Managing and Querying Distributed Multimedia Metadata

Sébastien Laborie, Ana-Maria Manzat, and Florence Sèdes
Université Paul Sabatier

Nowadays, almost every human activity generates multimedia content. This content is acquired in real time and stored in different, heterogeneous locations. In order to deal with the proliferation of multimedia content and metadata, some research teams (see the “Related Work” sidebar) advise the use of distributed architectures where information can be stored on multiple remote servers that compute queries in parallel more efficiently than centralized metadata repositories. However, in this architecture, queries are sent to each server, which is not truly efficient because it’s possible to know in advance that only some servers will contain the desired information. Moreover, with the emergence of Semantic Web technologies, such metadata queries have become even more expressive and thus difficult to compute.

To manage and retrieve multimedia content efficiently in a distributed context, we propose a generic distributed framework where indexing not only is distributed across multiple servers, but also is centralized through a metadata resume that offers guidance about the location of the desired multimedia content. The originality of this resume is that it is not specified a priori, but rather is constructed automatically on the basis of the extracted metadata. Consequently, the resume has the benefit of indexing and retrieving some multimedia content reliably from very large amounts of acquired information without knowing the application context, such as video surveillance or patient medical records.

Consider the following use case where several police vehicles acquire different multimedia content, such as videos and images (car camera footage), audio (police radio transmissions) and text (police reports). As illustrated in Figure 1, the police vehicles are assigned to multiple sites and their acquired multimedia content is stored on local servers. To retrieve the desired information such as “a video containing a blue car in Paris with the registration plate number 1234 AB 56,” a central server must be aware of the distributed servers and capable of responding to the request for information.

In this context, the traditional method for answering user queries consists of storing all the metadata on the central server. This strategy has several drawbacks:

- **Query processing.** Querying a huge amount of centralized metadata might overload the central server.
- **Bandwidth restriction.** All multimedia content or all metadata must be transmitted across the network.
- **System centralization.** If the central server goes down, the metadata has to be recomputed on another server.
- **Copyright violations.** Specific metadata might not be stored on the central server because of copyright licenses.

To overcome these drawbacks, we can benefit from a distributed context by indexing and storing all multimedia content and corresponding metadata locally. Our idea is to transfer to the central server only a concise version of the distributed metadata. The central server could be used to answer very general queries or locate remote servers that might contain the desired specific information. Then, when a set of servers has been identified by the central server as containing data likely to match a particular user query, the query
could be computed locally on each of those servers.

This query architecture would minimize data transferring through the network because only a subset of the metadata stored on each remote server would be sent to the central server. Moreover, user queries would be computed only on specific remote servers and could be processed in parallel. Consequently, only some remote servers would be monopolized for processing a specific user query, while the others would preserve their computation resources for indexing new incoming media or responding to other kind of queries.

There are two projects closely related to ours. One is based on the NASA Earth Observing System where the metadata is distributed, queries are processed on remote servers, and not all queries go to all remote servers because of a descriptive hierarchy of the metadata contained on each server. The other project suggests using data summaries to query several databases efficiently, with the main idea consisting of grouping several values into one entity and thereby simplifying data descriptions and optimizing query evaluation.

Nevertheless, in these projects, the descriptive hierarchy and the summarized descriptions are constructed beforehand and are use-case dependent. In contrast, our proposed framework relies on a metadata resume that is built automatically from extracted metadata.

Related Work

In the past few years, a fair amount of research has been conducted in the field of multimedia indexing and retrieval. One difference in the various approaches taken is the indexing process, especially its generated output, which may be either nonsemantic metadata, like feature vectors, or semantic metadata. Some research projects consider metadata centralization while others recommend metadata distribution.

One project presents a centralized system for retrieving any kind of media. It employs the Support Vector Machine (SVM) for classifying and retrieving multimedia content. The features of the SVM are obtained from the combination of low-level descriptors and conceptual context obtained from Trecvid 2006 (see http://www-nlpir.nist.gov/projects/trecvid/) files.

The Candela project, which deals with video content analysis and retrieval focuses on the MPEG-7 metadata standard and proposes to store these descriptions in distributed relational databases. One of its limitations is that the mapping process between the semantic metadata and the database schema might lose information.

The Caim project focuses on context-aware image management in distributed and mobile environments. In this project, multimedia content, such as pictures, is acquired by mobile phones. This multimedia content is then sent to a central server to identify information about the content.

In another project, researchers present a distributed search engine based on mobile software agents. In this system, the multimedia content is distributed over several servers and some indexing agents migrate from one site to another. To limit the data transfer over the network, the researchers propose two architectures: one centralizing the index and the other distributing the indexes on remote servers. In the first architecture, a user query is executed on one centralized metadata collection, while in the second one the query is sent to all remote servers.

