Graffiti: A Framework for Testing Collaborative Distributed Metadata

Nikhil Bobb Damian Eads Mark W. Storer Scott A. Brandt Carlos Maltzahn
Ethan L. Miller

1. Introduction

The growth in metadata has been triggered by two key catalyzing events. The first is the explosive growth in storage size and storage demands. As the number and variety of files grows the need for metadata to organize this information windfall becomes more critical. An example of this need for metadata can be seen in music collections where filenames and folders alone are not sufficient to organize a sizable collection. Instead most users rely on external programs to dynamically organize their collection based on rich metadata. This need for organizing has influenced the second key factor which is the additional metadata capabilities that have been built into modern operating systems and application level programs. The problem of organizing large amounts of information has thus lead to the exploration of new metadata structures and metadata operations.

While this rich metadata helps users wade through large volumes of information, there is also the problem that metadata is often confined within the scope of a single program or single system. The metadata that shows up alongside a particular photo in one photo library program may not appear in another. Additionally, the metadata capabilities on one system may not exist on another. In this case moving files from one system to another has the effect of wiping clean much of the file's metadata and leaving only the raw data behind. This has the unintended consequence of deterring users from utilizing the full capabilities of a rich metadata system if that user fears being locked into a single program or platform.

We propose a distributed metadata layer which exists above the file system. The purpose of this layer is to manage the metadata for a collection of systems and make this metadata available to applications through well-published interfaces. Separating the metadata management from the application has the dual benefit of freeing application designers from the problem of metadata storage as well as making metadata available across applications and systems. This distributed design also encourages the development of a new class of applications that harness the power of a shared metadata library. Such a library could assist in searching for data within a collection of systems as well as encourage collaborative organization of data.

2. Distributed Metadata

In its current design, Graffiti provides three distributed metadata capabilities: tags, links, and indices. Tags are short text labels associated with a file. Two files that share a tag in common suggest a possible relationship, thus a group of files that share a tag in common have an implicit loose grouping. In contrast, links are an explicit relationship between two files. Unlike the relationship imposed by tags, links have an explicit source and destination. Indices can be shared on features such as text content, image features, and video content. Once indices are shared, they can enable fast distributed search and location of files.

As the number of files increases filenames begin to lose some of their meaning as their purpose is unclear outside of a context within the file system. An example of this would be a Makefile. On a single system there may be many files with the name "makefile" and their name alone is not sufficient to positively identify the file. This scenario is exacerbated in a large distributed context where the number of files can be extremely large. To remedy this, Graffiti references files in a non-local context by their SHA-256 checksum, thus making sure that all identical copies of a file can be positively identified. This use of checksums as a global file identification allows the system to share metadata on matching files across the entire distributed system. In the local context, the user or application will have access to a set of tags and links which they have set. In a distributed context, the system will leverage all available tags, links and indices to allow better search and organization of files.

3. Preliminary Design

Our distributed metadata file system is an overlay network consisting of client systems and federated servers. The client layer allows the client to take advantage of certain metadata operations even if they are disconnected from

This paper appeared at the 7th Workshop on Distributed Data and Structures (WDAS 2006), Santa Clara, CA, January 2006.

the network while federated servers allow ubiquitous metadata access as well enabling richer metadata structures.

3.1. Architecture

The overlay architecture provides two key features. The first is that it is possible to add the distributed metadata features to any existing file system without effecting the underlying structure of the file system. In this manner it is possible to quickly deploy the capabilities of the system with a minimum of effort and the lowest possible risk to a system's data contents. The second benefit is that it provides a useful test bed for new metadata structures such as tags and links. Testing new metadata structures and operations in the file system can be quite time consuming in terms of engineering and testing resources. By moving the features to the application layer it is possible to quickly develop and test new operations on existing data with little to no impact on the underlying system. This test bed is especially helpful for discovering new uses of novel metadata primitives and operations not imagined by their designer. By providing a platform to produce tools and application that utilizes these new metadata structures, valuable usage model data as well as new uses can be obtained.

The architecture for the client consists of a application layer overlay and integrated relational database. This allows the client to quickly access local metadata as well as provides the client access to metadata when it is disconnected from the network. Files on the client are identified by a SHA-256 checksum. The database is accessible by a metadata engineer so that tables can be added as required. This is considered important since new tables may be needed in order to implement new metadata structures or to aid in gathering usage model and workload data.

