TagFS — Tag Semantics for Hierarchical File Systems

Stephan Bloehdorn*, Olaf Görlitz, Simon Schenk**, Max Völkel*****

*Institute AIFB, University of Karlsruhe, Germany
{bloehdorn}@aifb.uni-karlsruhe.de

**ISWeb, University of Koblenz-Landau, Germany
{goerlitz,sschenk}@uni-koblenz.de

***Forschungszentrum Informatik, Karlsruhe, Germany
{voelkel}@fzi.de

Abstract: Today, most computer users work with traditional hierarchical file systems for organizing large amounts of personal files. Recently, tagging has grown popular as an alternative means of organizing information resources. We argue that tagging is a powerful paradigm for efficient information access which overcomes many deficiencies of hierarchical file systems, especially in the context of the organization of large quantities of personal files. In this paper we analyze the different semantics of strictly hierarchical and tagging-based organisation and present TagFS, a filesystem with tagging support which aims at a seamless integration of the tagging paradigm with local applications. While retaining the notions of directories and files and providing all standard filesystem operations for easy integration with existing applications, the semantics of these operations are changed in TagFS to modifications of the respective tag annotations.

1 Introduction and Background

The amount of digital information stored on single computers is increasing steadily [Lyman and Varian, 2003]. Most users are familiar with the hierarchical file system of their operating system which they use for organizing the large amount of available files.

Classical Filesystems

The major building block of a hierarchical file systems is the *directory* as an organizational unit which acts as a container for files or further (sub-)directories. A structure of nested directories is commonly used for a sophisticated organization of files at multiple levels according to the semantics of their content from most general at the top level to most specific at the leaf levels of the directory hierarchy. Behind the scenes, an index structure (the file allocation table or inodes) provides a mapping from a set of *access paths* to the corresponding blocks of information. As such, the access paths act as *keys* for accessing the file content. The perceived organization of files into directories is a result of the structure of the access paths. Each access path can be decomposed into a sequence of directories (which we will call the *location*) and a *file name* at the end.

As an example, a typical organization pattern for storing digital versions of scientific papers of interest would be to maintain different directories for different research areas possibly including refinements for different scientific subfields. Correspondigly, the access path at `/paper/2003/semweb/gruber03.pdf` can be decomposed into *location* = `paper`, `2003`, `semweb` and *file* = `gruber03.pdf`.

Unfortunately, traditional hierarchical file systems don't serve this purpose well. Barreau and Nardi [Barreau and Nardi, 1995] observed in a study, that *"Every user [...] indicated that their attempts to establish elaborate filing schemas for archived information failed because they proved to require more time and effort than the information was worth."* We argue that the reasons for these problems are rooted in the hierarchical filesystem as such, namely the *single location property* of traditional filesystems and its consequences, namely the requirement to browse to maximum specificity, the missing orthogonality of nested directories, the dependance on a static order of directory names as well as a number of user interface problems as the lack of query refinement capabilities and navigational aid.

The Tagging Paradigm

Recently, *tagging*, which typically refers to the simple annotation of information resources with an arbitrary number of freely chosen descriptors, has become an alternative approach for semantically organizing information resources, and is typically considered to be a core component of the emergent Web 2.0. As such, tagging has grown in popularity especially in the context of collaborative tagging systems as for example del.icio.us[1] for bookmarks or flickr[2] for digital images.

The actual storage of an information item in a tagging system is typically opaque to the user. The file content is accessed by means of a *resource key*[3]. On the tagging layer, *tag annotations* are freely attached to resources. Formally, tag annotations can be seen as unary predicates applied to the resources. A set of arbitrary tags is often treated as a conjunctive query, denoting the intersection of the tag-induced sets. Consequently, tags or tag combinations, and the sets they imply, become the major conceptual units of tag-based information management. Information organization based on tags overcomes the problems identified for information organization in hierarchical file systems, which are described in the next section. However, current desktop interfaces do not support this task and, hence, cannot smoothly be integrated with shared, tagged resources.

[1] `http://del.icio.us`
[2] `http://www.flickr.com`
[3] The resource key may in fact be (and often is) a simple access path on a lower "storage" layer.

Contribution

The main contribution of this paper is a clean mapping of non-hierarchical tagging and tag-query operations to existing hierarchical file system operations. We present TagFS, a *file system with tag semantics*, as a means for managing, browsing and retrieving large amounts of files efficiently. On the surface, the resulting system behaves very much like a traditional file system but instead of interpreting directory structures as static storage hierarchies they represent dynamic views on a set of tagged information objects. Within those views, filesystem operations are mapped to manipulations of tags on information objects. At the same time, the tagging information is stored in RDF in order to enable the easy integration with semantic web and other semantic desktop applications. By doing so, many external applications and data repositories can be accessed as parts of a virtual file system and semantic web inferencing can be exploited based on schema-level descriptions about the tags. Storing metadata separately from the data allows for easy development of collaborative applications through metadata sharing. In this extended abstract, we restrain to a description of the conception of the mapping, which we will complement with a detailed description of our prototype system TagFS in the final version upon acceptance of this paper.

2 Mapping Tagging Semantics to Hierarchical File Systems

The following section describes the mapping between the tag-based and the hierarchical file system approach on the basis of three use cases: retrieval, reorganisation, and adding of files to TagFS.