References

This architecture, based on the police example described previously, consists of several parts:

- **Multimedia content** refers to a single media item, such as a piece of text, an image, a video, or an audio snippet.

- A **multimedia collection** contains several pieces of multimedia content. Each multimedia collection is stored on a server dedicated to acquire remote site information (such as from police car cameras).

- A **set of extractors** applied to a given piece of multimedia content returns a set of content metadata. Examples of extractors have been proposed elsewhere for text, images, video, and audio.

- **Content metadata** contains information about the media characteristics (such as size, name, or file type) and its content (such as vehicle registration plate or person identification). Other research goes into more detail about content metadata for multimedia.

- A **metadata collection** contains all the content metadata describing objects from the multimedia collection. This metadata might be encoded according to several standards, such as Exif, Dublin Core, MXF, and so forth.

- A **metadata resume** is a concise version of the metadata collections. This resume is useful for identifying a set of servers that contains some specific descriptions. For instance, it might identify that a set of servers contains some multimedia contents about vehicles, without detailing the vehicle characteristics.

When a police vehicle acquires multimedia content, it is stored in a multimedia collection and indexed by multiple extractors. These extractors produce the corresponding content metadata. For example, if a police vehicle acquires a video, the extractors might identify some cars and their corresponding vehicle registration plates. All content metadata is then stored in a metadata collection. Finally, a subset of the metadata collections, named **metadata resume**, is transferred to the central server.

By using this generic indexing architecture, multiple indexing engines could be integrated into each server. More precisely, the extractors could correspond to several indexing engines. Furthermore, different engines could be located on each server.

**Applying metadata extractors**

In step A in Figure 2, an extractor applied to a piece of multimedia content must identify information about this content, such as the occurrence of persons, vehicles, discussion topics, and so forth. An automatic license plate recognition system would be one kind of extractor. Such extractors are key features of our system because they produce the semantic meaning of multimedia content. Moreover, when several extractors are applied to multimedia content, they can enrich its description considerably.

The following example illustrates how extracted Resource Description Framework (RDF) descriptions can be combined when manipulating several extractors. An RDF description can be represented by a graph, as shown in Figure 3.

Let $C_1$ be video content. We associate with $C_1$ a URI that uniquely identifies $C_1$ (for example,
Thereafter, we apply several extractors to \( C_1 \): an extractor of person’s hair color (\( E_1 \)) and an extractor of vehicle characteristics (\( E_2 \)). In this example, we assume that the extractor outputs are RDF descriptions or that their outputs are translated into RDF descriptions. Figure 3 illustrates the extracted RDF descriptions of the video content \( C_1 \). As shown in this figure, some extractors might consist of several extractors: a registration plate identification extractor (\( E_{2a} \)) and a vehicle color recognition extractor (\( E_{2b} \)).

When multiple extractors are applied to multimedia content, all extracted descriptions are aggregated (or linked) to the defined multimedia content URI. Hence, the proposed RDF description of Figure 3 means that \( C_1 \) is a video that contains a blond person and a blue vehicle identified with the registration plate number 123 AB 456. As this example illustrates, the construction of the content metadata, and especially RDF descriptions, can be fully automated.

**Constructing metadata collections**

A metadata collection contains several pieces of multimedia content metadata. It might also contain additional information about the server and also about the metadata itself, which has not been obtained by extractors. This additional information is defined in the literature as context. (Other work7 presents various definitions of context and its acquisition and modeling.) This additional information might be added manually to metadata collections or deduced with inference rules. It is important to note that, with or without this supplementary information, users could still retrieve semantic information about multimedia content in this framework.

As a specific example of step B in Figure 2, let \( Server_1 \) be a server metadata collection, and \( C_1 \) and \( C_2 \) two pieces of multimedia content. We associate with \( Server_1 \) a URI that uniquely identifies \( Server_1 \) and aggregate to this URI all RDF metadata descriptions produced by extractors (that is, all content metadata). Figure 4 illustrates the RDF description of the \( Server_1 \) metadata collection.

In Figure 4, some additional descriptions about the context, such as the location of \( Server_1 \) (colored in blue in Figure 4), are attached to the proposed RDF description. Moreover, thanks to the registration plate number, specific information about \( Vehicle_1 \) (such as the owner’s name) can be deduced and added to the RDF description (colored in light red in Figure 4). Other research8 shows how to model context in such cases by using an ontology and a reasoner to infer additional metadata.

**Summarizing a metadata collection**

In order to locate servers that contain particular information and to answer user queries efficiently, we propose storing on the central server only a concise version of all metadata collections (step C in Figure 2). If we look at the metadata collection presented in Figure 4, we could sum up this collection by saying that this server, located in Paris, contains a video about a person and a vehicle along with audio content. A more summarized version would be that this server located in Paris contains video and audio content. Thus, a metadata resume can describe part of a collection and can vary according to the summary details. To select part of a
Degree = 0 means that no information from a metadata collection is selected.

