A novel idea for a new Filesystem

Storing facets as key-value pairs for individual files

Pieter Omvlee
mail@pieteromvlee.net

ABSTRACT

This paper outlines a high-level idea of new kind of a filesystem. It takes inspiration from commonly-used third-party applications that manage music, photos or other files. The fact that so many people use these applications is already an indication that the regular filesystem is not the best way to manage these kind of files. The new filesystem will rely on metadata to describe the different facets of individual files. As such, this new filesystem will be more flexible because in hierarchical filesystems, only one trail of facets can be represented in the hierarchy.

A popular way to represent metadata is by using tags, or keywords. We acknowledge that tags need semantic information to allow relevant queries on the data. The semantic meaning of the tags should be kept on a per-file basis because words can have different meaning in different contexts. Because of this, metadata will be kept as a key-value dictionary for every individual file. Another way of visualizing this system is as one large database table: the rows are the files, and the tags are placed in the appropriate columns where the title of the column is the semantic meaning of the tag in a cell. Viewing a subset of files can then be done using regular SQL queries.

Keywords

Hierarchical, filesytem, Tags, Metadata, Facets, Dictionary

1. INTRODUCTION

The filesystem of all modern desktop operating systems is a hierarchical one. The primary organizational tool is a folder which can contain files and other folders. In this way a hierarchy, or tree-like structure is created. It is a simple way of organizing files, based on the metaphor of real-life office cabinets containing file folders and files.

This basic idea has lot of problems associated with it [1], one example being that a file can only reside in one location without being duplicated. Researchers have known these problems since the beginning [2], but they still have not been solved. The difficulty here lies in the fact that these problems are on a very deep conceptual level and what is really needed is a fundamentally different approach to organizing files. A lot of research has been done and many alternatives have been suggested [3], [4], [5] but none of these ideas have yet taken it into mainstream computing. As such, the question is still open as to what would really be a good replacement for the current hierarchical filesystems.

11th Twente Student Conference on IT, Enschede, June 29th, 2009

Finding a suitable alternative to current hierarchical filesystems will be the main objective of this paper. This filesystem should have none of the problems that plague current filesystems and it should be easy enough to be used by the average computer user.

We will see that the principal problem of hierarchical filesystems lies in their lack of flexibility. The problem I described earlier is a good example of what I mean by that. In online communities like Flickr or Delicious, users organize and annotate files and urls with so-called 'tags' in a more flexible way. Tags are keywords a user can apply to a body of information describing what it is about. These entities can be given any number of tags or none, if that makes sense. As such, the user has a lot of flexibility in organizing their files. It will not come as a surprise that many researchers have argued to introduce tags in desktop systems and give them a a central role there [1], [2], [8], [9]. Although I share their opinion, this paper will introduce a fundamentally different way of looking at tags. I share with them the opinion that we should use tags as metadata to describe our files and distill a way of ordering out of them. The way I differ from previous research is the question of how to organize these tags, and in this paper I will explain why that is and what is in my opinion a better way of managing metadata.

Closely related to this is the observation that many users use separate applications to manage certain kind of files. Multimedia files like music [6] or photos [7] are good examples of files that are often managed by dedicated applications. Because these applications fill a void that was left open by the filesystem, they must be offering someting of value to these users, otherwise they wouldn't be used. Are there lessons we can learn from these applications and apply them to our new filesystem-idea? How do they attempt solve the problems that hierarchical filesystems are facing?

The main contribution of this paper is a high-level proposal for a new filesystem, a new way to organize files. This filesystem should have none of the deficiencies of existing hierarchical filesystems and should be easy-to-use and flexible for the average computer user. To find a solution, I will answer the following questions:

- What are the problems with current hierarchical filesystems? Why are these problems and what do they prevent the user from doing?
- What are the characteristics of these dedicated applications, such as those that manage music or photos? Can we apply some of the characteristics to our filesystem?
- We acknowledge that metadata should play a central role in this new filesystem. Can we use tags to describe metadata, and how should this metadata be structured?

Throughout this paper, we will see that the concept of 'facets' keeps reappearing. This concept is important to remember. The ability to handle the different facets of files will be the major requirement for this new filesystem. While I will explain later

why I think this is so, it is the red wire through this paper and as such important to remember.

