Lifting File Systems into the Linked Data Cloud with TripFS

Bernhard Schandl
bernhard.schandl@univie.ac.at

Niko Popitsch
niko.popitsch@univie.ac.at

University of Vienna, Department of Distributed and Multimedia Systems
Liebiggasse 4/3-4, 1010 Vienna, Austria

ABSTRACT

A major fraction of digital information is stored in file systems. File systems organize files usually in labelled directory trees and provide a minimum support for user-driven file annotation, linkage and categorization. Although file systems play a major role in knowledge organization, both in enterprise contexts as well as in the personal information sphere, they have rarely been considered in Web-based information integration. To a large extent, this can be contributed to the limited metadata support of file systems and to the lack of stable identifiers for file and directories, which makes it hard to expose these objects in a global Web. We present TripFS, a lightweight approach for exposing parts of local filesystems as Linked Data. Serving file system objects via dereferenceable HTTP URIs paves the way to integrate them with the Web of Data, and enables new possibilities of exploiting file system data, for example, by linking them with other data sources or by annotating them using Semantic Web technologies.

Categories and Subject Descriptors

D.4.3 [**Operating Systems**]: File Systems Management; H.3.5 [**Information Storage and Retrieval**]: Online Information Services

General Terms

Algorithms, Design

Keywords

Linked Data, file systems, file metadata, information representation, information integration, event detection

1. INTRODUCTION

File systems store and organize data and documents of all sorts and of arbitrary complexity, ranging from small information snippets that can be put into single files, to large repositories of heterogeneous content that are organized within deep hierarchical structures. They act as the storage backbone of many information processing systems and can be considered as one major fundament of personal and corporate information management. Since common file systems do not impose major restrictions on creating, naming, and arranging directories and files, they support a user's individual preferences for data organization. File systems do not only store files that were created or modified locally: a large share of files originates from other sources, like multimedia devices, other desktops, or the Web. In corporate environments it is common to store data of collective interest on shared file servers that enable a simple form of collaboration.

Overall, file systems can be considered as one of the primary information sources both for organizations and individuals, and it is quite likely that they will remain to be important in the future. Therefore they are of high interest for information integration. However, file systems have only rarely been considered in the field of Web-based data integration. This stems mostly from their limited possibilities of data organization[1], limited metadata support, and the lack of stable identifiers for files and directories.

One promising strategy for Web-based information integration is the *Linked Data* paradigm. This term denotes a set of technologies and best practices that facilitate information integration and linkage on a global scale. To expose information as Linked Data means to follow simple principles [6]: first, identify each resource of interest with a globally unique, dereferenceable HTTP URI; second, provide useful information for clients when they access the URI (usually expressed in RDF and HTML); and third, include links to other resources so that clients can retrieve more potentially interesting information.

In this paper, we present *TripFS*, a lightweight approach that applies these principles for file systems in order to expose their contents as Linked Data, and therefore enables their direct inclusion in Web-based integration scenarios. It assigns stable, globally valid, dereferenceable URIs to files and directories, monitors changes in the system, serves metadata extracted from files as RDF data, and interlinks files with external data sources. It provides a plug-in architecture so that it can easily be extended to support additional file types and linking components, it adapts to the specifics of the underlying file system, and it provides a sophisticated file change tracking component that increases the stability of file identifiers.

Because it is easy to set-up, TripFS also facilitates ad-hoc sharing of file-based resources using standardized (semantic)

[1]By now it seems commonly accepted that a single hierarchical scheme is insufficient for the organization of large amounts of data as we encounter them on today's desktop environments.

Web technologies. Moreover, it overcomes shortcomings of hierarchical organization mechanisms, because its metadata-centric approach allows to query for descriptive information instead of file location, and to establish multiple, orthogonal views on file system data.

After outlining application scenarios and describing how users can benefit from exposing file systems as Linked Data (Section 2), we discuss which steps have to be taken in order to realize this idea (Section 3). We present details about the TripFS architecture and implementation (Section 4). After a discussion of related work (Section 5) we conclude the paper in Section 6.

2. BENEFITS OF LINKED FILE SYSTEMS

The benefits of exposing data as Linked Data resources are manifold [9]. In this section we outline three scenarios that illustrate how the quality of file system usage can be increased by exposing files as Linked Data.

A) Integrating File Systems into Enterprise Data. A substantial fraction of enterprise data is available in the form of file systems. While these data can be accessed in a distributed context using protocols like CIFS or WebDAV, it is difficult to integrate them in a global enterprise context due to the lack of stable identifiers for files and platform-independent metadata-based file access mechanisms. Linked Data has been shown to be a viable approach for lightweight enterprise information integration [16]; therefore, making file systems part of a global or enterprise-internal Web of Data enables them to be seamlessly integrated with, and semantically connected to other data sources.