0 < degree < n means that only a part of the information contained in a metadata collection is selected. The size of the selected part depends on the degree value.

Degree = n means that the metadata collection is not summarized and all metadata descriptions are selected.

Consider the RDF description presented in Figure 4. To summarize this description, we can focus on the `Server_1` URI and select the information that is linked to this URI. Indeed, the server URI is the main subject of the summary. To build a summarized version of an RDF description, our idea is to select all metadata present on a path of a specific length, starting from the server URI. In this context, the length of this path will correspond to the summary degree.

We can apply this method to the RDF description in Figure 4:

- If degree = 0, the summary contains only the `Server_1` URI, that is, no information about this server.

- If 0 < degree < 4, the summary contains the `Server_1` URI and a subgraph of Figure 4. For instance, if degree = 2, the summary is specifying that the `Server_1` contains two pieces of multimedia content, an audio and a video file, and that there is a person and a vehicle on the video (see Figure 5).

- If degree = 4, the summary contains all the metadata.

This method for building RDF summaries is automatic and depends only on the metadata collection content. This generic approach can be used in multiple cases and can be supplemented manually by adding some important concepts or removing less important ones. In this framework, the RDF summaries, computed by each server, are sent to and merged on the central server. (A definition of the RDF merge operation can be found at http://www.w3.org/TR/2001/WD-rdf-mt-20010925/#merging.)

**Querying distributed metadata**

In the proposed framework, as shown in Figure 2, all metadata collections are distributed across multiple servers and a metadata resume is produced on the central server to respond to general queries or locate servers that might contain the desired information. Suppose a user wants to find videos that contain a particular vehicle and its registration plate. If some results are found, they are returned to the user. If no results are retrieved, it’s possible that some distributed servers will contain the desired information.

To locate these servers, we propose to transform query $q$ into $q_0$, such that $q_0$ is looking for servers that contain the constraints specified by the query $q$. In our example, $q_0 = \text{find servers with some videos that contain a vehicle and its registration plate.}$ However, because the central server contains only a concise version of the global metadata, $q_0$ might not retrieve some results, because $q_0$ contains variables that have no corresponding mappings in the metadata resume.

The metadata resume described in Figure 5 doesn’t contain information about registration plate numbers. To deal with such situations, the query could be relaxed, meaning some query constraints would be removed. For example, $q_0$ might be relaxed into $q_{00}$ such that $q_{00} = \text{find servers with some videos.}$ As a result, responding to relaxed queries on the central server would become faster. Finally, when a list of servers is computed, the initial query $q$ could be computed in parallel on each server.
to retrieve the desired information and display the results to the user.

The following example illustrates the applicability of this framework on SPARQL queries,\(^9\) the language for querying RDF graphs. Consider the query, encoded in SPARQL to find videos that contain vehicles and their corresponding registration plates:

\[
\text{SELECT ?video, ?plateNumber FROM RDF-resume WHERE} \\
?video \text{ typeOf ex:Video .} \\
?video \text{ contains ?vehicle .} \\
?vehicle \text{ plateNum ?plateNumber .} 
\]

If this query returns no results from the metadata resume, we can add to the query a variable ?server to retrieve results from particular servers: ?server typeOf Server. Because we know that a server contains multimedia content, we can also add to the query the triple pattern ?server contains ?video, where the variable ?video is the multimedia content that we are looking for.

As we described, we must relax the query constraints to retrieve some results. Several other projects have focused directly on query relaxation and in particular on SPARQL algorithms.\(^{10,11}\) The goal of this article is not to compete with such related work but to propose the use of a metadata resume, so we don’t present a relaxation algorithm here. However, it is still important to note here that removing some query constraints, that is, removing some variables, might be one approach to relaxing SPARQL queries.

**Experimental results**

We implemented our framework in Java using Jena (see http://jena.sourceforge.net/) for managing and storing RDF metadata, and ARQ (see http://jena.sourceforge.net/ARQ/) for querying this metadata with SPARQL. To be as general as possible, we generated random RDF graphs by specifying particular graph sizes and densities using RDFizer (see http://simile.mit.edu/wiki/Random_RDFizer).

In our proposed framework, multimedia content URIs are linked to remote server URIs. To preserve this information for each randomly generated RDF description, we randomly selected a specific RDF node and typed it as a server. Thereafter, we evaluated the efficiency of the metadata resume by measuring the query response time. For particular contexts, that is, specific number of servers and RDF graph sizes, our goal was to:

- run an initial query on a centralized metadata resume that hasn’t been summarized and on one remote server metadata collection;
- run a relaxed query on a centralized metadata resume that hasn’t been summarized and on another centralized metadata resume that has been summarized; and
- run a relaxed query that looks for servers that might contain the desired information on a centralized metadata resume.

