible to store that photo in another directory say “c:\photos\museum*.jpg” without copying the file. This severely limits the searching capabilities in a file system. The user is faced with the dilemma of which directory best represents the context of current file. While storing, the file is identified by its file name alone, which serves as its identifier. For searching a particular file, the user has to accurately remember the path and file name. A file cannot be searched by any other information relating to its context. Creating the directory structure is based on the user’s organizational skills. Searching or browsing through someone else’s data is tricky as the organization is different for every user.

Previous approaches to such problems provided symbolic links and aliases as an incomplete answer. Symbolic links become redundant when the target file paths are changed. Similarly, aliases may become redundant or may not function properly with certain programs. Working with such solutions requires advanced skills on the user’s part.

Keyword based searches which extract metadata from files were brought to fore by Apple's Spotlight and Google's Desktop Search. Both function only on limited file types and do not allow manual categorization.

This led to the development of semantic file systems, containing categorization of files based on context. It provides access to files by using categories formed from extracting metadata. It is similar to how music files can be searched by artist, genre, album etc. However, this presents a limitation on the amount and capabilities of what metadata can be extracted from a file. Virtual directories are used

to represent data from the file system. These directories do not have a permanent listing and the user has to explicitly query for data.

There have been several implementations based on semantic file systems. However, they have several limitations in usability. Most of the projects are based only on a few key points, such as limitations over file types. Our project thus is to create a semantic solution to the problems and shortcomings of traditional file systems while covering the limitations of other implemented projects.

PROBLEM STATEMENT

Due to the several drawbacks of traditional mono-hierarchical file system, organizing and retrieving information becomes difficult. Semantic approaches to solve this problem have addressed some issues but contain several drawbacks. Features are spread between different implementations and often focus on requiring the user to have advanced sills. The purpose of this project is to focus on performing efficient information retrieval based on context.

A semantic file system allows searching and storing of data based on context rather than filenames and keywords. The advantage of creating a file system over keyword based search tools is that file systems allow system level management of data. Keyword based tools such as Apple Spotlight and Google desktop work by maintaining a database of metadata extracted from files. They do not provide any facilities to manage storage. These tools can only extract metadata from a pre-configured list of known file types. Plus, they do not allow metadata to be added by users. These kinds of tools do not provide a complete semantic experience that can be developed by semantic file systems.

Thus, the goal of this project is to develop a semantic file system that extracts metadata from files and allows storage and searching based on its context. Such a system should overcome the drawbacks of traditional file systems while leveraging the limitations of other such similar implementations.

AIM, OBJECTIVE & SCOPE

1. AIM

To enable storage and retrieval of information based on context through the use of a semantic file system and to model the categorization of information based on user's perspectives.

2. OBJECTIVE

1. To create a virtual semantic file system based on tags
2. To allow efficient searching and querying for information
3. To allow automated and manual tagging of data
4. To create a modular system which is portable and extendible

3. SCOPE

This project is a virtual file system capable of storing semantics with which it facilitates the finding of relevant information. Information is stored in tags, which are extracted from a files metadata. This information may be generated implicitly by the system or supplied explicitly by the user. Thus, the validity of information is based on the user's level of organizing things.

The current system can extract metadata from a limited set of known file types. However, the modular architecture allows for plugins to be added which can add functionality for other file types. This allows for the project to be extended and modified according to the functionality required.

The level of awareness generated by the system is based on the frequency of access and input provided by the user. The amount of relevance is determined by the tags generated and their associated files. This affects how the system categorizes and searches files. Thus the actual outcome of the system which is the searching capabilities is totally dependent on these relationships.

The current implementation is based on the Linux kernel. Future implementations can be extended to other platforms and devices. Furthermore, as the system is a virtual one, it needs only slight modifications to be ported to other file systems and operating systems.

FEASIBILITY STUDY

Our project is to create a semantic solution to the problems and drawbacks of traditional file systems while covering the shortcomings of other similar projects.

Kwest will use multiple tagging to categorize files based on its context. Its primary functionality is to automatically organize files in a transparent and seamless manner so as to facilitate easy retrieval. The virtual directories created in Kwest will be superficial and won't change or modify the location in which the data will be initially stored. It will only provide links and assign tags under which data can accessed.

Kwest will also provide a feature which will either automate the classification of the files or suggest possible tags to the user if the user wants to classify files manually.

Kwest can efficiently manage and navigate gigabytes of data. It can be used on a large scale to manage and access data in organizations where the need for effective and fast retrieval of relevant data is needed.

The Project is being implemented using loadable kernel module known as FUSE (Filesystem in USErspace). It is a kernel module in UNIX based operating systems that lets non-privileged user to create their own file system without editing the kernel code. Being open sourced, this module is available for public use. In the current versions of some UNIX based OS this module is included in the kernel itself.