B) Web-based Ad-hoc Data Sharing. Despite the vast amount of possibilities for digital communication we have at our disposal, ad-hoc sharing of meaningful information (e.g., the exchange of digital documents between participants' laptops during face-to-face meetings) is still cumbersome. We can regularly observe that collaborators use e-mail or instant messaging to quickly exchange files. This approach, however, does not allow more complex data to be shared, or to exchange files together with metadata that describe their correct context. Linked Data builds on top of common Web technologies, thus any Linked Data source can be directly accessed using a common Web browser. A tool that allows users to temporarily share selected parts of their local file systems as Linked Data (which implies not only sharing plain files, but also extracted metadata, annotations, and links) facilitates efficient information exchange amongst collaborators.

C) Semantic Web-based File Annotations. Semantic annotation and interlinking of files is badly supported today: although modern file systems support the storage, management, and retrieval of file annotations (e.g., extended attributes or file forks), these data are not accessible in a standardized and platform-independent way. This makes the organization of files into logically connected units difficult, and reduces the efficiency of file retrieval especially in distributed environments. If file systems were published as part of a Web of Data, they could be annotated and interlinked using tools like the LEMO annotation framework [13] or the Silk framework [22], which would lead to an increased quality of search and retrieval, as well as linkage with other relevant data sources. In turn, these Linked Data and Web-based annotations could be propagated back into the working context of the file system user, e.g., by being considered by desktop search engines.

3. REPRESENTING FILE SYSTEMS AS LINKED DATA

Since the characteristics of file systems and Linked Data differ significantly, a number of steps have to be performed in order to lift file system data into a Web of Data:

1. Appropriate representations for files and directories have to be found, which comply to the Linked Data principles.

2. Vocabularies that convey the characteristics of data found in file systems have be to be specified and aligned to already existing relevant vocabularies.

3. Descriptive metadata about files have to be extracted from the file system and transformed into the RDF data model.

4. Meaningful links to other, external data sources have to be detected and established.

5. Consistency between the file system and its corresponding Linked Data representation has to be ensured.

6. Data have to be served according to Linked Data principles, i.e., in a form that is usable for both, humans and machines.

In the following we outline how each of these steps can be realized.

3.1 File URIs in the Web of Data?

Within the context of a file system, files and directories can be uniquely identified using their absolute paths, each of which consists of a sequence of directory names and a file name. The `file:` URI scheme [5] is a means to directly reuse these paths to form URIs, which can in turn be used to access local file resources in a computer system.

However, file URIs are neither *globally unique*, since they describe a local path to a resource on a particular host, nor *stable*, since the referenced files and directories may be removed, moved, or renamed. Therefore they are not suitable for being used in a global Web of Data.

To solve this identifier problem, we chose to use opaque, randomly generated UUIDs, and assign them to files and directories. The usage of random UUIDs in a global distributed context is assumed to be safe since the probability of a collision is sufficiently low. Further, since UUIDs are fully opaque, they do not convey information about the physical location of files and directories, and are therefore stable even when the underlying file system objects are changed. However, this requires to maintain a mapping between stable, UUID-based URIs on the one hand, and unstable, path-based identifiers on the other hand, to ensure that modifications in the file system are properly reflected in the Linked Data representation. In Section 3.6 we outline our strategy to accomplish this.

```
<urn:uuid:887d728e-bc12-4f28-a497-7d66439086e9>
   a tripfs:File ;
   rdfs:label "eswc2009-schandl.pdf" ;
   tripfs:local-name "eswc2009-schandl.pdf"^^xsd:string ;
   tripfs:path "/Users/bs/Data/work/papers/2009/eswc/eswc2009-schandl.pdf"^^xsd:string ;
   tripfs:size "425561"^^xsd:long ;
   tripfs:modified "2009-03-11T02:38:45"^^xsd:dateTime ;
   tripfs:parent <urn:uuid:35069c61-451e-4688-98f5-080924b261f4> .
```

Figure 1: A Linked Data representation of a PDF file

```
<urn:uuid:887d728e-bc12-4f28-a497-7d66439086e9>
   a tripfs:File, foaf:Document, nfo:FileDataObject ;
   tripfs:parent <urn:uuid:35069c61-451e-4688-98f5-080924b261f4> ;
   nfo:belongsToContainer <urn:uuid:35069c61-451e-4688-98f5-080924b261f4> .

<urn:uuid:35069c61-451e-4688-98f5-080924b261f4>
   a tripfs:Directory, dctype:Collection, nfo:FileDataObject, nfo:Folder ;
   tripfs:child <urn:uuid:887d728e-bc12-4f28-a497-7d66439086e9> ;
   nie:hasPart <urn:uuid:887d728e-bc12-4f28-a497-7d66439086e9> .
```

Figure 2: Interoperability through the usage of multiple overlapping vocabularies

3.2 Files and Directories as Web Resources

The parent-child relationships between files and directories can be represented as RDF triples with appropriate predicates. Several triples are added to each file or directory resource that convey data that are directly retrieved from the file system: the local name (i.e., the actual file or directory name without the entire path information), the file size, and the dates of creation and last modification. An example of a file's RDF representation is depicted in Figure 1. Resources that represent files or directories are internally identified by UUID-based URNs; for serving them as Linked Data they are dynamically rewritten to HTTP URIs (cf. Section 3.7).