Table 1 illustrates one initial SPARQL query (query 1), a possible relaxed query by removing some variables (query 2), and a relaxed query looking for servers (query 3). We chose these SPARQL queries to be as generic and exhaustive as possible. The relaxation method used is

<table>
<thead>
<tr>
<th>Query Benchmark</th>
<th>Query</th>
</tr>
</thead>
</table>
simple, to make it easier to see the relaxation influence on the results. The queries 4, 5, and 6, are more complex SPARQL queries with more variables. The results shown in the Tables 2 and 3 and in Figure 6 contain the response times obtained when computing the SPARQL queries on different RDF descriptions. All tests were done on a Mac OS X system with a 2.16-GHz Intel Core Duo CPU and 2 Gbytes of RAM.

In Table 2, the experiment results show that the RDF graph sizes have an influence on query response times. When we executed several instances of query 1, we found that answering queries on one remote server metadata collection was more efficient than answering queries on a centralized metadata collection that had not been summarized. This finding is not surprising because other research\textsuperscript{12} has shown that the complexity of computing SPARQL queries (especially simple SPARQL queries) increases according to the function of the RDF graph size to be queried and the number of query variables.

We also found that, for query 2 instances, returning results for relaxed queries was also quicker. Answering such queries on a summarized metadata collection was more efficient than querying a metadata collection that had not been summarized. For query 3, our results show that when a variable is fixed on the relaxed query (such as for retrieving servers), query evaluation is even more efficient.

Table 3 presents evaluations that are based on more complex SPARQL queries. These
results show the evolution of the response time for more complex queries. We found that the metadata resume still proved to be efficient in these tests.

On the basis on the overall results presented in Tables 2 and 3, we can conclude that querying a metadata resume with a relaxed query that determines some servers might contain relevant information, and then querying those particular servers simultaneously, is more efficient than querying a centralized metadata collection that has not been summarized. The results presented in Figure 6 demonstrate the efficiency of the resume when multiple queries are executed simultaneously on a central server. The results also show that evaluating five queries in parallel on a centralized metadata resume is more efficient than evaluating two queries in parallel on a centralized metadata collection that has not been summarized.

**Conclusion**

Our results from testing this proposed framework show that distributing metadata is more efficient than centralizing it, and that querying a metadata resume to locate servers that might contain relevant information can improve search performance. This RDF-based framework is going to be integrated in the context of the Lindo project (see http://www.lindo-itea.eu/), which is focused on managing distributed multimedia indexation. However, our proposed framework is not limited to one particular representation language. Indeed, it can handle other existing languages, such as XTM\(^{13}\) or Topic Maps models.\(^{14}\)

As for future work, we plan to continue to evaluate this proposed framework with some metadata based on RDFS\(^{15}\) and OWL,\(^{16}\) and more expressive query languages such as CPSPARQL.\(^{17}\) We also plan to experiment with the influence of the summary degree on the system’s performance, for example, finding the best summary degree.

Finally, we intend to test our proposal by mixing several languages in metadata collections, for example, some servers will contain RDF descriptions while others will contain topic map models. Consequently in order to produce a resume, a generic representation has to be defined.

**Acknowledgment**

This work has been supported by the Eureka project Lindo (ITEA2–06011).

**References**

4. J.L. Gauvain, L. Lamel, and G. Adda, “Audio Partitioning and Transcription for Broadcast Data...


Sébastien Laborie is a postdoctoral fellow at the Information Technology Research Institute laboratory, University Paul Sabatier, Toulouse, France. His research interests include using Semantic Web technologies for managing and retrieving distributed multimedia content. Laborie has a PhD in computer science from Joseph Fourier University, Grenoble. Contact him at Sebastien.Laborie@irit.fr.

Ana-Maria Manzat is a PhD student at the Institut National Polytechnique de Toulouse and a member of the Generalised Information Systems group at the Information Technology Research Institute laboratory, University Paul Sabatier. Her research interests include modeling of multimedia metadata in a distributed information retrieval system, Semantic Web technologies, and distributed information retrieval systems. Contact her at Ana-Maria.Manzat@irit.fr.

Florence Sédès is a professor, heading the Generalised Information Systems group at the Information Technology Research Institute laboratory, University Paul Sabatier. She is the director of the Information-Interaction-Intelligence National Research Network. Her research interests include adaptation within semistructured documents in different applications and contexts. Sédès has a habilitation à diriger des recherches in Computer Science from University Paul Sabatier. Contact her at Florence.Sedes@irit.fr.