Proceedings of the 2nd international Petascale Data Storage Workshop (PDSW '07)
held during Supercomputing '07. November 11, 2007, Reno, NV.

Searching and Navigating Petabyte-Scale File Systems Based on Facets

Jonathan Koren
jonathan@soe.ucsc.edu

Yi Zhang
yiz@soe.ucsc.edu

Sasha Ames
sasha@soe.ucsc.edu

Andrew Leung
aleung@soe.ucsc.edu

Carlos Maltzahn
carlosm@soe.ucsc.edu

Ethan Miller
elm@soe.ucsc.edu

Jack Baskin School of Engineering
University of California Santa Cruz
Santa Cruz, CA 95032, USA

ABSTRACT

As users interact with file systems of ever increasing size, it is becoming more difficult for them to familiarize themselves with the entire contents of the file system. In petabyte-scale systems, users must navigate a pool of billions of shared files in order to find the information they are looking for. One way to help alleviate this problem is to integrate navigation and search into a common framework.

One such method is faceted search. This method originated within the information retrieval community, and has proved popular for navigating large repositories, such as those in e-commerce sites and digital libraries. This paper introduces faceted search and outlines several current research directions in adapting faceted search techniques to petabyte-scale file systems.

Categories and Subject Descriptors

D.4.3 [**File Systems Management**]: File organization; E.5 [**Files**]: Sorting/searching; H.3.3 [**Information Search and Retrieval**]: Search process

General Terms

design, management

Keywords

faceted search, information retrieval, petabyte-scale storage, enterprise search, metadata, semantic file system, virtual directory

1. INTRODUCTION

As storage capacities increase, organizations have, and will continue to, take advantage of the situation to store larger and more diverse amounts of data. While these repositories are useful, they impose challenges. Organizing billions of shared files is difficult, especially when considering the differing needs of users. Furthermore, it can be difficult for a single user to know what is even in the repository, let alone where to find it. An attractive method to solving this dilemma is to make search an integral part of file system.

Traditional keyword and metadata search is most effective when a user can correctly express what he/she is searching for with a few key terms. But what about when the user cannot express him/herself clearly or effectively? What if the user is not even aware of what he/she is searching for in the first place? In these scenarios, the user must explore the search space, without being overwhelmed by the size of the space. Traditional directory hierarchies are not a viable solution to this problem as it presupposes that the user is already familiar with where the files he/she is searching for are stored and how they are named. This problem is not unique to file systems. A similar problem exists in the information retrieval (IR) community. One method used in IR to solve this problem is faceted search.

2. SEARCH AS NAVIGATION

Traditionally, file systems have been organized as a single monolithic hierarchy. While this paradigm has served well, it does suffer from a major drawback. Each file exists in a specific location. Even with the inclusion of symbolic and hard links, files tend to have only a few specific locations and names. As the diversity of the files increase it becomes more difficult to fit each file into a specific location. Complicating matters, often there is not one "right" answer, but rather multiple, equally reasonable answers. Deciding which one to use, is often a rather arbitrary process [12]. This can lead to confusion when a user believes that one of the other possibilities is somehow "more correct."

One solution to solving this problem is to simply avoid imposing a hierarchy on the files, and instead allow users to search for the files they are interested in. While conventional keyword search queries no doubt have their place, they are not very conducive to exploration, since they require the user to possess some knowledge about the contents of the file system and relevant files in order to issue effective queries. What is needed is a technique that converts

Supercomputing'07 Nov. 10-16, 2007, Reno, NV

search for a series of query-responses into a more interactive browsing scenario. One such technique that has gained popularity, especially with large data repositories, libraries, and e-commerce sites, is *faceted search*.

Faceted search is built on the idea of *faceted metadata*. Faceted metadata are sets of key-value pairs associated with a file. The keys, called "facets," allow the values to be grouped into semantically meaningful ways. Each file has multiple facets associated with it, and can have multiple values associated with each facet. For example, an MP3 music file could have facets such as *title*, *artist*, and *album*, in addition to traditional POSIX metadata such as *owner*, *group*, *file size*, and *access time*.