2. HIERARCHICAL FILESYSTEMS

The metaphor of hierarchical filesystems is based on the real-life file folders used in office cabinets. Over time, this idea evolved to allowing nested folders from the original systems which supported folders only one level deep. A commonly heard argument in defense of the current filesystems is that the metaphor makes it very easy to understand the system. When one critically looks at this argument, it will be obvious that this argument doesn't hold. In real life, file folders cannot contain file folders themselves.

It has already been mentioned a few times that there are big problems with current hierarchical filesystems. The most important and prolific problems are listed below but there certainly is overlap between individual points. In fact, one concept lies at the root of several of these problems.

2.1. A unique path to a file

A single file one disk can only reside in one folder. After having saved a document, one can of course move it to create a completely different folder structure if that makes sense. It is however impossible to keep two alternative folder structures, with the same files in both hierarchies, at the same time. A file can only exist in a single location, and because of that there's only a single categorization possible, even though multiple classifications can make sense [1]. When we come back to this issue later, We will see it lies at the root of all other problems and this will be the first problem we tackle later on by introducing the concept of facets to get aroud the single-categorization issue.

This inability is an unnecessary artifact from the paper-based world where a piece of paper can really only exist in one place. There is no need for such a limitation in the digital age, and there should be nothing from stopping a file from belonging to multiple folders if that makes sense. It has to be noted that just as in the real world, creating copies is not a solution because if one copy of the file changes, the others will not.

This first issue has so many consequences because, to repeat a previous statement, it means that there is only one hierarchy, or categorization possible. There are many different angles from how we can look at files and for each type of media this angle will be different. For example, with photos we might want to organize pictures by year, by event or by place. With music we might organize by artist/album or composer/album. However, we are limited to choosing only one classification, or facet, to store the pictures, music and all our other files. Folders are created to represent a particular (trail of) facets but only one representation is possible at the same time.

It may be useful at this stage to explain some filesystem-related terminology. In doing so, I will also argue against a common-heard argument that states that the capabilities I am looking for are already present in current filesystems. When we talk about a file we mean both its contents on the disk (a series of bits) and the label or path (a string) the filesystem uses to identify this file. This way, there cannot be two paths pointing to the same file. Unix systems have the notion of hard links and soft links.

A hard link is what I described above but in Unix systems a user can create multiple hard links pointing to the same file. Another way to look at it is that several labels point to one series of bits on disk. As such, when an application saves to one of these hard links, it will appear to the user that these other files have also changed. The actual contents on disk are only removed when all the hard links pointing to it are also removed. When you delete only one hard link pointing to a file on disk that has two or more hard links associated with it, the contents on disk will remain.

The other kind of links are soft links. Instead of pointing to the contents on disk, they point to the label of another file. Consequentially, when the original file is moved the link is broken. Soft links are much like Aliasses on the Mac or Shortcuts in Windows.

The common-heard argument I mentioned earlier is of course about these kind of links, and hard links in particular because it seems that we can actually have multiple representations of the same file on disk. However I have t disagree with this argument. The first reason is that they are scarcily used in practice even though people know of their existence. Soft links of course have the big problem that when the original file is moved, the link breaks and you won't know this until the system at a later point fails to open the soft link. Hard links have the problem that they will make it more difficult to keep track of what files still exist on disk somewhere. It is easy to forget to remove one hard link in some remote folder. You may be under the impression that the file is removed but disk space is still being consumed. However the main issue with both of these is the fact that they are hard to manage. They work well if you for example collect aliasses to often-used files on the desktop. They will not work well if you want to represent the different categorizations (place or event) that may exist for a few thousand photographs. If this could ever work, for sure it would have to be done and maintained by our filesystem automatically. The way to accomplish it currently is too labour intensive, prone to error and requires a lot of maintenance and time.

2.2. Files are identified by a filename

Files are identified by a filename and the folder they are kept in, but this name alone does not show enough information about a file. For example, a picture will have been taken at a specific location during a specific event and there might be several people on it. Nobody would put all this information in the name of the file though, even though it might be very useful to do so as a means to idenfify this particular picture on disk.