3.3 Vocabularies

In order to describe files, directories, their metadata and their relations as RDF, we have developed a simple OWL vocabulary published at `http://purl.org/tripfs/2010/02#`. We have derived our vocabulary from existing semantic vocabularies as much as possible. However, as it is currently uncommon to expose file resources as Linked Data, we observed a lack of community-accepted vocabularies for this purpose. To the best of our knowledge, only the NEPOMUK File Ontology[2] (NFO) has been specifically defined to model the contents of file systems. It provides terms to describe files, directories, and their properties. Our vocabulary is aligned with NFO and provides more specialized terms, according to our system's requirements.

A number of other vocabularies, however, have a general notion of the concept of *documents*, and usually align this concept to the `foaf:Document` class. On the other hand, several vocabularies have a notion of *collections*, which can be compared to directories in a file system; for instance, OAI-ORE [17] or Dublin Core[3]. The Dublin Core Type Vocabulary[4], as another example, defines terms for different resource types as well as collections. Additionally, there exists a large number of vocabularies that can be used to identify media types and their specifics; e.g., the MPEG-7 ontology[5], the Music Ontology[6], or the set of NEPOMUK ontologies.

To reach a maximum level of interoperability, a data source should aim to adhere to commonly accepted vocabularies as much as possible. The RDF semantics allows to arbitrarily mix different, unrelated vocabularies; therefore we propose—in addition to using a custom vocabulary—to model file system data using the NFO vocabulary, and to add type information from popular vocabularies like Dublin Core and FOAF as they fit. By serving data using multiple, even already aligned vocabularies, we disburden data consumers from the need to perform additional inference. An example of such a mixed representation is presented in Figure 2.

3.4 Extracting Semantic File Metadata

Current file systems provide only a limited set of low-level metadata attributes associated with files such as name, owner, size, creation and modification date, or permission attributes. Modern file systems provide additional means to store higher-level metadata, like extended attributes or multiple data streams; however these are only useful if they are actually populated by applications, which is rarely the case.

[2]`http://www.semanticdesktop.org/ontologies/nfo/`

[3]`http://dublincore.org/groups/collections/collection-application-profile/`

[4]`http://dublincore.org/documents/dcmi-type-vocabulary/`

[5]`http://metadata.net/mpeg7`

[6]`http://musicontology.com`

```
1  <urn:uuid:887d728e-bc12-4f28-a497-7d66439086e9>
2     nie:mimeType "application/pdf" ;
3     nie:title "The Sile Model --- A Semantic File System Infrastructure for the Desktop" ;
4     nfo:pageCount 15 .
5
6  <urn:uuid:a998272d-45f0-4814-8f15-be5db5fe811a>
7     nie:mimeType "audio/mpeg" ;
8     nid3:title "Bohemian Rhapsody" ;
9     nid3:leadArtist [ nco:fullname "Queen" ] ;
10    nid3:length 355106 .
```

Figure 3: Metadata extracted from a PDF and an MP3 file

```
<urn:uuid:887d728e-bc12-4f28-a497-7d66439086e9>
   owl:sameAs <http://dblp.l3s.de/d2r/resource/publications/conf/esws/SchandlH09> .

<urn:uuid:a998272d-45f0-4814-8f15-be5db5fe811a>
   rdfs:seeAlso <http://musicbrainz.org/track/c7faf83f-9cb3-4de4-a39f-1c1f98b8d81a> ,
      <http://musicbrainz.org/track/95ebc842-9926-4658-8012-12c358247946> ;
   owl:sameAs <http://musicbrainz.org/track/bbd5a2e7-9814-4988-8f5a-dc38c208eeea> ,
      <http://musicbrainz.org/track/064c440c-4eba-47a6-83c4-c91a979eeb4b> .
```

Figure 4: External links to DBLP and MusicBrainz

As it is one of the Linked Data principles to "*provide useful information*" about a resource when a client dereferences its URI, it is desirable to extract additional, descriptive metadata from files and directories and expose them also as Linked Data. Reconsider, for example, Scenario A described in Section 2, where the value of file-system level metadata (like file size, file type, or file permissions) is limited; higher-level *descriptive metadata* that can be used for selective retrieval of files respectively their descriptions, e.g., via SPARQL, is required. However, the combination of these metadata enables sophisticated discovery, retrieval and access methods based on (i) the parent/child relations of file system objects, (ii) low-level file system metadata, and (iii) high-level content-based metadata.

The problem of extracting metadata from file systems has been studied for a long time. The biggest challenge in this field is the *data diversity* found in file systems, which is imposed by the multitude of different file types. To illustrate this, currently more than 51,000 file types are registered at the popular FILExt service[7]. Different file types exhibit different internal structures, and consequently different metadata can be extracted. It is therefore impractical to provide metadata extractors for this large amount of different file types within a single software component. It is instead more feasible to define a generic metadata extraction framework that allows specific extraction components for different file types to be plugged-in. By this, the system can be tailored to the respective application context.