In faceted search, users are presented with a list of facets associated with the files stored in the file system, along with some of the values associated with each facet. By presenting multiple facets and their values simultaneously, the user is shown multiple parallel categorization schemes. By arbitrarily combining these facet-value pairs, users can formulate effective search queries without any prior knowledge of the contents of the system being searched. Each time a user adds or removes a facet-value pair to/from his/her query, the list of results changes, as does the facet-value pairs that are available to the user for refining the search. By updating the facet-value list as dynamic taxonomies, the user is never presented with options that can lead to empty results.

Since it is the combination of metadata that describes the location of a file, there can exist multiple equally valid "paths" to the file, thereby enabling the files to be organized and reorganized according to the current needs of the user instead of in a single one-size-fits-all approach. Extending this analogy, one can think of the current query as being the current working directory, with the results of the query being the contents of the directory.

3. CREATING AND INDEXING FACETED METADATA

In order to use faceted metadata for navigation, we must make two assumptions. First, that the metadata exists in some form and second, that the metadata can be stored in some easily retrievable form. This first assumption is self-evident. The second assumption is most commonly satisfied in both the information retrieval and file system communities through the use of an inverted index. In the simplest form, an inverted index maintains a table that maps pieces of metadata to the files that contain that metadata. Information is loaded into the index through the use of *indexers*[1], which use special routines, called parsers, that understand the internal structure of specific file types, and can therefore extract useful features from the files.

Currently, many deployed faceted search systems rely on structured data with some manual metadata annotations. This is not a reasonable solution when dealing with billions of files. Research into automatically extracting useful faceted metadata from existing files is on going within both the information retrieval and semantic file system communities.

The first source of faceted metadata about a file is the structured metadata that is already associated with a file. For example, MP3 music files often have metadata such as *title*, *artist*, and *album* stored within them. By using indexers, the metadata can be re-indexed automatically whenever the file is modified.

Unfortunately, not all files contain such conveniently formatted metadata. In many files, the relevant metadata is embedded in the content of the file. One focus of current IR research is to extract faceted metadata from unstructured text. One such project is Castanet[19]. Castanet reads small snippets of text that are associated with specific files (e.g. document titles), and combines these terms with WordNet[6], a thesaurus that lists synonyms and hypernyms of English words, to create multiple facets that can be used to describe the document collection.

Both of the previous approaches assume that the files are indexable, that is useful information can be easily and reliably extracted from the contents of the files. This is not true for many types of files. Numerous proprietary data formats are essentially opaque outside of the originating application. Ideally, an interface between applications and the file system would be utilized to allow applications to directly store relevant metadata about their files. Failing that, user supplied metadata must be used to supplement what can be gathered from the files themselves.

Graffiti[14] is an experimental client-server application that allows individual users to tag files with short descriptive keywords. These tags can then be combined with the tags from other users in order to help organize and manage the file system. By being a collaborative system, the work of annotating files becomes distributed. Multiple opinions can encourage a richer and more inclusive description of the files. Given these features, a collaborative system is believed to be fundamental to the successful employment of user annotated metadata in any large scale system.

While not directly extracting metadata, research into monitoring file access[18] and interprocess communication[17] patterns have been used to determine how closely related files are. With this information, it is possible to allow the metadata that is associated with one file to propagate to the other related, but unannotated files.

4. SYSTEMS SUPPORT FOR FACETED METADATA

The most closely related work to faceted search in the file system community has been with semantic file systems. The common ability shared by these systems is a means to query for groups of files within the file systems interface. The Semantic File System [9] contained its own B-Tree based structures to store key-value pairs within the file system, and utilized *transducers* to assign these pairs automatically based on file content. Be File System [8] followed suit with key-value pairs and B-Tree storage, but added a non-POSIX compliant query interface that contained logical operations for matching and range queries. The Inversion File System [15] contained an interface for SQL style queries for

[1]Indexers are also known as transducers in the semantic file system community.

files based on their metadata, but only added one extended attribute for file type. It stored its metadata in an embedded POSTGRES database. The Logic File System [16] integrated a simple query language with into the POSIX interface for searching for files based on tags. It used Berkeley DB for its metadata store. Other systems, such as Spotlight [3], Google Desktop Search [10], and Beagle [4], place both the search interface and indexing along side the file system, either requiring frequent reindexing or a notification mechanism from the file system to keep its indices up-to-date with file system changes.