Above the clients there is an array of federated servers. Changes at the client level can be synchronized with the server to allow system wide searches of metadata. These system wide searches are also enabled by inter-server communication. As with the client the server database is also accessible for modification for the purposes of adding functionality or data collection capabilities.

One of the key purposes of the current design is for testing and evaluation of metadata structures and operations. Thus an important aspect of the system is logging. While the ordering of logged events can be totally ordered within the context of a single system the use of time-stamps with a synchronized clock will allow, in the very least, a partial ordering of events on a system level view. This may be expanded to include communication protocols to allow a more complete ordering of system-wide events.

3.2. Security

We have come up with a preliminary security model which reflects some of the functionality we would like to add to the system. The system has two security categories: metadata and ownership. Each of these two categories can be set to either private or public. The metadata category reflects the access to your metadata which you grant to the public. The ownership category allows you to publicly acknowledge or deny that you own a file.

As an example, public metadata would be used to share the genre of an MP3 file. Private metadata could also be used to hide that I have tagged the PDF agenda of a meeting "boring". Public ownership might apply when entertaining requests for GPL binary libraries, which others might want to use. Private ownership would be used to hide ownership of the latest bootlegged software.

4. Usage Scenarios

New metadata structures and operations are often created without a specific application in mind. Additionally, these structures and operations are often found to be applicable outside of the scope they were created in. Thus an important part of metadata research is exploring what is capable and what can be improved by applying novel metadata ideas to new scenarios. As part of this active research area we have identified a few of the current experimental areas that we have begun to explore using our distributed metadata file system.

4.1. Collaborative Descriptions and Filtering

Tags have been popularized by online services such as Flickr and del.icio.us [3, 2, 6]. They offer a way for the user to easily add descriptive metadata to a file which in turn allows effective searching of files. On a system such as Graffiti once a critical mass of user's is reached who have files in common, the system can begin to suggest relevant tags. Take for example Alice and Bob who both have the same file music.mp3 on their independent computers. If Alice has chosen to share her metadata, and has tagged music.mp3 with the tag "classical", then when Bob adds the file to Graffiti, the system can suggest the tag "classical".

The collaborative filtering community (for two recent surveys see [4, 7]) has developed techniques to map ownership information to recommendations by clustering groups of users. Such techniques combined with the metadata in Graffiti could be used to suggest music, video, or academic papers to a user.

4.2. Distributed Indexing and Data Management

Highly distributed, large collections of unsynchronized data are the norm in today's multinational organizations. Due to resource constraints in many such organizations centralized access and storage of the whole corpus of data is infeasible. Graffiti could alleviate this situation by facilitating aggregation of local indices by a series of federated servers. Distributing indices is a much lighter weight commitment than transferring all available data, and would allow system wide data and document search. Taking the idea even further, agents using Graffiti as a directory could be responsible for retrieving requested documents from remote locations, as well as caching and mirroring such documents based on their popularity.

4.3. Package Management

Links allow the system to explicitly define a relationship between two files. One possible application that could benefit from explicit relationship is package management. The problem faced by package management schemes is that files are often dependent upon other files and thus copying data between system is not as straight-forward as it would appear. Additionally, current package management schemes often require the users to commit to a particular package management tool in order to maintain consistency within the system or require a dedicated central server to house a repository [11]. Links may allow a new type of package management that would be lighter in weight and useful for quickly packaging and distributing files such as LaTeXdocuments.

One simple use of links within the file system is to establish dependency relationships between files. For example, a program source file might link to a makefile, libraries, headers and other related files. Since links carry an explicit direction there might also be a link from the makefile to all the files it references. In this manner a dependency search starting from any given node could quickly produce a graph describing the related files. This example of a program source file linking to a makefile and a makefile linking to a source file brings up an important question about the semantics of links within a file system. What direction should a link point? More abstractly, what are the semantics of a link in a file system. Does it make more sense for a makefile to point to source files or a for source files to point to a makefile. This problem is made more confusing when the transitive closures between linked files is examined or what it means to encounter a cycle within the graph of links. A metadata primitive that could help define the semantics of the links are attributes that can be attached to links. These attributes are key-value pairs that describe a link. A link with no attributes states:4 "There is a relationship between these files" while a link with one more attributes might be able to state, "File A can be generated from file B". In this manner a link from a makefile to a source file and a link from the source file to a makefile can be distinguished as representing distinct relationships.

