Kwest

A Semantically Tagged Virtual File System

Aseem Gogte, Sahil Gupta, Harshvardhan Pandit, Rohit Sharma
Department of Computer Engineering, RSCOE, University of Pune, Pune, India
Presented at: International Conference on Advanced Computer Engineering and Applications (ICACEA) 2012

***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 become 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*— semantics, indexing, classification, database, tagging, virtual file system, information access, metadata

I. 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 users organizational skills. Searching or browsing through someone else's data is tricky as the organization is different for every user.

Previous approaches [1] 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 [3] and Google's Desktop Search [2]. 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 [11] 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 systems are based only on a few key points, such as limitations over file types.

Our aim thus is to create a semantic solution to the problems and shortcomings of traditional file systems while covering the limitations of other implemented systems.

II. RELATED WORK

Over the years, organizing and retrieving information accurately and efficiently has attracted lot of attention. While few have been successful, a number of innovative implementations [1] have emerged. The idea of using a file's semantics as the means to categorize it has been around for quite some time. This section discusses the various implementations made in the field of semantic file system. An efficient implementation of keyword based searching was brought to the desktop by Google's Desktop Search [2] and Apple's Spotlight [3]. Both allow efficient and quick file retrieval based on keywords. They support many file types and have a simple interface which attracts a large number of users. However, both of them are limited to returning search results without any way to organizing contents. In addition, they do not provide any provision to the user for classification of data. This limitation prevented the user from having a personalized way to retrieve data stored by them.

Semantic systems depend on data stored inside the files rather merely relying on an file's attributes. Most implementations use common methodologies like content recognition [4], tagging [5], extracting metadata, etc. to categorize files by using various algorithms.

"Semantic File System" [6], as developed by O'Toole and Gittord in 1992, provides access to file contents and metadata

by extracting the attributes using special modules called ”transducers”. It was one of the very first attempts to classify files by semantics using metadata. Its biggest drawback was the need for file type specific transducers which were necessary to extract meta information and content from the file. Also, the user does not have any say in what kind of category the file is classified under. This drawback makes it an unattractive option to the general user. It was decided during designing Kwest, that it is necessary to involve the end-user in the tagging process. This allows each user to have their own personal way of classification and organization of files.

NHFS (Non Hierarchical File System) [7] was a system developed by Robert Freund in July 2007. It allows the user to place any file into any number of directories. Likewise, any directory can be placed into as many directories as required. NHFS therefore allows one to create a non-hierarchical structure with poly-hierarchically connected files. This allows for a powerful metaphor of finding a file in any of the category (directory) it could be stored under. Therefore, we decided to retain this feature by using tags in place of actual directories. Tags are associated with files and other tags as well. Thus, a tag may be placed under multiple tags allowing a relationship to be defined between them. This analogy is much more powerful than restricting files to actual directories. Using tags prevents duplication and redundancy, making it an efficient implementation.

A more recent implementation is Tagsistant [8], which is a semantic file system that also attempts to organize files using tags. It interacts with the Linux kernel using the FUSE module. Under Tagsistant, directories are considered to be equivalent to tags. As a consequence, creating a directory is creating a tag and putting a file inside a directory means tagging that file. After you have tagged your files, you can search all of them by using queries. Queries are just paths where each element is either a directory or logical operators. The entire system has a modular design and uses SQLite. However, it suffers from some speed issues and the lack of SQL indexes. Major flaws of this design were high consumption of inodes on real file systems and high computational time which was required to fulfill each request. Most of the features of Tagsistant were decided to be included in Kwest. These were modular design, SQLite repository, tagged structure, etc. which enhance the semantics of a file system. However, care must be taken to prevent the occurrence of similar drawbacks.

Another implementation called Tagster [9], is a peer-to-peer tagging application for organizing desktop data. It is platform independent and is implemented in JAVA. Multiple files and also directories can be tagged through its interface. The selected directories are recursively examined and all files contained within them are tagged. The GUI for a Linux system consists of three main areas. Namely - ”Tag view”: which displays a list of tags, ”Resource view”: which lists resources that have the currently selected tags assigned and ”User view”: that displays a list of users that have tagged the currently selected resource with some selected tag. It also includes GUI support for Windows with some unresolved issues. However, it lacks auto classification of data due to which several common tags may be generated for each user increasing the database size.

III. Proposed System

Kwest is a virtual file system that is designed to help users organize information using the familiar hierarchical file/directory structure. It 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 system is implemented using open source technologies, which greatly reduce the cost and compromises associated with paid software. Thus the system 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. The major design features are described in this section.

A. Tags

1) Manual Tagging:

Manual tagging is the basis of semantics in Kwest. The user can assign any tag to the files in Kwest. These tags are then stored internally in a database. The user can create new tags or use tags already defined by the system. Total freedom is given to the user to organize data.