In our approach, *extractors* read files and extract an RDF graph that contains triples representing the extracted metadata. Multiple extractors can be cascaded into an *extractor pipeline* and are sequentially applied to each object. The resulting RDF graphs are stored in the triple store and are then served as part of the file's and directory's description via the Linked Data interface. Extractors may extract not only file metadata (i.e., data about the *documents* represented by files), but also *entities* that are related to files (e.g., the artist who has performed the music stored in a MP3 file) and can in turn be linked to external data sources.

As an example, Figure 3 shows the RDF representation of metadata that have been extracted from two files; the first resource represents a PDF document containing a scientific publication, the second represents an MP3 audio file[8]. The blank node used to identify the artist in this example (line 9) needs to be dynamically rewritten to a stable, dereferenceable URI by the Web server (see Section 3.7).

3.5 Linking Files to External Sources

Once files and directories are represented as RDF resources it is possible to link them to other related resources on the Web. Doing so allows clients to retrieve more, potentially interesting information about the resource. For instance, files may be classified according to a classification scheme that uses dereferenceable URIs as identifiers; in this case, clients are enabled to query for files using these terms.

The task of linking files and directories to external resources can be accomplished by tools that provide this functionality for generic Web resources, which usually apply various heuristics to detect semantically related resources (e.g., shared identifiers or object similarity [22]). These heuristics depend on the information that is available for a particular entity; therefore in the context of a file system they depend on the data provided by metadata extraction components, as described in the previous section.

[7]`http://filext.com`

[8]In this example we have used terms from the OSCAF/NEPOMUK ontologies (`http://www.semanticdesktop.org/ontologies`).

Event	Reaction
Creation	Mint URI, add resource to RDF graph, perform extraction and linking
Deletion	Delete respective resource and associated metadata from RDF graph
Move/Rename	Update local path properties in the RDF graph
Update	Re-extract features, re-link, update RDF graph

Table 1: Reactions on file system events detected by the *watcher* component

As a consequence, we follow the same strategy as for metadata extractors and do not provide an all-in-one solution to the problem of linking files to external resources, but instead provide a framework that allows specialized linking components to be plugged in. These linking components can access not only the raw file data, but also extracted metadata, and use this information as basis for interlinking. Like extractors, *linking components* return RDF triples which are added to the metadata model and served via the Linked Data interface.

As an example, Figure 4 shows to which external sources a scientific publication and a music file can be linked, based on string similarity between the publication title and the combination of track title and artist name, respectively. In this example the PDF document from Figure 3 has been linked to the Linked Data variant of the popular DBLP publication database, and the MP3 file has been linked to resources of the MusicBrainz service.

3.6 Maintaining Consistency

As described in Section 3.1, it is required to mint a UUID-based URI for each file and directory, which can be considered globally unique from a practical point of view. However, without further precautions such URIs might be quite unstable as the mapping between an UUID-based external URI and a file-based internal URI is invalidated whenever a referenced file is moved, removed, or renamed. Further, updating such files may result in inconsistencies between a file and the metadata that has been previously extracted and stored. Note that this could lead also to invalid links between resources if these were automatically created based on file metadata, as described in Section 3.5.

In order to preserve a stable mapping between these URIs and the local files and directories they represent, we have to employ a *watcher* component that is responsible for detecting file system events that may result in different file URIs or modified file contents of referenced files. Whenever such an event is detected, appropriate actions have to be taken, and the RDF model has to be updated. Note that in this sense, the mapping between stable UUID-based URIs and instable file and directory paths acts as a kind of translation service between external, globally valid UUID-based URIs and corresponding local file URIs, comparable to PURL or DOI services [2]. Table 1 summarizes the reactions that have to be taken after file system events have been detected.

3.7 Serving File Systems as Web Resources

Once the RDF-based representation of files and directories has been generated and enriched with extracted metadata and links to external data sources, the resulting RDF graph can be served according to Linked Data principles. For this purpose, internal UUID-based URNs are dynamically rewritten to HTTP-based URIs with a configurable host part; e.g., `http://example.com:8080/resource/<uuid>`. It is considered good practice [7] to serve at least two variants of the data, an RDF representation for machines and an HTML representation for human consumption, and to let clients choose which representation they prefer using HTTP content negotiation. In addition to serving resources according to Linked Data principles, it is recommended to provide a SPARQL endpoint [10] to allow clients to search for resources based on their RDF descriptions. Furthermore, the actual file data itself can be downloaded to the client. In the special case where the Linked Data resources are retrieved locally (i.e., server and client are executed on the same machine), the Web server can add links to the HTML interface that allow the user to directly open directories or launch files from the browser, thus providing a seamless interaction experience. Figure 5 shows a screenshot of such an HTML-based interface, which provides these options to the user.

4. IMPLEMENTATION

TripFS has been designed as a modular service framework, which defines plug-in interfaces that can be used to extend and adapt the system to the actual needs of the use case, the file types to be served, and the special characteristics of the underlying operating system. Such interfaces exist for RDF *storage components*, file metadata *extractors*, file *linkers*, and file system *crawlers* (responsible for crawling a configured subtree of the file system) and *watchers* (responsible for maintaining the consistency of the mapping between external UUID-based URIs and internal file-based URIs). The system's architecture is depicted in Figure 6.

