Data Integration Hub for a Hybrid Paper Search

Jungkee Kim[1,2], Geoffrey Fox[2], and Seong-Joon Yoo[3]

[1] Department of Computer Science, Florida State University, Tallahassee FL 32306, U.S.A.,
jungkkim@cs.fsu.edu,
[2] Community Grids Laboratory, Indiana University, Bloomington IN 47404, U.S.A.
[3] School of Computer Engineering, Sejong University, 98 Gunja-Dong, Gwangjin-gu, Seoul, 143-747, Korea

Abstract. In this paper we describe the design of a hybrid search that combines simple metadata search with a traditional keyword search over unstructured context data. This paradigm provides the inquirer additional options to narrow the search with some semantic aspects through the XML metadata query. We demonstrate a paper search for a case study of the hybrid search, and describe a data integration hub to integrate those data dispersed on the Net.

1 Introduction

To discover and share heterogeneous resources on the Net has been a long term challenge since computer communication networks were popularized. There are two traditional approaches to organizing the data to be searched—one is structured data and the other is unstructured data. A Web search engine is a typical example of search on the Internet. Its technologies are rooted in information retrieval that represents search over the unstructured data.

Web search engines provide clues for resource location, but they have no semantic schema and often produce meaningless keyword search results. The Semantic Web is a ambitious extension of the Web. It also includes multiple relation links with directed labeled graphs by which machines like Web crawlers can interpret the relationship between resources. Meanwhile the ordinary Web has a single relationship and a machine cannot infer further meaning. To represent the relations of the object on the Web, the object terms should be defined under a specific description-an ontology. Domain experts are usually needed to design an ontology due to the sophisticated definition required. Currently, many Web pages included no such semantic content, and no unified definition of general semantic agreement exists.

Our hybrid keyword search aims to give an intermediate search paradigm on the Internet—providing semantic value through XML metadata that are simpler than those of the Semantic Web. In this paper we describe our design of hybrid search systems. In earlier experiments, we had suffered from performance problems in a local level, and we proposed scalable hybrid search on distributed environments [4, 5]. In the architecture, a group of independent search providers share

their information through their own search systems. But some data providers, who possess small amounts of data, may join such group. They may not want to develop their own search services. Otherwise, participation of many nodes possessing small data in a group will increase the communication traffic and drop a chance to reach the target information under Time-to-Live (TTL) strategy. Partial integration may be one possible method to increase the data portion queried in the search group. In this paper we also present our architecture for data integration hub, which is an application communicating through a message broker with centralized control. This hub can act as a partial integrator on distributed databases or peer-to-peer systems.

This paper is organized as follows. In the next section we describe one of our hybrid keyword search architectures. We present a data integration hub to integrate those papers in Section 3. In Section 4, we summarize and conclude.

2 Hybrid Keyword Search

Our hybrid keyword search combines metadata search with a traditional keyword search over unstructured context data. Each chunk of unstructured data, usually represented by a file, has an assigned metadata. We use XML—the de facto standard format for information exchange between machines—as a metalanguage for the metadata. To demonstrate the practicality of the hybrid keyword search, we design and evaluate hybrid search systems based on a native XML database and a file system based text search library, as well as a market-leading relational database management system that integrates XML and text management.

We have already introduced a relational database based implementation [3, 4]. It utilized an XML-enabled relational Database Management System (DBMS) with nested subqueries to implement the combination of query results against unstructured documents and semistructured metadata. The other implementation is based on a native XML database and a text search library. We use Apache Xindice [2] for XML instance repositories and XML query processing. Jakarta Lucene [1] is used to manage context query over unstructured data in our hybrid search. The query processing architecture is shown in figure 1.

In the Xindice database, we associate an XML instance with an unstructured document by assigning the file name for the document as the key of the XML instance. For example, an XML instance in a file named "pt1.xml" with unstructured data in the file "apaper.pdf" by inserting the XML instance into the data collection as follows:

```
xindice ad -c /a_collection_path -f pt1.xml -n apaper.pdf
```

The assigned key—the file name of unstructured document in the example—as an attribute value on the root element of the XML created by executing an XPath query. For example the query result may start:

```
<bibliography src:col="/a_collection_path" src:key="apaper.pdf"
xmlns:src="http://xml.apache.org/xindice/Query">
```

The key and result XML tuples are stored in a hash table. Another keyword search against unstructured documents returns names of documents which in-