2) Automatic Tagging

Kwest also features automatic tagging of files. The user can define certain rules under which files will be assigned tags. The system will implement those rules for all files satisfying the defined constraints. This would prevent repetitive tagging operations for the user.

3) Importing tags

Certain popular file formats such as mp3, jpeg etc. have metadata embedded in them. Kwest supports such popular format and uses this metadata to automatically assign tags to the files. This feature enables the user to collectively classify and store the data under these tags.

B. *Database*

1) *Consistency*

Kwest uses an internal database to store and manage data. It is vital that the database always remains consistent. Kwest uses logging mechanisms to ensure that operations on the database always reach an endpoint.

2) *Access*

The database is included in the same directory as the Kwest executable. The files are not locked down or are access restricted. Other applications, modules or tools can access the database. However, this feature is made available with the understanding that the integrity of the database will be maintained always.

C. *Relation with existing data*

1) *Importing semantics*

Users already have certain organizational structures in the way they store data in file systems. Kwest imports these semantics by converting the storage hierarchy to tag-based hierarchy. This allows the entire file system to be imported into Kwest along with the users' previous organization structure.

2) *Reflecting changes to filesystem*

When users carry out certain changes in Kwest such as copying files, deleting files etc., these changes are virtual and do not affect actual file systems. However, Kwest can enforce these operations on the real files in certain cases.

D. *Exporting semantics*

1) *Export filesystem*

As the entire file system exists as a virtual entity, Kwest provides the export feature. Where the file system can be exported to another system where the data can be imported by another instance of Kwest.

2) *Export tagged files*

It is also possible for the user to export data under certain tags to an external location. The semantic organization showed by tags is converted to actual directories and files are then copied to these directories. This way the user can export Kwest semantics and data to outside locations.

E. *Modularity*

1) *Modules as plugins*

Kwest is an extendible system. It can use external modules to increase functionality or to modify existing operations. Support for using modules is built into Kwest right from the design stage.

2) *Support for developers*

Kwest provides support to developers by providing access to all internal features and database. The API layer allows developers to easily supplement internal operations with their modules.

IV. System Design

Kwest is implemented using loadable kernel module known as FUSE. User may interact with kwest like any other file system via Command line or file managers like Nautilus. Data is passed on to FUSE through the virtual file system. FUSE implements the operations of file system. FUSE uses Glibc and Libfuse for performing its operations. Glibc is the GNU Project's implementation of the C standard library. It provides functions for tasks like I/O processing, mathematical computation, memory allocation, etc. Libfuse contains functions internally used by fuse to create and manage virtual file system. SQLite will be used for storing all data relevant with the file system. To extract metadata, Kwest makes use of external libraries such as Taglib, EXIF.

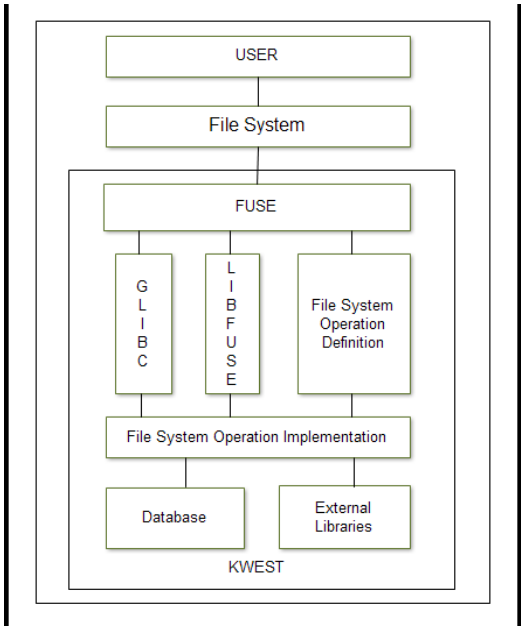

Figure 1: Design of Kewst

V. Mathematical Model

The relationship between files and tags can be represented by using Set theory. Set theory is the branch of mathematics that studies sets, which are collections of objects. The following mathematical model represents the working of this file system.

The following dynamic and variable sets are defined as,

F: Set of Files

T: Set of Tags

S: Set of Tags in query ($S \subseteq T$)

A. *Relation between Files (F) and Tags (T)*

$$R = \{ (f, t) \mid f \text{ has tag } t; f \in F, t \in T \}$$

Here R defines the relation between a file f and its tag t where $R \subseteq F \times T$. This relationship is many-to-many. That is a file can have many tags, and a tag can describe many files.