The Linking File System (LiFS) [2] also stored its metadata within the file system itself. but did not contain indexing structures or a working query interface (a simple one was proposed). Like the other file systems, LiFS allowed arbitrary key-value pairs to be assigned to files, but also allowed files to be associated together through *relational* links. Each link, in turn, could also have multiple key-value pairs associated with it. These features allowed for richer metadata such as interfile relationships and data provenance to be expressed.

ViewFS follows the LiFS metadata model and integrates traditional file system interaction with both keyword and faceted search [1]. ViewFS accomplishes this by modifying existing POSIX file system interface to support search queries as file and directory names. Queries are specified in a path-based language called QUASAR. Queries are first-class file system objects in the form of virtual directories. The name of such a virtual directory specifies the query and the content of the directory represent the results of the query. Modifications to the file system are immediately reflected in the contents of virtual directories. This immediate reflection of file system modifications requires a tight coupling of the search functionality with the file system so that updates in the file system are efficiently forwarded to the indexing and virtual directory update components.

To enable faceted search applications, ViewFS and QUASAR feature some key functionality. Obviously, it should provide term-based search over faceted metadata. This feature we carry over from previous semantic file systems. Secondly, paths in QUASAR should contain a number of facets so tuples of metadata values for each file shall be returned. We adopt this feature from SQL. Finally, we improve existing search through supporting the return of browseable dynamic virtual directory hierarchies. This improves searching, since such organized results may be presented directly to the user, as opposed to dealing with long lists of results. We know of no other system at present that enables browsing and search using this abstraction. There are other novel features of the QUASAR query language, including relationship-based search and location-based search. We don't discuss them in this paper because they would not be utilized by our current faceted search interface design. Nonetheless, we may work to incorporate such features into future interfaces that utilize faceted metadata.

4.1 Petabyte-Scale Indexing

In addition to building search into the file system interface, there must be facilities that allow the indexing system to scale with system size. An index for a petabyte-scale file system must handle billions of files and the many tags, attributes, facets, and links for each of these files. Moreover, an index for a petabyte-scale file system must handle thousands of queries per second, given the large number of users for such a file system.

For petabyte-scale file systems, the indexing problem is closer to that of web indices than those found in desktop search tools, such as Apple's Spotlight [3], Google's Desktop Search [10], or Beagle for Linux [4] because of the scale of the index. Google's uses GoogleFS [7] as an underlying storage system for their approach to large-scale indexing, which handle the large scale and query rate that petascale faceted search requires; other web-scale systems such as that used by Yahoo! adopt similar approaches. However, these approaches have several major shortcomings that make it unsuitable for use in file system-based faceted search. First, these approaches can only be udpated slowly, typically requiring latencies of hours to incorporate new information into the index. Google uses a separate, much smaller index to handle rapidly-changing news sites such as CNN, but their approach does not scale to the billions of files in a petabyte-scale file system. Second, web indexes have a single facet, typically text-based content; petabyte-scale file systems will have many types of files, each of which has a features in a number of facets. Some facets, such as provenance, are far more ambiguous, in contrast to hyper-link relationships; others, such as information about files linked to particular file, are larger and more context-sensitive than text-based features. Thus, faceted search for petabyte-scale storage cannot use existing techniques.

We are considering several approaches for indexes to support faceted search in very large file systems. To solve the problem of immediate update, we are considering the use of multiple levels of indices. The "top" level changes relatively slowly, producing results that may be an hour or two old. The system will also maintain smaller (and less efficient) indices across the entire file system to allow searches to cover more recently-changed metadata. Periodically, the system will merge the two indices, keeping the top-level index up-to-date. Doing so in a "live" system is a challenge; the standard approach of simply building a new index and throwing out the old one may not work because the system must *always* provide an up-to-date view of the index, in contrast to web-based systems for which the use of an hour-old index during the merge process is acceptable.

We are also exploring approaches to partitioning the index. Traditional text-based inverted indices used by web search engines can be partitioned along two axes: documents and terms, with most search engines using both approaches to partitioning. This technique is successful because the partitions are one-dimensional, and because each term lists all of the documents associated with it. With multi-dimensional faceted search, however, this approach is much less successful because any partition along one facet is likely to distribute other facets broadly, requiring *all* partitions to be searched to find relevant documents using a distributed facet. An alternate approach, distributing each facet separately, would make the index very large, requiring each document to be listed many times, and would result in slow searches because the index would have to merge lists for

each facet. Since a single value in a facet could apply to many files, with only the *combination* being rare, this approach could be very slow. Because searches on subsets of the facet space and subsets of the "linked" file space will be common, we are considering approaches that partition the space based on both facets and location.