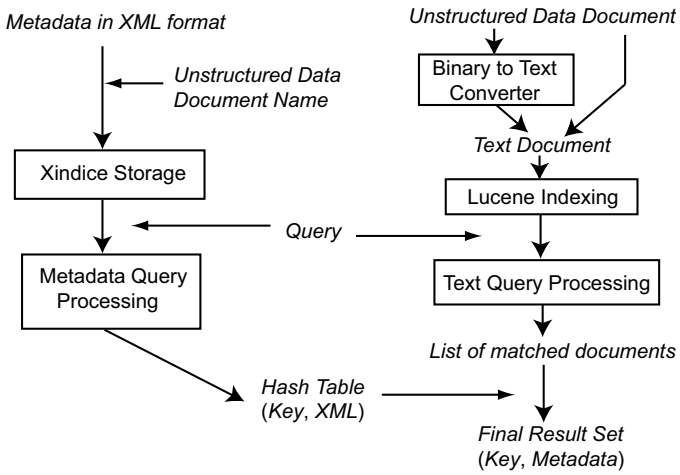

Fig. 1. Query Processing Architecture

clude the given keyword in the text. The returned names are mapped in the hash table and the combined results are collected in a Java hash table.

For efficient text search, Jakarta Lucene provides an index class along with a stop word filtering analyzer class. The analyzer filters out stop words—articles and other words that are meaningless to the search. Some binary format files should be converted to pure text files before indexing. We pass the binary file name as well as the text file name as parameters to the index generating class in order to indicate the original document format.

2.1 Case Study: Hybrid Paper Search

The initial demonstration of the hybrid search is in a simplified model—a paper search. The paper search is a content search across various types of documents. Each document has metadata presented as an XML instance. An example XML instance is shown in figure 2.

In this demonstration we use a relational DBMS instead of a native XML database. Two relational database tables are used for metadata and documents. For the XML instances representing the metadata, the XMLType of Oracle 9i [6] is used. A column with *BFILE* large object type is used for the external document table. Those large object rows are indexed using Oracle Text—a text management system integrated into Oracle DBMS. Through an Oracle Text index, we can search the target content. The document table has a special attribute for a document type—*BINARY* or *TEXT*. This attribute is necessary for the filtering option of Oracle Text. Oracle Text filters binary files to pure text instances before making an index. A one-to-one relationship set is used for relation between the paper document and metadata, but a one-to-many or many-to-many

```
<bibliography>
  <authors>
    <author>J. Kim</author>
    <author>O. Balsoy</author>
    <author>M. Pierce</author>
    <author>G. Fox</author>
  </authors>
  <title>Design of a Hybrid Search in the OKC</title>
  <source>Proceedings of the International Conference on IKS</source>
  <year>2002</year>
</bibliography>
```

Fig. 2. An Example XML Instance

relationship set could be used for other applications. The relationship table is not necessary in one-to-one relationships, but it is essential to decompose relations to avoid anomalies in one-to-many or many-to-many relationship. With two data tables and a relationship table we can query keywords in the content, which can be associated with particular metadata through nested subqueries. For example, we can find documents with a keyword "XML" and published in 2002. Figure 3 shows the relational schema of our hybrid paper search.

```
papers(paperND: string, descriptions: XMLType)
paperfiles(filename: string, doctype: string, contents: BFILE)
filelocator(paperND: string, filename: string)
```

Fig. 3. Relational Schema of Hybrid Paper Search

3 Data Integration Hub for Hybrid Search

In our earlier work [5] we assumed query clients only read resource in the other nodes. In this paper we take into account that clients may desire to be a data provider without providing an independent search service. Several or many data providers can share their information through a centralized database. They may upload, query, and download unstructured data with attached metadata via a central DBMS.

Another aspect for the integration hub, which was not introduced in the local hybrid paper search, is the metadata validation against XML schema. The storage for the local hybrid paper search is managed by database administrators, but ordinary users can upload their own data to the database of the integration hub. We utilize an Oracle DB operation to check the validity of metadata presented in XML instances against a registered XML schema. The XML schema for our hybrid paper search is shown in figure 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="bibliography">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="authors">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="author" type="xs:string"
                maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="title" type="xs:string"/>
        <xs:element name="source" type="xs:string"/>
        <xs:element name="year" type="xs:decimal"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Fig. 4. An XML Schema Example for Hybrid Paper Search

We demonstrate a data integration hub on the central DBMS using message-oriented middleware. This integration hub is an integrated version of the hybrid paper search introduced in section 2.1. The general architecture of a data integration hub is summarized in figure 5. Those clients can communicate with an integrant through a message broker—NaradaBrokering [7]. This broker includes JMS compliant topic-based communication—a publish/subscribe model. As in the read-only query case, clients are publishers and the integrant is a subscriber. Clients and the integrant use the same topic. The integrant manages uploading of the metadata and unstructured data, query wrapping, and file transfer. Requests are classified by the job property, which is attached to the message sent from client to integrant. A *temporary topic* is also delivered to the integrant. This temporary topic provides unique identification for the requester client, and the integrant can return back to the client by publishing the message on a dynamic virtual channel—a temporary topic whose session object was attached to the message.