B. Operations

$$g(f) = \{ t : f \, R \, t \}$$

g is an operation which takes input as files f and returns the set of tags ($t \in S$) related by R to that file.

$$h(t) = \{ f : f \, R \, t \}$$

h is an operation which takes input as tags t and returns the set of files ($f \in F_S$) related by R to that tag.

C. Storing Tags and Files

The relation R is stored as a set of ordered pairs (f, t), where $R \subseteq F \times T$. The operations g and h operate on these ordered pairs and return mapped or matched elements. A relation which has to be added must be represented in the form of ordered pair (f, t). Storage of all relations is given by $F \times T$ where ordered pairs exist according to

$$R = \{f \in F, t \in T \mid f \, R \, t\}.$$

For example, we have the sets and their relations as:

$$F = \{f_1, f_2, f_3\}, T = \{t_1, t_2, t_3\},$$

$$R = \{f_1 \, R \, t_1, f_2 \, R \, t_2, f_3 \, R \, t_1, f_1 \, R \, t_3\}$$

Then we store this relation by its ordered pairs given by:

$$R = \{(f_1, t_1), (f_2, t_2), (f_3, t_1), (f_1, t_3)\}$$

D. Queries

A query operation on a single tag is expressed as:

$$q(t) = F_S \text{ where } \{ f \in F_S \mid h(t) = f \}$$

The general form of a query is a string which contains tags and operators. For example, we have two tags (t_1, t_2) and operator σ. The query Q can be defined in terms of q as:

$$Q(t_1, \sigma, t_2) = q(t_1) \, \sigma \, q(t_2)$$

The operation σ can be any one of Union ∪, Intersection ∩, Symmetric difference Θ etc. If no operation is explicitly mentioned, by default Intersection ∩ is performed.

VI. Conclusion And Future Work

In this paper we have proposed a system for organizing files using meta information by exploiting semantic information to provide efficient and scalable architecture. The system handles complex queries while enhancing functionality. Its novelty lies in the way it associates tags and derives rules that enables traversal based on semantics rather than path.

Currently, Kwest is in its initial stage of development. Its features are limited but its modular architecture allows plugins to be added which can add additional functionality, and recognition for more file types. This allows the system to be extended and modified according to the functionality required. The current implementation is based on the Linux kernel. Future implementations can be extended to other platforms and devices. As the system is a virtual entity, it does not need extensive modifications to be ported to other file systems and operating systems.

References

[1] Mangold. C, A survey and classification of semantic search approaches, *Int. J. Metadata, Semantics and Ontology*, Vol. 2, No. 1, 2007, Page(s): 23-34.

[2] Google Desktop Search, http://googledesktop.blogspot.in

[3] Apple Spotlight, http://developer.apple.com/macosx/spotlight.html

[4] Gopal. S, Yang. Y, Salomatin. K, Carbonell. J, Statistical Learning for File-Type Identification, *2011 10th International Conference on Machine Learning and Applications*, Page(s): 68-73.

[5] Bloehdorn. S, Grlitz. O, Schenk. S, Vlkel. M, 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.

[6] Gifford. D, Jouvelot. P, Sheldon. M, OToole. J, Sematic File Systems, 13th ACM Symposium on Operating Systems Principles, *ACM Operating Systems Review*, Oct. 1991, Page(s): 16-25.

[7] Freund. R, File Systems and Usability the Missing Link, *Cognitive Science, University of Osnabruck July 2007*.

[8] Tagsistant, http://www.tagsistant.net

[9] Tagster, http://www.uni-koblenz.de

[10] Chang. K, Perdana. I, Jain. M, Kartasasmita. I, Ramadhana. B, Sethuraman. K, Le. T, Chachra. N, Tikale. S, Knowledge File System - A principled approach to personal information management, *2010 IEEE International Conference on Data Mining Workshops*, Page(s): 1037-1044.

[11] Mohan. P, Venkateswaran. S, Raghuraman, Dr. Siromoney. A, Semantic File Retrieval in File Systems using Virtual Directories. *Proc. Linux Expo Conference, Raleigh, NC*, May 2007, Page(s): 141-151.

[12] Hua. Y, Jiang. H, Zhu. Y, Feng. D, Tian. L, 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.

[13] Schroder. A, Fritzsche. R, Schmidt. S, Mitschick. A, Meiner. K, A Semantic Extension of a Hierarchical Storage Management System for Small and Medium-sized Enterprises, *Proceedings of the 1st International Workshop on Semantic Digital Archives (SDA 2011)*.

[14] Eck. 0, Schaefer. D, A semantic file system for integrated product data management, *2011 Advanced Engineering Informatics*, Page(s): 177-184.

[15] File system in USERspace (FUSE) homepage and documentation, http://fuse.sourceforge.net

[16] SQLite database http://www.sqlite.org