5. USER INTERFACE

With support for faceted metadata and indexing integrated directly into the file system, we propose to use a faceted search interface to allow users to interactively search and browse their shared petabyte-scale file system. We believe this is a novel approach to file system interfaces. Within the information retrieval community, faceted search has been primarily focused on static document collections, such as libraries and e-commerce catalogs. Dynamic collections, such as those found in shared storage, are not well studied. In the file system community, work on semantic file systems has primarily been focused on low-level issues such as load balancing and system APIs, not on the end-user application level. This proposed work intends to bridge this gap.

A recurring theme in our proposed system is personalization and user collaboration. In a shared petabyte-scale system, different groups of users will have differing needs. By customizing the search/browse interface to each user, the user can have a detailed view of the portion of the file system he/she is most concerned with, without becoming overwhelmed by extraneous portions of the file system.

Personalization is most effective when the system has large amounts of user feedback in order to learn a model for each user. This presents two problems. First, users must often endure a period of poor performance before the user sees any benefit from personalization. The second problem is given the size of the file collection, each user will only interact with a tiny fraction of the available files. We propose to solve these problems by using a combination of content-based and collaborative recommendations. Content-based recommendations measure the similarity of the internal structure, in this case the faceted metadata, of relevant files to make suggestions to a user. Collaborative recommendations on the other hand, make suggestions based on the similarity of the users/queries and which files were considered the most relevant to each query. This hybrid approach has found success in large scale e-commerce systems for document recommendation [21], and can easily be adapted to this domain.

Applying faceted search to file systems present some unique challenges from a IR perspective. IR research into faceted search has primarily been concerned with homogenous document collections. In file systems, homogeneity is not assured. Each type of file could potentially have widely different facets. We propose two different approaches to handle this. The first method is to use personalization and user collaboration to present the facets that are most useful to users.

Relevant files are determined by noting which files are accessed after each query. Since each user is likely to see only a small fraction of the files that are relevant to his/her query, the users must work together in order to improve the search experience for all. This is accomplished through a combination of content-based and collaborative recommendations. Once the set of most useful facets are determined, these facets are initially presented to the user during the search/browse interaction. The intuition behind this idea is that some facets contain information that is more recognizable to users, and therefore the browse/search interface should focus on those facets rather than overwhelming the users with more esoteric facets.

The second approach is to present a "zooming" interface. [13] In this approach, the browsing/querying interface presents the facets that are both prevalent the files returned by the current query, and have values suitable for query refinement. As the user navigates towards files of a specific type, the facets that are specific to those files become available for query refinement. Zooming interfaces are potentially useful since they present users with the major features of the current search space, while simultaneously avoiding overwhelming users with minutiae.

Depending on the specifics of the types of files stored in the repository, there may not be facets that are common to all files that are also useful for users. In this case, it may become necessary to group facets with similar semantics together. This is known as the "linkage problem," and it has been studied in both the IR and database research communities. [5].

One possible approach to solving this problem is to cluster the facets with common values together. For example, suppose a file repository that contains both design documents and C source code. The design documents and source code files feature the facets *author* and *developer* respectively. If the design documents and the source code are written by same persons, then these two facets can be combined to create the *author/developer* metafacet. Combining these facets not only reduces the amount of information to display in the user interface, but also groups files together in an intuitive way.

Another problem with current faceted search interfaces is choosing which values for a facet should be suggested to a user for query refinement. Current selection methods either naïvely suggest the pairs that most frequently occur in the current search results, or simply avoid the issue by presenting every possible value for a facet. We propose to solve this problem by personalizing the search interface with a technique similar to that used for selecting relevant facets.

Finally, collaborative and content-based recommendations can be used to help rank the files returned by a query in order of relevance. Proper ranking of results for queries for file system is an open problem in the IR community [11, 18, 17]. Current file systems maintain little information about the relationships among stored files.

6. CONCLUSION

This paper highlights some of the problems with traditional file access mechanisms when applied to petabyte-scale data stores. We presented faceted search as one possible approach to alleviating these problems, and gave an overview of some of the research into utilizing faceted metadata.