The query processing and programing will be done using SQLite. It is a relational database contained in a small C programming library. In contrast to other database management systems, SQLite is not a process that is accessed from the client application, but an integral part of it. This is also open source and is freely available.

Thus the cost for developing Kwest will be minimal and hence will be feasible enough without the need for large capital.

The semantic file system developed in Kwest can be used in large organizations as well by the average computer user. Being an open source and modular based project the details and the specifications can be modified according to the priorities, making it flexible enough to implement.

Kwest will be implemented at this stage keeping in mind the usage of the average computer user. This will help us in deciding our scope and the number of modules to implement. Our main focus thus would be on providing seamless and efficient access to relevant information.

TECHNICAL DETAILS

Kwest is implemented as a virtual file system. It manages the organization of data while the storage is maintained by the underlying file system. It uses virtual directories to allow representation in the traditional model of directories and files. Tags are used to store metadata attributes of files. Searching is based on these tags, and the results are listed as virtual directories. The user can browse through the directories to search for relevant files. Kwest supports automated as well as manual tagging to allow user's choices for categorization of information. It also offers suggestions to the user in the form of related tags.

1. HARDWARE REQUIREMENTS

1. Minimum 128 MB RAM
2. 4 GB Hard Disk or Higher
3. At least 800MHz processor

2. SOFTWARE REQUIREMENTS

1. SQLite Version 3.7.13
2. FUSE lib 2.8.6
3. GCC 4.7.1

TECHNOLOGIES USED

1. FUSE library 2.8.6

Filesystem in Userspace (FUSE) is a loadable kernel module for Unix-like computer operating systems that allows non-privileged users create their own file systems without editing kernel code. This is achieved by running file system code in user space while the FUSE module provides only a "bridge" to the actual kernel interfaces.

2. SQLite Version 3.7.13

SQLite is a relational database management system contained in a small C programming library. In contrast to other database management systems, SQLite is not a separate process that is accessed from the client application, but an integral part of it. It implements a self-contained, zero-configuration, transactional SQL database engine which can be embedded in applications.

3. GCC 4.7.1

The GNU Compiler Collection (GCC) is a compiler system which provides front ends for various languages including C. It provides optimizations, debugging and other features to help program development. GCC is often chosen for developing software that is required to be executed on a variety of platforms.

INNOVATIVENESS AND USEFULNESS

1. INNOVATIVENESS

There have been many implementations on semantic file systems in past. Tagsistant allows manual tagging of files along with a modular design. SemFS provides API's and allows use of logical operators to filter results. KFS classifies files based on automated rules and also allows manual tagging. Each of these implementation promises some new feature, but there is no system developed that integrates them all.

Kwest is going to combine features of all existing implementations. While combining all these functionalities, the distinguishing features of Kwest that set it apart from others will be:

- Kwest sorts files automatically while giving users the flexibility to organize files in their own way.
- Modular design provides scope to add functionalities later. This refines the systems portability and extendibility.
- Kwest provides an export feature that allows the user to save contents of search results to an external location. Files are exported as a traditional hierarchy supported by generic file systems.
- It allows both, automated as well as manual classification of files based on metadata tags and file attributes.
- The user is in complete control over the operations carried out by the system.
- Metadata can be generated for all file types.
- Files are also categorized based on use, creation, size, etc.
- Virtual Directory Listings can be created at runtime to address searches.
- System can ported to any file system being a virtual file system.

2. USEFULNESS

- The user is more likely to find related information in fewer searches.
- The user does not have to remember path of file, only the context.
- System does not require advanced skills to use.
- A file can be found under any one its related tags.
- Virtual Directories enable files to be displayed under more than one directory.
- Virtual Directory listings provide a traditional layout for the user while maintaining the semantic tagging associated with files.
- Support for plugins allows users to add functionality to the project which can further be tagged under appropriate context.
- As it is a virtual file system, Kwest can be used with existing file systems already present on the machine.
- Querying for data is helpful when complex searches are to be implemented.

PROJECT OVERVIEW

This project aims at providing a feasible solution towards efficient contextual storage and searching of information. It implements a semantic file system which structures data according to their context and intent. This allows the data to be addressed by their content and makes relevance in searching an efficient operation.

The system extracts metadata into tags and stores it in a relational database. These tags can be file attributes such as size, type, name etc. as well as extracted metadata such as author, content title, etc. Categorizing files by metadata allows linking a file in multiple ways while being able to search it using its context. This enables the users to find relevant information in as few searches as possible.

Assigning tags can be managed by automated rules and manual inputs. This makes the semantics mold according to the user's perspectives and helps make information relevant to the person managing it. The modular architecture of the system allows for plugins which can extend the functionality. For example a plugin to add more detection capabilities for certain file types will enhance the metadata extraction on those files. This makes the system highly customizable to power users. The automated rules help automate tasks and data categorization based on user inputs.