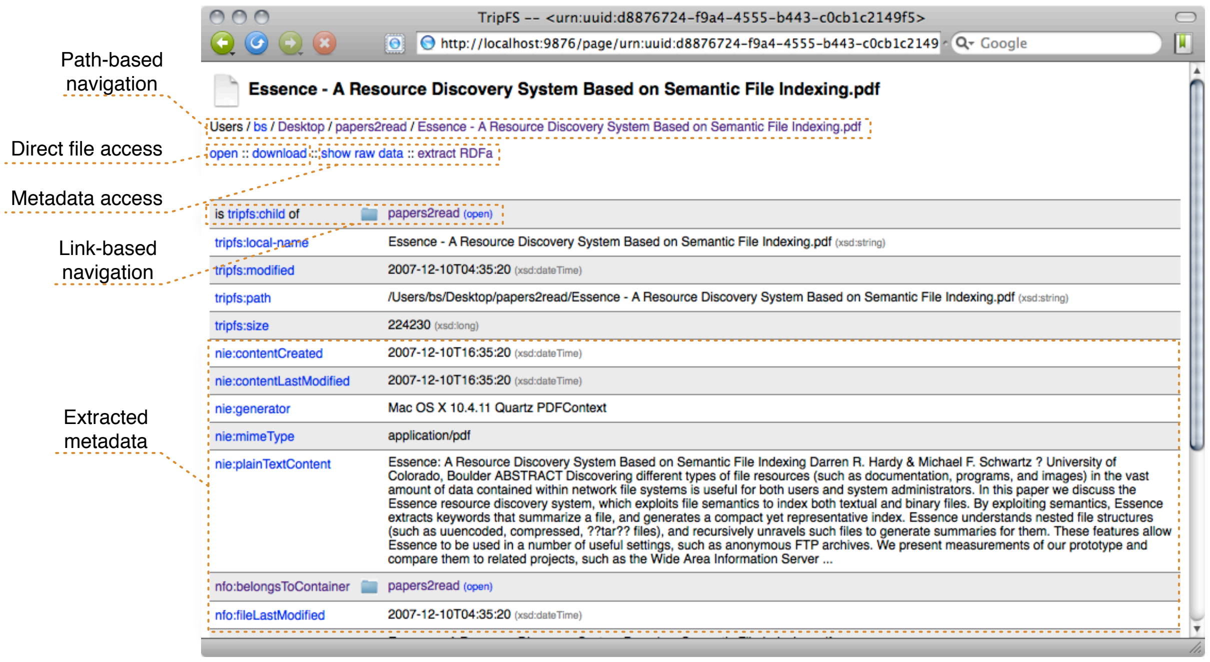
The TripFS core is a standalone server application, which has been implemented in pure Java, based on the Jena Semantic Web framework[9]. On startup, it crawls a configured sub-tree of the local file system, applies extractor and linker components to crawled files, and stores the resulting RDF triples in a triple store (either in memory or persistent). It initializes the *watcher* component to monitor the exposed file system sub-tree, which in turn notifies TripFS upon changes to files or directories. Subsequently, the RDF model is updated accordingly, and *extractors* and *linkers* are re-applied to the modified objects.

Metadata Extraction and Linking. We have implemented simple extractors that extract low-level file metadata, such as name, file size or a hash sum that could for example be used to identify and link equal files across different TripFS instances.

Further, we have implemented extractor components based on the Aperture metadata extraction framework[10], which provides a multitude of extractors for many different file types, including Office documents and multimedia data. As a proof of concept, we have also implemented several linker components: one that links documents, based on their titles, to resources in the DBLP data set; one that links audio files to MusicBrainz by analyzing track title and artist

[9]An evaluation version of TripFS can be obtained from `http://www.cs.univie.ac.at/tripfs`.

[10]`http://aperture.sourceforge.net`



TripFS -- <urn:uuid:d8876724-f9a4-4555-b443-c0cb1c2149f5>

http://localhost:9876/page/urn:uuid:d8876724-f9a4-4555-b443-c0cb1c2149

Essence - A Resource Discovery System Based on Semantic File Indexing.pdf

Users / bs / Desktop / papers2read / Essence - A Resource Discovery System Based on Semantic File Indexing.pdf