By locating the file, I mean that the computer can distill useful information about the file, to aid us in our search queries for example. In case of a picture, we as people can immediately see what people are on it, but a computer can't. When searching for 'Diane', a computer cannot look at the pictures itself and recognise Diane, unless some form of facial recognition is used which then has to be stored in the metadata of the file [7]. The file may very well be labeled something like IMG_123.jpg but this filename is completely uninformative of anything, it has only been chosen by your digital camera to quarantee uniqueness.

2.3. Choosing a location is overhead

Related to the previous problem is that choosing a location for a file is also unnecessary overhead. We are all familiar with this problem when we first save a document and we have to choose a name for the file. Very often the filename is either useless for retrieval purposes or redundant. An example: let's say we are keeping digital copies of a magazine like this: magazine-name/year/month/file.pdf. Finding this file using a normal search is only possible if we include the magazine-name, the year and the month in the filename. All of this is redundant information because it's already made implicit in our directory structure.

Also, as I noted earlier, it is important to keep in mind that a folder structure only conveys one particular kind of ordering, out of the many available ones.

2.4. Representing Facets

Facets were already mentioned when I talked about the first problem with hierarchical filesystems. In my opinion, proper handling of facets is an essential requirement for any system to succeed [10]. For instance, a website selling jewelry will most likely present the user with different ways of browsing its inventory. A potential customer could be searching for a nice ring, or he might be interested in something silver. The material (silver or gold) and the kind (ring or necklace) are different facets of the jewelry and at different times, people may be interested in browsing by either facet [11].

The same point can be made about digital files. Indeed we will see that this idea is already present in third-party applications and I think the point is important enough to be stated here separately because the fundamental problem with current hierarchical filesystems is that only one facet can be represented at the same time. Taking the example of jewelry again, we can create subfolders for kind (necklace or ring) or subfolders for material (silver or gold), but not both because the file representing a ring can only exist in one folder. We don't want to create duplicate copies for obvious reasons and playing around with aliasses will most likely end up with a mess.

We can say the same about music; facets can be artist, album and genre or with photos; facets can be place, event or people. The same principle applies and it will be even more obvious that playing around with aliasses won't work. We have thousands of photos to manage and aliasses don't scale up to this level.

With this new terminology in mind, we see that the problems I stated earlier are all concerned with a lack of flexibility and most importantly, the inability to represent multiple facets in a logical and easy way.

3. THIRD-PARTY APPLICATIONS

When discussing some of the above issues I used the example of photos or music files. This may seem weird to some people because we usually don't primarily use our filesystem to organize these kind of files [13] [14]. A lot of people use special applications for managing our photos or our music so one might regard these as just bad examples, stating non-existent problems. In these applications we don't have these aforementioned problems and limitations like naming a file or choosing a location.

I am arguing the contrary, these are not bad examples. These are quite excellent examples, not despite, but because these files are usually handled by special applications. The fact that they are means that the filesystem is bad at managing these and that these applications fill a void that cannot be filled by the filesystem itself. After all, what use is a filesystem if it is inadequately suited to help us organize our important files.

People taking the time to learn how these applications work is a testimony to the fact that the filesystem lacks in this department. There must be something these dedicated applications offer that the filesystem does not. We can use applications like iTunes to organize our music or iPhoto or Picasa [12] to organize our digital photographs but in the end they all just manage files. Can we learn from these applications to enhance our filesystem? In my opinion this is a rhetorical question and I will argue that we certainly can.

3.1. Separating Location and Representation

The first characteristic of these applications is that they take care of storage for you. Somewhere on your disk they maintain the files and present them to you in an interface suited for the kind of files you are dealing with. Often you can create albums, playlists or smilar metaphors which let you put a file in multiple places. For example, in iPhoto '09, photos are organized by events, places (their location) or faces (using facial recognition to identify people on the pictures). These can indeed be seen as different facets of the same data. Every picture will have all these facets, while the actual file only exists in one location on the disk.

If we look at this characteristic from a higher perspective we observe that there's a fundamental difference between these applications and the normal filesystem. These applications will figure out where to put these files themselves without the user having to bother with that. Essentially, they achieve a separation of a file's location and file's representation. The file itself only exists in one place on the disk but can be represented in different facets.

The filesystem itself does not have this distinction. The location of the file and the representation of the file are the same. This makes it therefore impossible to present a file in different places since its representation has a one-to-one link with its physical representation. You can only view the file in the folder where it is kept. Some may argue that the filesystem itself is just a representation and that storage is handled on the hardware level; one could use symbolic links and hard links to create these facets but as I argued before, this functionality is hardly ever used. From the user's perspective, the unique path of a file is both its location and its representation.