Virtual directories are used to display stored files in a semantic organization. Search results are displayed through dynamically created listings, which correspond to semantic segregation. The entire implementation is based on a virtual file system which manages only the data organization. The underlying file system takes care of storage. This allows it to be ported in future to any file system.

Finally, the project is implemented using open source technologies, which greatly reduce the cost and compromises associated with paid software. Thus the project aims to address the current shortcomings of relevant information access and storage by creating a virtual semantic file system which manages the data and provides search information based on semantics.

PAPERS REFERRED

1. Knowledge file System

The Knowledge File System (KFS) is a smart virtual file system that sits between the operating system and the file system. Its primary functionality is to automatically organize files in a transparent and seamless manner so as to facilitate easy retrieval. Think of the KFS as a personal assistant, who can file every one of you documents into multiple appropriate folders, so that when it comes time for you to retrieve a file, you can easily find it among any of the folders that are likely to contain it. Technically, KFS analyzes each file and hard links (which are simply pointers to a physical file on POSIX file systems) it to multiple destination directories (categories). The actual classification can be based on a combination of file content analysis, file usage analysis, and manually configured rules. Since the KFS organizes files using the familiar file/folder metaphor, it enjoys 3 key advantages against desktop search based solutions such as Google's desktop Search, namely 1) usability, 2) portability, and 3) compatibility. The KFS has been prototyped using the FUSE (Filesystem in USErspace) framework on Linux. Apache Lucene was used to provide traditional desktop search capability in the KFS. A machine learning text classifier was used as the KFS content classifier, complimenting the customizable rule-based KFS classification framework. Lastly, an embedded database is used to log all file access to support file-usage classification.

This paper describes approach to personal information management through Knowledge File System (KFS). It is designed to help users organize information using VFS (virtual file system) to reduce the problem of manual information classification and retrieval. KFS provides functions so as to automatically classify the information based on the content similarity with respect to predefined ontologies or also give the option for manual classification of the information. The operations carried out on the KFS can also be monitored with the event logger feature. Searching of files can be carried out by keyword with the help of a text indexer. Furthermore the comparisons between Google desktop file system, beagle

and KFS are given. Finally the details of the implementation of the KFS on the Linux platform using of FUSE are given.

2. Semantic file system

In light of this context, we will initially discuss a number of disadvantages of current file management systems. In the body of the paper our main contribution is presented. That is, a formal mathematical model of a new semantic file system, SIL (Semantics Instead of Location), that allows engineers to access data based on semantic information rather than storage location is proposed. A major difference between our approach and previous related work is that we do not aim at yet another point solution and, instead, propose an approach that may be employed by next generation engineering data processing systems on a larger scale. In addition, a corresponding programming interface along with a graphical user interface used as a file browser is presented and the benefits of utilizing the proposed semantic file system for product data management in the field of integrated design of mechatronic systems are discussed.

This paper describes a formal mathematical model that allows engineers to access data based on semantics rather than actual storage location. The main goal of this file system is to search files based on content of data. The browsing of the system is done based on file's metadata and attributes. Logical operator such as AND, OR, NOT are used to filter the results. The classification allows multiple tags to be created for files. Furthermore API's are written to create views and for the automation of notification updates.

3. File-Type Identification

File-type Identification (FTI) is an important problem in digital forensics, intrusion detection, and other related fields. Using state-of-the-art classification techniques to solve FTI problems has begun to receive research attention; however, general conclusions have not been reached due to the lack of thorough evaluations for method comparison. This paper presents a systematic investigation of the problem, algorithmic solutions and an evaluation methodology. Our focus is on performance

comparison of statistical classifiers (e.g. SVM and kNN) and knowledge-based approaches, especially COTS (Commercial Off-The-Shelf) solutions which currently dominate FTI applications. We analyze the robustness of different methods in handling damaged files and file segments. We propose two alternative criteria in measuring performance: 1) treating filename extensions as the true labels, and 2) treating the predictions by knowledge based approaches on intact files as true labels; these rely on signature bytes as the true labels (and removing these signature bytes before testing each method). In our experiments with simulated damages in files, SVM and kNN substantially outperform all the COTS solutions we tested, improving classification accuracy very substantially – some COTS methods cannot identify damaged files at all.

This paper explains File-type Identification (FTI). FTI is the task of assigning a predefined label (the file type) to each instance (file) based on observed data in the file. The conventional application of FTI is in operating systems where computers need to choose different programs to process the information based on the type of each file.