open :: download :: show raw data :: extract RDFa

is tripfs:child of	papers2read (open)
tripfs:local-name	Essence - A Resource Discovery System Based on Semantic File Indexing.pdf (xsd:string)
tripfs:modified	2007-12-10T04:35:20 (xsd:dateTime)
tripfs:path	/Users/bs/Desktop/papers2read/Essence - A Resource Discovery System Based on Semantic File Indexing.pdf (xsd:string)
tripfs:size	224230 (xsd:long)
nie:contentCreated	2007-12-10T16:35:20 (xsd:dateTime)
nie:contentLastModified	2007-12-10T16:35:20 (xsd:dateTime)
nie:generator	Mac OS X 10.4.11 Quartz PDFContext
nie:mimeType	application/pdf
nie:plainTextContent	Essence: A Resource Discovery System Based on Semantic File Indexing Darren R. Hardy & Michael F. Schwartz ? University of Colorado, Boulder ABSTRACT Discovering different types of file resources (such as documentation, programs, and images) in the vast amount of data contained within network file systems is useful for both users and system administrators. In this paper we discuss the Essence resource discovery system, which exploits file semantics to index both textual and binary files. By exploiting semantics, Essence extracts keywords that summarize a file, and generates a compact yet representative index. Essence understands nested file structures (such as uuencoded, compressed, ??tar?? files), and recursively unravels such files to generate summaries for them. These features allow Essence to be used in a number of useful settings, such as anonymous FTP archives. We present measurements of our prototype and compare them to related projects, such as the Wide Area Information Server ...
nfo:belongsToContainer	papers2read (open)
nfo:fileLastModified	2007-12-10T04:35:20 (xsd:dateTime)

Figure 5: Accessing local files via a Linked Data representation

name, and one that links files to potentially interesting DBpedia resources via the DBpedia lookup service. Both, the set of extractors and linkers are to be understood as proof-of-concept; by far they do not leverage the full potential of the presented approach. However, as described before, more extractors and linkers can be integrated easily according to the needs of an actual use case.

Maintaining Consistency. We have used DSNotify [19] as an implementation for the *watcher* component. DSNotify is a change detection add-on for datasources, supporting them in maintaining link integrity in their data. At its core, DSNotify extracts feature vectors from considered data entities that are used in heuristic comparisons to determine whether items that are no longer found at their original locations were in fact removed or moved to another location. DSNotify can easily be extended by implementing custom crawlers, feature extractors, and comparison heuristics.

We have implemented a generic file-feature extractor for DSNotify that extracts low-level features from local files (cf. Table 2)[11]. Further, we have developed a simple heuristic that calculates the plausibility that a file (described by the feature vector X) was moved to another location (the file there being described by the feature vector Y). This heuristic consists of two parts: first, plausibility checks are performed. For example, if the last modification date of file Y is before the one of file X, it cannot be a successor of X. Another example is that a file cannot become a directory or vice versa (checked by the *isDirectory* feature). Second, a similarity metric between the remaining features is calculated by using the strategies listed in Table 2. The resulting

Feature	Datatype	Similarity	Weight
Last access	Date	Plausibility	
Last modification	Date	Plausibility	
IsDirectory	Bool	Plausibility	
Checksum	Integer	Plausibility	
Name	String	Levensthein	3.0
Extension	String	Major MIME type equality	1.0
Path	String	Levensthein	0.5
Size	Long	Equality	0.1
Permissions	Bitstring	Equality	0.1

Table 2: Extracted features, their data type and the strategy used to calculate a similarity between them. Features that are used only in plausibility checks have a value *Plausibility* here.

similarities are weighted[12] (e.g., the name similarity is considered more important than equal file sizes), summed up, and normalized. These similarities are then used by DSNotify to detect *move*, *remove* and *create* events. Furthermore, DSNotify reports update events based on changes in the extracted feature vectors (cf. [19]).

DSNotify periodically monitors the file subtree that is exposed by TripFS, extracts feature vectors based on the file attributes described before, and stores these vectors in an index. DSNotify uses a native C++ component for efficiently monitoring the local filesystem that makes use of the Windows API *FindNextChangeNotification()* method. We

[11]The set of extracted features used by DSNotify is overlapping but not equal to the set of metadata attributes extracted and exposed by the TripFS. In the current implementation, these latter metadata are stored in the RDF graph while DSNotify stores features in its own indices.

[12]The selection of features as well as their weight was our own subjective choice based on several test-runs with the system. We consider an extensive evaluation of DSNotify as a tool for detecting file system events as future work.

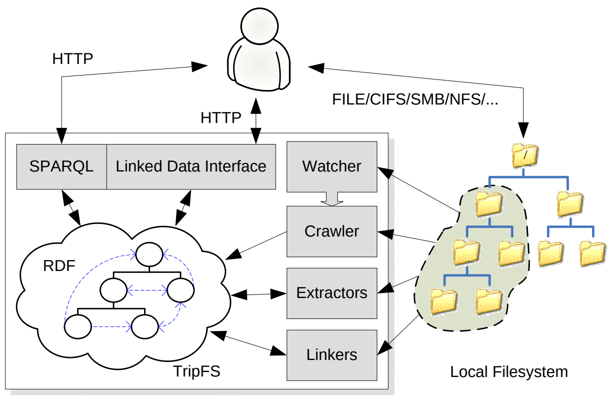


Figure 6: TripFS architecture

have also implemented a generic, yet less efficient Java-based monitor component that should work on all common platforms. This allows us to re-crawl the respective subdirectory tree only if there were actual changes reported by the operating system. The detected events are then forwarded to TripFS; the file's path is updated in the RDF model, and extractors and linkers are re-applied.