Previously I have stated that the ability to present a file in multiple facets should be an important requirement of our system. The only way to achieve this, without storing a document multiple times on disk is to detach location from representation. While I haven't ever found this distinction explicitly being made, there are a few filesystem-replacement projects that leverage exactly this separation [4] [5], which further lets me belief that this is important. As a sidenote, the Lifestreams [4] system focusses on a single facet (time) and postulates that as the most important ordering mechanism. One of the reasons I think this system will (and has) never succeed(ed) is because it is too limiting and forces all your documents into a single facet which may not even suit the kind of documents a user is working with. The user is again forced in some kind of rigidity which is, albeit disguised as being more flexible, equally unwanted as the current hierarchical filesystem.

Let's look back at the hierarchical filesystem-problems I noted earlier. The first issue in particular (a file can only exist in one location), can be restated: the single-location issue isn't the real problem, but the single way of representation is, and hence the inablity to represent multiple facets. Only if we separate location and representation, we have the possibility to solve this problem.

Coming back to this idea of separating location and representation, we observe that these third-party applications are all concerned with the representation while they handle the storage for you. If we would devise a good system, we should do the same. Storage should be something the system does without needing any input from the user such as choosing a destination folder or a filename. The first priority for a user should be to save the file to disk, later he or she can concern himself with a specific categorization.

3.2. Organizing by Metadata and Tags

The benefit these third-party applications have is that they have been specifically designed to handle a certain kind of files and they know (or at least think think they do) what categories make sense. It is at the core their usage of the metadata associated with these files that allows them to present multiple facets to the user. Faces and places for photos or album name and artist are all good examples of relevant metadata for these files. Metadata is used here to cover the different facets that

make sense. When files are no longer identified by their unique path, the only way to distinguish one file from the other is using metadata.

Pictures taken with digital cameras often have these obscure names like IMG_123.jpg. Nobody ever bothers to change this name to something more meaningful and still we are able to find the pictures we need, using the metadata and the facets our photo-management application offers us. For music we could make the same case, where songs can be named 'Track 1' instead of conveying something useful.

How then is this metadata structured? If we take iTunes as an example, we see its metadata is divided into useful facets like 'artist' or 'album'. These values can be changed for every individual file, which gives the user a lot of flexibility. For example, one file can have 'Mozart' as a composer while another can have 'Mozart' as the artist without the system protesting.

4. PRELIMINARY CONCLUSIONS

We have identified two key aspects that our new filesystem should support. First, this system should manage the storage of files for us. Second, it must use metadata to describe and identify files.

4.1. Handling Storage

In traditional hierarchical filesystems, when we save a file, we want to save it in a place we can find it back later and we use some kind of ordering that makes sense, choosing one of the relevant facets and discarding the other facets. Instead we should let the system save a file and independently from that, define those facets that make sense to us. How our system exactly handles this storage should be of no concern to the user. Any storage system that makes sense could be used.

Traditionally, saving a file is done by giving the file a name and choosing some folder to save it in. However, as we noted in hierarchical filesystems, a filename is either too restrictive or redundant and a folder structure is too limited because it only covers one trail of facets. The aforementioned dedicated applications do not even use the filenames to describe files, IMG_123.jpg for photos is a good example here. For them a filename is something that is part of the location-layer, not the representation-layer. Since we want to follow the same approach, we too should remove the need to name a file as well as the need to choose a location for the file.

4.2. Handling Metadata

As said before, when files can no longer be identified by their unique path, the only way to distinguish one file from the other is using metadata. The question of how to structure metadata therefore becomes very important. We already say how these third-party applications structure their metadata, but a lot of research has been done on the subject, and the proposed solutions are very different. I think it is therefore appropriate to take a good look at this research and to see what way to structure metadata would suit us best.

5. HOW TO STRUCTURE METADATA

Metadata can take the form of tags, the last one has been popularized by online communities such as Flickr [15] and Delicious [16]. Many papers acknowledge that metadata is important to find a better hierarchical filesystem-alternative [10] [17]. Structuring this metadata as a list of tags is an often-heared solution [1] [13] [18].