As with other possible applications there are many open questions with package management using attributed links. The overlay network design of the system as described in section 3.1 allows us to easily explore many of these questions.

For example, there may exist conditional links that are only required under certain conditions. This leads to the question of the best way to encode this conditional behavior. Additionally the distributed nature of the system may allow these conditional dependencies to be obtained only when needed and thus the system may want to support links to files that it does not currently have. In other words, the system knows that the file exists, when it would need them, and how to obtain it even though it does not presently need it. This conditional linking may be further optimized by recommended links. For example, a link might suggest that while the file linked to is not required it is strongly recommended. This would be useful when recommending dependencies such as a codec for a video file.

4.4. Scientific Data Management

Scientific data management needs new approaches for organizing, sharing, and using masses of scientific data among a large community of users, especially in a decentralized and distributed context. Our approach of combining tags, links, and indices is well-suited to meet these needs. First, organization is greatly enhanced by the semantically-rich data layouts facilitated by metadata. These layouts enable fast, context-dependent query and retrieval of scientific data. Second, the distributed nature of Graffiti encourages scientific collaboration and sharing– two essential components of scientific inquiry. Third, Graffiti greatly simplifies the scientific programmer's task of managing data; scientists can focus their efforts on using data as opposed to getting to it.

The field of scientific computing will demand well-organized distributed storage systems. Geophysical sensors collect data at an overwhelming rate. For example, LANDSAT-7 collects approximately 150 GB of data per day[10]. IKONOS collects over 1100 images per day [8]. EOSDIS has a storage archive that exceeds one petabyte [12]. The Rosetta Genomics Group projects that their storage needs for their Human Genome Database will grow from 10 TB to 100 TB [9]. As the mass-storage needs of science grow, data organization becomes a much bigger challenge.

Named relational links enables flexible organization of

files leading to easier navigating, more intelligent querying, and adaptive clustering. Suppose a university is conducting a multi-year study of farm health to support more accurate modeling of crop yields. This study might require the use of data collected from multiple kinds of sensors. Infrared sensors, which measure heat emitted from objects being sensed, are good indicators of crop health. Multispectral sensors can provide a rich spectral profile to help distinguish between heat emitters, say roof tops and crops. Panchromatic sensors provide very fine-grained spatial resolution which can greatly aid visual analysis of data. Also associated with the scene might be topographical surveys or land-cover classification maps. These maps may be vectorized or rasterized, and depending on their format, and may require different format reading routines to process them as input into a scientific application. Sensors exhibit different properties: they have different band centers, widths, calibration parameters, artifacts. A particular sensor, such as a lens, may always produce an artifact such as a blur at a particular spot, and thus a sensor-specific preprocessing routine may be needed to eliminate such artifacts. A sensor might also annotate a scene with a tag to indicate the cloud-cover or time of day the image was taken. Ground-based lightning triangulators can be used to measure storm history. Temperature time series provides a history of the temperature conditions of the scene. Sensors may take multiple measurements over time, which is useful for temporal-oriented modeling. A scene might be adjacent to other scenes under study. Thus, we see that a single composite observation can be quite complicated. Distributed metadata can be used to address several key issues:

- **modalities:** Data is captured in different modalities, e.g. images, vectorized maps, time series, and spectrograms. These modalities can be easily related using links, and searched using tags. Additionally, a "format" link can be invoked to appropriately locate a format reading routine.
- **sensor differences:** Even very similar sensors may differ in minor ways, and thus, may require different processing algorithms. An index on a sensor feature might help locate an artifact removal algorithm.
- **time:** Data from the same sensor might be collected at different times. The data files at different can be linked together in sequence using links.
- **processing algorithms:** Different combinations of sensors may require different models. These models can be found by invoking a link on a group of sensors.
- **indexing related data:** Links enable related data to be quickly retrieved. For example, the hash of a satellite image might link to all the maps that overlap.

Graffiti metadata provides a means for organizing complicated sets of distributed scientific data files through descriptive linking, tagging, and indexing.

Scientific applications often need to manipulate large quantities of data scattered about the file system. Traditionally, writing programs which make use of many files is cumbersome. The programmer must manage a list of files in some way, and generate valid path names to get to these files. Path names often change as files are moved over the network or about the file system. Graffiti offers a greatly simplified interface for the programmer; management of related files is embedded into the system. This will reduce the complexity of writing scientific applications, users can invest more of their time on how to use data, and focus less on where to find them.