Linked Data Interface. TripFS includes an embedded Jetty Web server, which serves data from the triple store, as described in Section 3.7. It dynamically rewrites the internally used UUIDs and blank nodes to dereferenceable HTTP URIS, and provides XHTML+RDFa and pure RDF representations of file and directory resources, as well as a SPARQL endpoint. It further allows clients to directly download file contents and, in the case of local requests, to directly launch these files.

Neither component of TripFS makes any changes to the exposed file system; i.e., no special files or directories (like needed e.g., for SVN) are created. Currently, TripFS also does not provide means to modify file systems via the Linked Data interface.

5. RELATED WORK

Although modern file systems support the creation, storage, management, and retrieval of file-related metadata (e.g., using extended attributes or file forks), they remain mostly isolated from Web-based information integration and exchange contexts. Even file systems that provide sophisticated support for file annotations or links (e.g., LiFS [1] or AttrFS [23]) do not consider a global Web context but restrict their features often to objects within the local system. On the other hand, Web-based file systems usually focus on performance (e.g., [12]) or security (e.g., [4]), but not on semantically rich file descriptions or metadata interoperability. In this respect, TripFS can be seen as complementary to metadata-rich or highly scalable file systems in order to bridge the gap between file systems and Web environments. In combination with other works that represent Web resources as virtual file systems (e.g., [21]), local file systems and remote Web resources can be seamlessly integrated, providing unified programming interfaces and a consistent user experience.

As described before, file system contents are highly diverse and heterogeneous, and contain information that is valuable in many scenarios. TripFS presents a generic framework to expose these contents as Linked Data, but does not by itself extract higher-level metadata from files. For this, it relies on additional components, of which a wide variety exists. The Aperture metadata extraction framework was already mentioned before; it is based on the Gnowsis adapter framework [20] and is capable of extracting RDF descriptions from a wide range of files and other data sources. For most file types there exist extractors that return RDF descriptions of the file content, ranging from BibTeX files over calendar data to JPEG images; a list of these extractors is maintained at the W3C ESW Wiki[13]. Such conversion or extraction components exist also for Web sources, e.g., PiggyBank [15] or Virtuoso Sponger technology[14], which create RDF descriptions from a multitude of Web sources on the fly.

TripFS is in line with a number of other generic frameworks that allow one to expose Linked Data based on a different underlying data representation. Frameworks in this area include D2R [8] and Triplify [3] for relational data bases, SparqPlug [11] for DOM-based sources, OAI2LOD [14] for OAI-PMH repositories, and XLWrap [18] for spreadsheet data. With TripFS, file system contents can likewise be made "first-class citizens" of the Web of Data and can be seamlessly integrated with all these other data sources.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have presented and discussed TripFS, a service that exposes local file systems according to Linked Data principles. This approach potentially brings benefit to a range of application scenarios (cf. Section 2). In an enterprise information integration scenario (Scenario A), files are assigned stable, globally unique URIs and can therefore be referenced from external systems. Metadata that are extracted from files can be indexed by Semantic Web search engines, and links to other (enterprise-internal or external) data sources can increase the quality of information organization and data retrieval.

A lightweight component like TripFS can also be used in ad-hoc file sharing situations (Scenario B): participants in a face-to-face meeting can easily set up and start the sharing server, which exposes a certain sub-tree of their file system as Linked Data. This enables collaborators in the same network to access and retrieve these files, based not only on low-level characteristics like file name, but also using extracted semantic metadata and links. Using additional components, more intuitive approaches like faceted navigation can be performed on top of extracted data, and more experienced users are enabled to issue complex SPARQL queries over the file system.

A Linked Data representation of file systems also facilitates the application of Web-based annotation services (Scenario C), which overcomes the limitations of the hierarchical directory metaphor for file organization. Such annotations can refer to single files or even parts thereof, and can range from simple text-based comments to complex descriptions that may refer to external entities and concepts. TripFS makes file systems a part of a global, uniform Web of Data and therefore allows one to apply Web-based annotation techniques immediately to file system objects.

In future work, we plan an extensive evaluation of TripFS,

[13]`http://esw.w3.org/topic/ConverterToRdf`
[14]`http://docs.openlinksw.com/virtuoso/virtuososponger.html`

in particular regarding the performance and scalability of our approach. For this purpose, we aim to apply TripFS in a concrete enterprise information integration setting, and we plan to develop a simple user interface that allows end users to more easily share their files using Linked Data technologies. Further, we plan to improve and evaluate the accuracy of the DSNotify component for detecting file system events.

Additionally, we plan to introduce a more fine-grained model for selecting what file system objects are exposed via TripFS (currently one can select only a single subtree of the file system) and implement a secure HTTPS version that takes privacy considerations into account.

Acknowledgements

Parts of this work have been funded by FIT-IT grants 812513 and 815133 from Austrian Federal Ministry of Transport, Innovation, and Technology.