5.1. Adding metadata with tags

Tags are keywords describing the contents or context of a file. A user can choose any combination of keywords and the obvious benefit is the freedom a user has. The user can choose his own keywords and describe whatever facet he sees fit. 'Gold' and 'necklace' could be two tags describing the aforementioned jewelry. Whether a user is organizing pictures, music or files in general, all these types of files can be annotated with metadata and tags seem to be perfectly suited for this.

As a side note, the complete path of a file could very well translate to the tags that are applicable to this file. If we take the magazine-ordering system, we observe that each directory in the path is actually a tag by itself. For example, magazine-name/year/month/file.pdf would translate to { 'magazine-name', 'year', 'month' }. This gives us a very reasonable set of tags to identify a particular issue of a magazine. With this directory structure we've only captured one facet of ordering, one by date, but another way could be ordering by topic for example, which we could also capture with tags.

Online, these tags are primarily used to aid a search query, but searching is not always enough, Cutting, Karger, Pedersen and Tukey in their paper, "Scatter/gather: A cluster-based approach to browsing large document collections" [21] have some excellent points on this subject. Often, you might not be searching for something in particular but you might want an overview of what you have on a particular subject.

Taking music as an example, we might want to view all albums we have of a particular composer. A flat list of tags will present a problem here because there is no way of telling what a single tag stands for, is it the artist, the album name or the composer?

5.2. Structuring metadata

There are two problems with the previously described simple tag system. The first problem is that when there's no ordering in these tags, identifying the proper tags becomes increasingly difficult when the list of all tags grows to a few hundred, or potentially more, items. There will be no way of telling if tags are concerned with the same subject or if there's some other kind of ordering. The second problem is that there's no way of telling what these tags stand for. As I claimed earlier, I think it is important that a system is able to describe different facets of a file and while tags can describe these, there's no way the system is able to magically know that 'gold' and 'silver' are actually instances of the same facet while 'gold' and 'necklace' are not. In short, we need semantic information about the tags, as previous research has also indicated.

When one looks for scientific studies trying to structure metadata or tags in general, one undoubtedly ends up finding a lot of research on the topic of creating ontologies, tree-like structures describing the the metadata in a filesystem ([9], [17], [18] and more). This should then both provide a kind of ordering and provide semantic information about tags making it possible to classify 'gold' and 'silver' as being related.

It is certainly not a new idea to classify or group tags, using for example a system such as WordNet [20]. This usually takes the form of some kind of a tree-like structure that represents an is-a relationship between tags [17] [18]. However I will argue that, even ignoring the fact that it may be considered overkill for a personal filesystem, it has some major deficiencies and as such is completely unsuited for our needs.

5.3. Structuring metadata using ontologies

The benefits of having a semantic tree describing our tags are obvious. When we know that 'Bob Dylan' is in fact an artist we can be much more specific in our queries. While such a tree structure has its benefits, it will have to be created automatically or it might be too much work for the average user. This is however not the main issue.

Researchers have been fascinated with creating ontologies for many years (I already mentioned a few, like [9],[17],[18],[19])

which are supposed to classify every relevant entity. Bringing this concept 'down' to the level of a desktop filesystem presents some difficulties in my opinion. There does not seem to be much literature to support my claim, but I will argue here why I think this is the case. One paper that comes close is [10].

What a system thinks the correct classification is may well be perfect for some person and completely inaccurate for another person and we personally may know things or relationships between things the system cannot ever know. John Doe can be a friend of ours so we can classify the tag 'John Doe' as being a friend but unless we tell the system so, it will never be able to properly classify him. While a very liberal system may well be able to cope with this problem, even these problems are not the main issue.

What an hierarchical tag-tree (possibly based on some ontology) cannot possibly capture is that different contexts can give different meaning to the same tag. In one context 'John Doe' may be classified as a writer (because the system knows he wrote several books), but in another context we might have a video of a presentation he gave on a conference where you would consider him to be the presenter, not the writer. Similar examples can be found in other domains but the key point is that a single tag can mean different things in different contexts.

Let's suppose I am a football lover and I also happen to be a web-developer, the keyword Ajax can mean two completely different things in my system. I may have been to a match, taken some pictures and tagged them with the word Ajax. Also I may have tagged a few development-projects of mine with the same tag. When I'm looking for Ajax, as in the football club, I will not be interested in my development projects. One last thing I will say on this subject is that ontologies change. What used to be Eastern-Europe is now part of the European Union and Yugoslavia does not even exist anymore. Things change, and ontologies are bad in coping with this change.