We use different message types for each service request, as follows:

– Upload request: initially the client should request to upload to the integrant, before the unstructured data files are sent. The message for this request includes a file name of unstructured data, metadata, and unique client user name. They are string contents in a `MapMessage` type message. The integrant checks the validity of the metadata against XML schema as clients may provide no well-formed or valid data. The central database should avoid

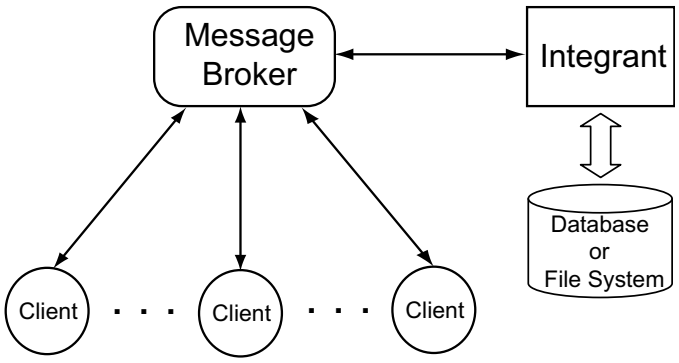

Fig. 5. An Integration Hub Architecture

redundancy for the unstructured data stored in a file system. The name and directory of the unstructured data, which is generated by combining the user name and the file name, is used for a primary key.

- File upload: after an upload request is granted by the integrant, the client can send a file for the unstructured data. For the file uploading, we use a `ByteMessage` type message, which is appropriate for sending a stream of uninterpreted bytes. The client sends a header message first, and the file is published buffer by buffer. A job property, a file name, a user name, and a temporary topic are included in the header message. A JMS message has a unique ID when it is created, and the ID is attached to each message. The integrant can classify those file upload messages by extracting the message ID. This mechanism is necessary because several file uploads can occur simultaneously.
- Query request: clients are publishers for a query topic, and the integrant subscribe on the same topic. A `MapMessage` type message, which includes query parameters and properties, is published to the integrant. The query results are delivered back to the inquiry client subscribed to a temporary topic.
- Download request: the target unstructured data in a file can be obtained from the query request. The client subsequently requests a file download with a `MapMessage` type message that includes the file name and a temporary topic. The listener on the client then captures the `ByteMessage` type message published from the integrant, and the target file is written message by message on the client machine. Each message includes the file name property. The message broker is responsible for preserving order of message transfer.

The database schema of our data integration hub are similar to those in the hybrid paper search in figure 3, but there is an additional table for the file uploading for the unstructured data—a temporary file locator table. Our system allows the file upload on a temporary directory only, and moves the

files to the designated directory later. When a user request a file upload and incidentally the same file name already exists, a new file name is assigned for the final destination by the integrant. The original file name is stored in the table for metadata, but the actual file name is stored in the unstructured data table. We assume that a user does not assign the same file name to two or more different unstructured data. A naming and directory for each row in the paper metadata table is generated from combining the unique user name and the file name, and it makes the naming and directory to a potential primary key.

Each data integration hub has a message broker and an integrant. A group of data integration hubs may provide a global search over a distributed information system by using the cooperative network features built in to NaradaBrokering, or by using an additional network layer—a peer-to-peer overlay network.

4 Conclusion

In this paper we described our approaches to hybrid search at a local level with a case study of a hybrid paper search. Our demonstration showed the possibility of the hybrid search paradigm for a practical semantic integration. We had another case study of a data integration hub, which is an application communicating through the message broker with a centralized control. The integration hub can be a search service node in a distributed database, or a peer in a peer-to-peer overlay network under more scalable environment [4, 5]. This scalable generalization may have a practical bridging role for information search—providing semantic value through metadata whose implementation are simpler than those of the Semantic Web.

References

1. Apache Software Foundation. Jakarta Lucene. World Wide Web. http://jakarta.apache.org/lucene/.
2. Apache Software Foundation. Xindice. World Wide Web. http://xml.apache.org/xindice/.
3. J. Kim, O. Balsoy, M. Pierce, and G. Fox. Design of a Hybrid Search in the Online Knowledge Center. In *Proceedings of the IASTED International Conference on Information and Knowledge Sharing*, November 2002.
4. J. Kim and G. Fox. A Hybrid Keyword Search across Peer-to-Peer Federated Databases. In *Proceedings of East-European Conference on Advances in Databases and Information Systems (ADBIS)*, September 2004.
5. J. Kim and G. Fox. Scalable Hybrid Search on Distributed Databases. In *Proceedings of International Workshop of Autonomic Distributed Data and Storage Systems Management*, volume 3516 of *Lecture Notes in Computer Science*. Springer, May 2005.
6. Oracle Corporation. *Oracle9i Application Developers Guide—XML*, June 2001.
7. S. Pallickara and G. C. Fox. NaradaBrokering: A Distributed Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids. In *Proceedings of International Middleware Conference*, June 2003.