7. REFERENCES

[1] Sasha Ames, Nikhil Bobb, Kevin M. Greenan, Owen S. Hofmann, Mark W. Storer, Carlos Maltzahn, Ethan L. Miller, and Scott A. Brandt. LiFS: An Attribute-Rich File System for Storage Class Memories. In *Proceedings of the 23rd IEEE / 14th NASA Goddard Conference on Mass Storage Systems and Technologies*, 2006.

[2] William Y. Arms. Uniform Resource Names: Handles, PURLs, and Digital Object Identifiers. *Commun. ACM*, 44(5):68, 2001.

[3] Sören Auer, Sebastian Dietzold, Jens Lehmann, Sebastian Hellmann, and David Aumueller. Triplify: Light-weight Linked Data Publication from Relational Databases. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 621–630, New York, NY, USA, 2009. ACM.

[4] Arati Baliga, Joe Kilian, and Liviu Iftode. A Web-based Covert File System. In *Proceedings of the 11th Workshop on Hot Topics in Operating Systems*, 2007.

[5] T. Berners-Lee, L. Masinter, and M. McCahill. *Uniform Resource Locators (URL) (RFC 1738)*. Network Working Group, 1994.

[6] Tim Berners-Lee. *Linked Data*. World Wide Web Consortium, 2006. Available at `http://www.w3.org/DesignIssues/LinkedData.html`, retrieved 08-Aug-2008.

[7] Chris Bizer, Richard Cyganiak, and Tom Heath. *How to Publish Linked Data on the Web*, 2007. Available at `http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/`, retrieved 02-Dec-2008.

[8] Chris Bizer and Andy Seaborne. D2RQ - Treating Non-RDF Databases as Virtual RDF Graphs. In *Poster at the 3rd International Semantic Web Conference (ISWC2004)*, 2004.

[9] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data — The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3), 2009.

[10] Kendall Grant Clark, Lee Feigenbaum, and Elias Torres. *SPARQL Protocol for RDF (W3C Recommendation 15 January 2008)*. World Wide Web Consortium, 2008.

[11] Peter Coetzee, Tom Heath, and Enrico Motta. SparqPlug: Generationg Linked Data from Legacy HTML, SPARQL and the DOM. In *Proceedings of the First International Workshop on Linked Data on the Web (LDOW)*, 2008.

[12] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google File System. In *19th ACM Symposium on Operating Systems Principles*, 2003.

[13] Bernhard Haslhofer, Wolfgang Jochum, Ross King, Christian Sadilek, and Karin Schellner. The LEMO Annotation Framework: Weaving Multimedia Annotations with the Web. *International Journal on Digital Libraries*, 10(1), 2009.

[14] Bernhard Haslhofer and Bernhard Schandl. The OAI2LOD Server: Exposing OAI-PMH Metadata as Linked Data. In *International Workshop on Linked Data on the Web (LDOW2008)*, 2008.

[15] David Huynh, Stefano Mazzocchi, and David R. Karger. Piggy Bank: Experience the Semantic Web Inside Your Web Browser. In *International Semantic Web Conference*, volume 3729 of *Lecture Notes in Computer Science*, pages 413–430. Springer, 2005.

[16] Georgi Kobilarov, Tom Scott, Yves Raimond, Silver Oliver, Chris Sizemore, Michael Smethurst, Christian Bizer, and Robert Lee. Media Meets Semantic Web — How the BBC Uses DBpedia and Linked Data to Make Connections. In *Proceedings of the 6th European Semantic Web Conference*, pages 723–737, Berlin, Heidelberg, 2009. Springer-Verlag.

[17] Carl Lagoze and Herbert Van de Sompel. *ORE Specification — Abstract Data Model*. Open Archives Initiative, 2008. Available at `http://www.openarchives.org/ore/1.0/datamodel`.

[18] Andreas Langegger and Wolfram Wöß. XLWrap - Querying and Integrating Arbitrary Spreadsheets with SPARQL. In *International Semantic Web Conference*. Springer, 2009.

[19] Niko Popitsch and Bernhard Haslhofer. DSNotify: Handling Broken Links in the Web of Data. In *19th International WWW Conference (WWW2010)*, Raleigh, NC, USA, 2 2010. ACM. to be published.

[20] Leo Sauermann and Sven Schwarz. Gnowsis Adapter Framework: Treating Structured Data Sources as Virtual RDF Graphs. In *Proceedings of the 4th International Semantic Web Conference (ISWC 2005)*, pages 1016–1028. Springer-Verlag GmbH, 2005.

[21] Bernhard Schandl. Representing Linked Data as Virtual File Systems. In *Proceedings of the 2nd International Workshop on Linked Data on the Web (LDOW), Madrid, Spain*, 2009.

[22] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Discovering and Maintaining Links on the Web of Data. In *Proceedings of the 8th International Semantic Web Conference (ISWC 2009)*, 2009.

[23] C.E. Wills, D. Giampaolo, and M.S. Mackovitch. Experience with an Interactive Attribute-based User Information Environment. In *Computers and Communications, 1995. Conference Proceedings of the 1995 IEEE Fourteenth Annual International Phoenix Conference on*, pages 359–365, Mar 1995.