We can only conclude that a tree-like structure based on some ontology is inadequate. It provided us with the classification we needed, but we also end up with a rigid structure and rigidity lies at the heart of hierarchical filesystems, something we were trying to get away from. If our main criticism of (possibly ontology-based) hierarchical tags lies in the fact that it is bad in dealing with different contexts we have to take a fundamentally different approach. Every 'global' system of truth we devise to give meaning to tags will suffer from the same flaws.

5.4. Structuring metadata on a per-file basis

We must realize that there is not a single truth and that we cannot have a single tag-ordering system for our entire filesystem. The solution to this is of course to keep both tags and their meaning on a per-file basis. Metadata like 'writer=John Doe' should be applied to one file and 'presenter=John Doe' can be applied to another file. For one file, Ruby can be set to be a gem and for another file it can be set to be a programming language. When the system is asked to display all files that have Ruby as their language, files about Ruby as a gem will be left out.

If this sounds as being very familiar, you would be right. It is the solution that is used by third-party applications like iTunes and it basically is a dictionary of key-value pairs per individual file (sometimes better known as a hash table). One could wonder if we are still talking about Tags, but basically it is all a variation on the same idea. A key-value pair is conceptually the same as a Tag with a name (the value) and a meaning given to it (the key) and could even be considered as tuples. However choosing the right representation for this in the storage-layer is irrelevant for the user. There is one thing we have to add to this idea. We do not want a strict key-value ordering because there can be multiple values for the same key. When applying tags for the keywords in scientific papers, we will need multiple tags with the key being 'keyword', one will not suffice. The same can be said of people in photographs or albums in music files.

In iTunes however, a user can only change the value of the key 'artist' for example, but not the key itself. As an anology, our system will also allow the user to define custom key. This is because we don't only handle music files or videos, but we invite the user to order all his information in our system. As such, we don't know what keys make sense. I believe it turns out that the solution researchers have been looking for so long has already been found and utilized by the third-party applications. The only thing we need to do is to add just a more flexibility as to what keys the user can define.

6. A SOLUTION

Earlier I already defined the two most important characteristics of our system: a separation between location and representation, and a focus on metadata. In the previous chapter we found a simple answer to the question of how to structure metadata. The main problem was that tags can have multiple meanings but still it was needed to give semantic information to these tags.

As the reader may have observed, the proposed system is primarily concerned with providing the user a useful interface. With this in mind, our 'filesystem' could very well be solved on an application level. Our filesystem could be a stand-alone application just like these third-party applications. It will make adaptation a lot easier since there will be no problem with legacy support.

The proposed system is still a very abstract one. To better understand the concept it may be useful to visualize the system as a flat collection of files, with a table for metadata attached to each file. These tables would have two colums. The first column lists the different keys like 'album', 'artist' or 'author'. The second column displays the value, or tags given to a file. For example, 'Mozart' would be in the second column and the first column in the same row would say 'composer'.

Another way to view our system is in terms of a database. This database would consist of only one large table. The rows in this table represent the files and the columns represent the keys such as 'album' or 'author'. As the user starts to add tags, new columns will be added to the table. Whether this is an efficient means of storage remains to be seen since most cells in the table will remain empty. Indeed only a small subset of files will have an 'author'-tag defined and the same goes for every other tag. It is however a nice way to visualize the system. The different views we can have on the data then easily translate to SQL queries. For example, a useful view could for example give the user all albums of a particular composer and this easily translates in an SQL query if the have the database structure I proposed earlier.

One may wonder if there is anythig new as such in the system I propose. Databases have been designed with exactly these capabilities in mind so couldn't the entire contents of this paper be replaced with the sentence 'we will use a database instead'? It will not come as a surprise that I disagree with this. Before we postulate something as the ultimate answer we should reason as to why it is so; starting with the acknowledgement that we need to represent the different facets of files, to tags that should have semantic meaning by themselves. These can be seen as the requirements for our system. Luckily we can fulfill these requirements adapting already existing systems. As a sidenote, because databases are so flexible, I suspect that whatever solution we would have come up with, it would be possible to implement it on top of a database.