4. Semantic file retrieval using virtual directories

Hard Disk capacity is no longer a problem. However, increasing disk capacity has brought with it a new problem, the problem of locating files. Retrieving a document from a myriad of files and directories is no easy task. Industry solutions are being created to address this short coming. We propose to create an extendable UNIX based File System which will integrate searching as a basic function of the file system. The File System will provide Virtual Directories which list the results of a query. The contents of the Virtual Directory are formed at runtime. Although, the Virtual Directory is used mainly to facilitate the searching of file, It can also be used by plugins to interpret other queries.

This paper describes the design of SemFS, which provides semantics based on the file's meta-data and attributes. It allows the usage of logical operators to filter query results. It is implemented as a user space file system upon journaling

storage. The architecture is server-client with support for API's to extend functionality. Features such as file tagging and versioning are also implemented.

5. Semantic-Aware Metadata Organization Paradigm

Existing data storage systems based on the hierarchical directory-tree organization do not meet the scalability and functionality requirements for exponentially growing data sets and increasingly complex metadata queries in large-scale, Exabyte-level file systems with billions of files. This paper proposes a novel decentralized semantic-aware metadata organization, called SmartStore, which exploits semantics of files' metadata to judiciously aggregate correlated files into semantic-aware groups by using information retrieval tools. The key idea of SmartStore is to limit the search scope of a complex metadata query to a single or a minimal number of semantically correlated groups and avoid or alleviate brute-force search in the entire system. The centralized design of SmartStore can improve system scalability and reduce query latency for complex queries (including range and top-k queries). Moreover, it is also conducive to constructing semantic-aware caching, and conventional filename-based point query. We have implemented a prototype of SmartStore and extensive experiments based on real-world traces show that SmartStore significantly improves system scalability and reduce query latency over database approaches. To the best of our knowledge, this is the first study on the implementation of complex queries in large-scale file systems.

The paper demonstrates the SmartStore system, which groups and stores file by their metadata semantics. The design of the system is modeled to be efficient and scalable with capability to handle complex queries. Versioning is used to maintain consistency amongst the internal data structures. The system tries to minimize the search area of any query by using pattern matching, caching and range dynamics.

REFERENCES

1. Kuiyu Chang, I Wayan Tresna Perdana, Bramandia Ramadhana, Kailash Sethuraman, Manish Rai Jain, Imam Kartasasmita, Truc Viet Le, Neha Chachra, Sahil Tikale, "Knowledge File System-A principled approach to personal information management", 2010 IEEE International Conference on Data Mining Workshops , Page(s): 1037- 1044

2. Oliver Eck, Dirk Schaefer, "- A semantic file system for integrated product data management", 2011 Advanced Engineering Informatics 25 177–184

3. Siddharth Gopal,Yiming Yang,Konstantin Salomatin,Jaime Carbonell, "Statistical Learning for File-Type Identification", 2011 10th International Conference on Machine Learning and Applications , Page(s): 68- 73

4. Prashanth Mohan, Raghuraman, Venkateswaran S and Dr. Arul Siromoney, "Semantic File Retrieval in File Systems using Virtual Directories", Proc. Linux Expo Conference, Raleigh, NC, pp. 141-151, May 2007.

5. Yu Hua, Hong Jiang, Yifeng Zhu, Dan Feng and Lei Tian, "Semantic-Aware Metadata Organization Paradigm in Next-Generation File Systems", IEEE Transactions On Parallel And Distributed Systems, Vol. 23, No. 2, February 2012, Page(s): 337- 344

6. C. McMahon, A. Lowe, S. Culley, M. Corderoy, R. Crossland, T. Shah, D. Stewart, "Waypoint: An integrated search and retrieval system for engineering documents", Journal of Computing and Information Science in Engineering (4) (2004) 329–338.

7. S. Faubel and C. Kuschel, "Towards Semantic File System Interfaces", 7th International Semantic Web Conference ISWC2008.

8. C. A. N. Soules, G. R. Ganger, “Why can’t I find my files? New methods for automating attribute assignment”, Technical Report CMU-CS-03-116, School of Computer Science, Carnegie Mellon University, 2003.

9. Stephan Bloehdorn, Olaf Görlitz, Simon Schenk, and Max Völkel (2006): TagFS - Tag Semantics for Hierarchical File Systems. In Proceedings of the 6th International Conference on Knowledge Management (I-KNOW 06), Graz, Austria, September 6-8, 2006. September 2006.

10. Michal Tvarožek and Mária Bieliková (2008): Personalized View-Based Search and Visualization as a Means for Deep/Semantic Web Data Access. In Proceedings of the 17th International World Wide Web Conference (WWW 2008), Poster, 2008.

11. File system in USERspace (FUSE) homepage and documentation : http://fuse.sourceforge.net/

12. SQLite database homepage and documentation: http://www.sqlite.org/