Retrieval: Mapping Locations To Queries

In our mapping, directories are interpreted as tags and locations (sequences of directories) are interpreted as conjunctions of tags. We define the two functions *view(location)* and *subfolders(location)* for browsing and retrieval. In a file system, the view of a directory is the union of both functions.

view returns all files, or – strictly speaking – resources, that have been tagged with all tags contained in the referenced location. Thus we treat folders as tags and files in a location represent the query results. E. g., a view of the folder `/paper/tagging/iknow/2006` might show all papers that deal with tagging related topics and have been published at iknow in 2006[4].

subfolders returns all those tags as subdirectories, which can be used as a refinement of the current conjunctive query without returning an empty query

[4] Note that the folder view for a different permutation of these tags would return the very same set of resources as it corresponds to the equivalent conjunctive query.

result. In the current example, the directory `/paper/tagging/iknow/2006` might contain a subdirectory `semweb` as an indication of a possible refinement.

Organization: Mapping File System Operations to Tag Changes

We use the following mapping from file system to tagging semantics, roughly mapping files to resources and tags to folders.

delete removes the most specific tag from the specified file. This operation is consistent with the traditional file system in that the file will no longer occur in the specified folder. In order to really delete a file, it has to be tagged with `delete`. This results in the immediate removal of the specified file from the storage backend and removal of all tagging references to this file. Alternatively, a file is deleted from the system implicitely if it does not carry any tag annotations any more.

copy assigns additional tags from the target folder to the file in question. Note that this will in general result in additional results for the subfolders(...) operation on the source folder.

move consistent with the *delete(...)* and *copy(...)* operations, this operation removes all the tags denoted by the source location and assigns all the tags denoted by the target location to the file in question. This behavior is again consistent with the behavior of traditional file systems in that the file disappears from the old location and appears in the new location.

createDirectory (tag) Creating a new directory is only necessary if no file carries the corresponding combination of tags so far. However, the newly created directory would not be shown at the location it was created, because only tags with non-empty query results are listed. Therefore the corresponding tag is handled exceptionally by means of the following heuristic: TagFS uses a special placeholder file `just_created`, tagged with all the tags implied by the current location plus the newly introduced tag. The placeholder file will remain tagged until an actual file receives the relevant tags, the user session has ended or a timeout has occurred. If the corresponding tag does not yet exist, it is created.

move, copy or delete a directory These commands are executed for each file at source location.

Adding Files: Mapping Files to Resources

Resources, by definition, unambiguously identify file contents. The file name as the last part of the traditional access path can, taken by itself, be ambiguous in

the context of a tag based file system, because the root folder lists all files in one directory and requires us to use variants as unique filenames. For presentation, we change the filename at insertion time by adding a disambiguation number into the file name just before the file type extension. In our previous example this would result in `gruber.pdf` and `gruber-1.pdf`[5] The corresponding operation can thus be described as:

put insertion of the external file content into the storage backend, uniquely mapped to a randomly generated resource key; and assignment of all tags denoted by the location to the resource key.

3 Related Work

A study on personal file management [Barreau and Nardi, 1995] states a strong preference of users to guess file locations and browse for a file in a list rather than using keyword search tools. Often they find their files in the first or a few tries. To achieve this, users consciously organize their files for easy retrieval, using directories to relate files with a work context.

There are some works which try augment file systems with additional semantics. Back in 1991, Gifford et. al. [Gifford et al., 1991] list about ten implementations of *virtual file systems* in UNIX. A virtual file system is described as a set of *virtual folders* which appear within existing tree structured file systems. A *virtual directory* abstracts storage locations away and lists files based on a query which determines the content dynamically. Gifford's system extracts attributes, modeled as key-value pairs, from file via a set of *transducers*, one for each file type. He maps virtual directory names to queries, using one path entry for the key and the following for the value. E. g. `/location/London/author/Max` would list all images showing London, shot by Max. Gifford distinguishes between normal and hidden virtual directories, to remain compatible with existing network file system protocols and file archiving tools such as *zip* and *tar*.

An extension [Ayala, 2005] to the popular browser Firefox allows users to browse their del.icio.us tags as a virtual hierarchy. However, the support is read-only and only two levels of tags are rendered as nested folders.

The Logic Information System FS [Padioleau and Ridoux, 2003] follows a logic approach and its navigation is very akin to the one in TagFS though not based on RDF and an explicit tagging paradigm.

[5] To avoid duplication of files within the system an additional check of a content hash is performed. This enables files which are copied from the semantic file system to another store (e. g. email attachment) to be identified as the same file, when re-inserted it into the semantic file system.

4 Conclusion and Outlook

In this extended abstract we have presented TagFS, a virtual file system with tagging semantics. TagFS allows users to manage their files collaboratively through their existing desktop applications for file management by overloading common file system operations with tagging semantics. We restrained to a description of the conception of the mapping, which we will complement with a detailed description of our prototype system TagFS in the final version upon acceptance of this paper.

References

[Ayala, 2005] Ayala, D. (2005). Foxylicious - firefox and del.icio.us bookmark integration.

[Barreau and Nardi, 1995] Barreau, D. and Nardi, B. A. (1995). Finding and reminding: file organization from the desktop. *SIGCHI Bull.*, 27(3):39–43.

[Gifford et al., 1991] Gifford, D. K., Jouvelot, P., Sheldon, M. A., and James W. O'Toole, J. (1991). Semantic file systems. In *SOSP '91: Proceedings of the thirteenth ACM symposium on Operating systems principles*, pages 16–25, New York, NY, USA. ACM Press.

[Lyman and Varian, 2003] Lyman, P. and Varian, H. R. (2003). How much information. `http://www.sims.berkeley.edu/how-much-info-2003`, retrieved on 02.11.2005.

[Padioleau and Ridoux, 2003] Padioleau, Y. and Ridoux, O. (2003). A logic file system. USENIX 2003 Annual Technical Conference, General Track, pages 99112.