The idea of experessively using a database to organize a filesystem is not new. There is one paper [22] that came very close to what I proposed in this paper. In the end however it

represented the different facets using tags but not on a per-file basis. The reader will understand the fundamental difference compared to my solution.

Since the previous chapter outlined a solution for the problems, we need to test the idea against the requirements we stated earlier; the problems with current hierarchical filesystems. How does our per-file tagging system stand up to this?

6.1. A unique path to a file

We borrowed the idea of the storage-layer being responsible for storage from the aforementioned third-party applications. Using the same model as they do, we will have solved the problem of storing multiple copies of the same file. When multiple categorizations make sense, we can use the metadata we keep for each file to present the user with an appropriate view.
For example, when we record 'events', 'faces' and 'places' as tags it would be possible for the system to present all tags labeled as being and 'event' or a 'place' and of each of these tags find all the appropriate images.

6.2. Files are identified by a filename

Filenames are currently being used for recognition-purposes and for uniquely identifying a file on disk. File-recognition is handled in our system by the collection of tags applied to a file with which we can capture all facets of a file, not only a subset. As such, the only remaining function of a filename is for storage purposes, which is now handled by the system independent of the user. Thus, we are no longer concerned with filenames at all.

6.3. Choosing a location is overhead

A file path often captures a single facet of a group of files (music classified by artist for example). We no longer have to choose what facet makes the most sense because all this information will be contained in the file's tags. The need to choose an appropriate directory structure disappears completely because the user is no longer in charge of storing the file. All problems with searching and single-categorization hence disappear.

Bloehdorn and Völkel in their paper "TagsFS" described the problem in the context of organizing scientific papers using keywords as the organizational-facet. Over time two separate directories may grow: ir/semweb and semweb/ir. When we look in one of these folders, we will never see the other folder while there could be interesting papers there. Both keywords make sense, but we cannot see the contents of two folders at the same time in one place. Instead if we would have used the filesystem proposed in this paper and we would have implemented it in a database, a view could use a simple SQL query to collect papers with both keywords. In traditional filesystem is however fundamentally impossible because location and representation are not separated in hierarchical filesystems; we cannot easily collect the contents of two folders in one view.

Our system however, will be perfectly capable to do so using a few simple operations because we have separated location from representation. The other issue with choosing a location, namely the duplicty between a filename and the path will also disappear because our system replaces both with tags describing these characteristics in metadata.

7. CONCLUSION

My research leads to the conclusion that the problems with hierarchical filesystems can mostly be brought down to its inability to separate between *location* and *representation*. By location, I mean a file's location on disk, identified by a unique path-name and by representation I mean the way and the context in which a file is presented to the user. In a traditional filesystem these two are the same.

Some may note that current filesystems already have this separation, namely the logical address versus the physical address. This is however a one-to-one releationship and not a separation in the way this paper has defined it.

In finding a suitable alternative to the traditional (hierarchical) filesystems, an important criteria was the ability of the system to deal with different facets of a file. Third-party applications for managing music or photos use exactly this idea to present the user with relevant views like 'events', 'faces' and 'places' which are all useful facets of digital photographs. Albums or playlists fall in the same category and are conceptually just facets.

In our proposed filesystem we need to achieve a separation between location and representation. The system should be concerned with naming a file and choosing a location for it and metadata should be used to identify a file since a filename and a file's location will have lost their meaning.

Metadata should be kept on a per-file basis and not in using some kind of ontology-based tree-like structure as what many papers are proposing. The main criticism to this approach is that different tags can mean different things in different contexts and when there is a single ordering, this system will break down. Keeping metadata on a per-file basis is the only thing that can work. Looking at how these third-party applications store their data, we discovered that this is exactly what they are doing. In iTunes for example, metadata can be edited for every individual file for optimal flexibility while iTunes at the same time uses this information collect all 'artists' and 'albums'.

Our system will be keeping a dictionary for every individual file and any graphical view can be laid out on top of it to present an appropriate view. Storing, naming and locating files on disk is then handled by the system itself. The solution as presented here is in concept a simple one, one that is already being used by the third-party applications we discussed. With this, we will have created a system that separates location from representation and which has optimal flexibility because it acknowledges that files have different facets, all of which can be described with ease.