Currently, development is proceeding on two parallel tracks. On the file system front, a prototype of ViewFS and QUASAR is being developed. Research into the personalized user interface is focused suggesting most relevant facets and facet-values. Promising statistical models, along with a metrics for evaluating user interface effectiveness have been created.

7. ACKNOWLEDGMENTS

This work was supported in part by National Science Foundation IIS-0713111, the Department of Energy under award DE-FC02-06ER25768, by Lawrence Livermore National Laboratory, and by the industrial sponsors of the Storage Systems Research Center and Information Retrieval & Knowledge Management Lab at the University of California, Santa Cruz. We thank the members of the SSRC, IRKM and the anonymous reviewers for their feedback.

8. REFERENCES

[1] S. Ames. The viewfs interface and query language. *UCSC tech report in preparation*.

[2] S. Ames, N. Bobb, K. M. Greenan, O. S. Hofmann, M. W. Storer, C. Maltzahn, E. L. Miller, and S. A. Brandt. LiFS: An attribute-rich file system for storage class memories. In *Proceedings of Mass Storage Systems and Technologies*, May 2006.

[3] Apple Developer Connection. Working with Spotlight. http://developer.apple.com/macosx/tiger/spotlight.html, 2004.

[4] Beagle Project. About beagle. http://beagle-project.org/About.

[5] E. Elmacioglu, M.-Y. Kan, D. Lee, and Y. Zhang. Web based linkage. In *Proceedings of the Workshop on Web Information and Data Management (WIDM 2007)*.

[6] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[7] S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google file system. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, Bolton Landing, NY, Oct. 2003. ACM.

[8] D. Giampaolo. *Practical File System Design with the Be File System*. Morgan Kaufman Publishers Inc., San Francisco, CA, USA, 1998.

[9] D. K. Gifford, P. Jouvelot, M. A. Sheldon, and J. W. O. Jr. Semantic file systems. In *Proceedings of the 13th ACM Symposium on Operating Systems Principles (SOSP '91)*, pages 16–25. ACM Press, December 1992.

[10] Google. Google desktop - features. http://desktop.google.com/features, 2007.

[11] D. Hawking. Challenges in enterprise search. In *Proceedings of the Australasian Database Conference*, pages 15–26, January 2004.

[12] S. Henderson. Genre, task, topic and time: Facets of personal digital document management. In *Proceedings of the 6th ACM SIGCHI New Zealand Chapter's International Conference on Computer-Human Interaction (CHINZ '05)*, pages 75 – 82, New York, NY, USA, 2005. ACM Press.

[13] A. K. Karlson, G. Robertson, D. C. Robbins, M. Czerwinski, and G. Smith. Fathumb: A facet-based interface for mobile search. In *Proceedings of CHI '06, Human Factors in Computing Systems*, New York, NY, USA, 2006. ACM Press.

[14] C. Maltzahn, N. Bobb, M. W. Storer, D. Eads, S. A. Brandt, and E. L. Miller. Graffiti: A framework for testing collaborative distributed metadata. In *Proceedings in Informatics*, pages 97–111, 2007.

[15] M. A. Olson. The design and implementation of the inversion file system. In *Proceedings of the Winter 1993 USENIX Technical Conference*, pages 205–217, January 1993.

[16] Y. Padioleau and O. Ridoux. A logic file system. In *Proceedings of the 2003 USENIX Annual Technical Conference*, pages 99–112, June 2003.

[17] S. Shah, C. A. N. Soules, G. R. Ganger, and B. D. Nobel. Using provenance to aid in personal file search. In *Proceedings of USENIX Annual Technical Conference (USENIX 2007)*, June 2007.

[18] C. A. N. Soules and G. R. Ganger. Connections: using context to enhance file search. In *Proceedings of the 20th ACM Symposium on Operating Systems Principles (SOSP '05)*, pages 119–132, New York, NY, USA, 2005. ACM Press.

[19] E. Stoica, M. A. Hearst, and M. Richardson. Automating creation of hierarchical faceted metadata structures. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computation Linguistics (NAACL-HLT 2007)*. NAACL-HLT, 2007.

[20] D. Tunkelang. Dynamic category sets: An approach for faceted search. In *Faceted Search Workshop '06*, 2006.

[21] Y. Zhang and J. Koren. Efficient bayesian hierarchical user modeling for recommendation systems. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*, New York, NY, USA, July 2007. ACM Press.