Graffiti is navigated via named links between files. A semantically rich description of the relation between files on the system will influence application design. Rather than sifting through flat directories looking for related files, applications can directly load related files by invoking a name. As we saw in our crop health study example, to find an appropriate artifact removal algorithm, the program simply invokes a "artifact removal algorithm" on a data file its already loaded to immediately retrieve and execute the appropriate algorithm.

5. Implementation

We have implemented a first prototype of the proposed system as a Java client and a single mySQL server. Client/server communication uses Java's RMI which is insecure but sufficient for the purposes of the initial prototype. For future prototypes we will use secure HTTP.

The prototype client allows users to search for files by conjunctions of tags and to edit tags of one or more files. The prototype uses Apache Derby [1] as an embedded relational database written entirely in Java. Since the database runs within the same Java Virtual Machine as the application the installation of the client is worry free on Linux, Windows, Mac OS, and other common platforms that support J2SE (Java 2.0 Standard Edition).

Users interact with the client either via a graphical user interface or a command line interface. The command line interface allows the use of Graffiti functionality in scripts that implement, say, administrative functions such as regular backups or automatic tagging based on file content.

6. Related Work

We are currently not aware of work that investigates collaborative management of file metadata that works across file systems and is independent of any particular file sys-

tem. We already mentioned examples of collaborative tagging of Web content such as del.icio.us. Web sites such as Amazon.com provide good examples for collaborative filtering in form of product recommendations based on a combination of individual and collective user behavior.

The Presto document management system extends traditional file systems with arbitrary attributes [5] that allow files to be grouped and searched by these attributes. The system presents itself as a file system and can mount other file systems via NFS and extending them with Presto functionality. Thus Presto's approach to providing metadata across multiple file systems is accomplished by a layered architecture that duplicates and mimics traditional file system functionality in addition to extended Presto functionality. Graffiti on the other hand maps directly to file content independent from any particular file system structure and strictly compliments traditional file system functionality. In addition we focus on collaborative management of metadata across many users.

7. Conclusions and Outlook

Graffiti currently is designed to share three kinds of metadata: links, tags and indices. We propose a metadata sharing framework that enables applications such as package management, distributed data management, distributed indexing, and search.

There are many open questions on distributed metadata which we are currently investigating. Amongst them are the privacy implications of shared metadata, efficient clustering methods to leverage shared metadata, and local event management leading to database updates.

One of the largest privacy concerns we have is how can we store and backup private metadata in a way that cannot be subpoenaed and not be traced back to it's original owner without compromising it's utility. We are in the process of evaluating peer-to-peer network architectures that are trying to address this issue.

References

[1] The apache derby project. http://db.apache.org/derby/.

[2] del.icio.us. http://del.icio.us, Nov 2005.

[3] Flickr - photo sharing. http://www.flickr.com, Nov 2005.

[4] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 43–52. Morgan Kaufman, 1998.

[5] Paul Dourish, W. Keith Edwards, Anthony LaMarca, and Michael Salisbury. Presto: an experimental architecture for fluid interactive document spaces. *ACM Trans. Comput.-Hum. Interact.*, 6(2):133–161, 1999.

[6] Scott A. Golder and Bernardo A. Huberman. The structure of collaborative tagging systems. Technical report, Information Dynamic Lab, HP Labs, 2005.

[7] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237, New York, NY, USA, 1999. ACM Press.

[8] Space Imaging. Space imaging releases top 10 IKONOS satellite images for 2003. http://www.spaceimaging.com/newsroom/2004_top10.htm, 2003.

[9] Microsoft. Rosetta genomics 10 tb human genome database., 2003.

[10] NASA. FGDC annual report to OMB. http://www.allaboutmems.com/memsapplications.html, 2003.

[11] John P. Rouillard and Richard B. Martin. Depot-lite: A mechanism for managing software. In *8th USENIX System Administration Conference (LISA VIII) Proceedings*. University of Massachusetts at Boston, USENIX, Sep 1994.

[12] UCAR. AIST NRA-02 presentation to SSO, 2002. Available at http://www.esmf.ucar.edu/esmf_presentations/pres_0507_lipingdi.pp