8. FUTURE WORK

Traditional hierarchical fileystems are as we know organizated around folders containing subfolders. The way files are presented in the user-interface is also done using the same metaphor.

Our new filesystems has no concept of folders and will have to serve the user a different kind of interface. The interface could maybe be very customizable and an easy way will have to be found to enter the all-important metadata.

No doubt applications like iTunes, iPhoto, Picasa and others can serve as inspiration as to what kinds of interfaces the user could comfortably work with. This paper outlined the basic concepts of such a filesystem. Future research could be conducted to provide a user-interface on top of this foundation.

When the idea presented in this paper is implemented on the application level as a stand-alone application, the traditional filesystem will still sit under our system. There might be interesting ways to combine both concepts.

9. REFERENCES

[1] S. Bloehdorn and M. Volkel, "TagFS—tag semantics for hierarchical file systems". in Proc of the 6th International Conference on Knowledge Management (I-KNOW 06), 2006.

[2] D. K. Gifford, P. Jouvelot, M. A. Sheldon and J. W. O'Toole. Semantic file systems. ACM SIGOPS

Operating Systems Review, 1991 vol. 25 (5) pp. 16-25

[3] S. Fertig, E. Freeman and D. Gelernter. “Lifestreams: an alternative to the desktop metaphor”. Conference on Human Factors in Computing Systems, 1996 pp. 410-411

[4] E. Freeman and D. Gelernter. “Lifestreams: A storage model for personal data”. SIGMOD Record, 1996 vol. 25 (1)

[5] A. Agarawala. “Enriching the Desktop Metaphor with Physics, Piles and the Pen”. Honeybrown.ca, 2006

[6] iTunes 8. Apple Inc, 2009 (http://www.apple.com/itunes)

[7] iPhoto ‘09. Apple Inc, 2009 (http://www.apple.com/iphoto)

[8] C. Soules and G. Ganger. “Why can’t I find my files? New methods for automating attribute assignment.” Proc. Ninth workshop on Hot Topics in Operating Systems, USENIX Association, May 2003.

[9] H. Ngo, C. Bac, F. Silber-Chaussumier and Q. Le. “Towards ontology-based semantic file systems.” Proceedings of the 5th International Conference on Research, 2007

[10] W. Jones, A. Phuwanartnurak, R. Gill and H. Bruce. “Don't take my folders away!: organizing personal information to get Things done”. Conference on Human Factors in Computing Systems, 2005, pp. 1505 - 1508

[11] S. Henderson. “Genre, task, topic and time: facets of personal digital document management”. Proceedings of the 6th ACM SIGCHI New Zealand chapter's international conference on Computer-human interaction: making CHI natural, 2005, pp. 75-82

[12] Google Picasa http://www.google.com/picasa (Visited March 17, 2009)

[13] A. Girgensohn, J. Adcock, M. Cooper, J. Foote and L. Wilcox. “Simplifying the management of large photo collections”. Proc. Human-Computer Interaction pp. 196-203

[14] K. Rodden and K. Wood. “How do people manage their digital photographs?” Proceedings of the SIGCHI conference on Human factors in computing systems, 2003, pp. 409-416

[15] Flickr: http://www.flickr.com (visited March 4, 2009)

[16] Delicious: http://delicious.com/ (visited May 18th, 2009)

[17] E. Stoica, M. A. Hearst and M. Richardson. “Automating creation of hierarchical faceted metadata structures”. Proceedings of NAACL HLT, 2007, pp. 244–251

[18] C. Van Damme, M. Hepp and K. Siorpaes. “Folksontology: An integrated approach for turning folksonomies into ontologies. Bridging the Gap between Semantic Web and Web”, 2007, vol. 2 pp. 57–70

[19] H. Ngo, C. Bac and F. Silber-Chaussumier. “Enhancing Personal File Retrieval in Semantic File Systems with Tag-Based Context”. www-inf.it-sudparis.eu, 2008

[20] Wordnet. http://wordnet.princeton.edu/ (visited March 4, 2008)

[21] D. Cutting, D. Karger, J. Pedersen and J. Tukey. “Scatter/gather: A cluster-based approach to browsing large document collections”. Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval, 1992, pp. 318-329

[22] O. Gorter. “Database file system: An alternative to hierarchy based file systems”. Master thesis, University of Twente, 2004