Scalable Hybrid Search on Distributed Databases

Jungkee Kim[1,2] and Geoffrey Fox[2]

[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.
gcf@indiana.edu

Abstract. We have previously described a hybrid keyword search that combines metadata search with a traditional keyword search over unstructured context data. This hybrid search paradigm provides the inquirer additional options to narrow the search with some semantic aspect from the XML metadata query. But in earlier work, we experienced the scalability limitations of a single-machine implementation. In this paper, we describe a scalable hybrid search on distributed databases. This scalable hybrid search provides a total query result from the collection of individual inquiries against independent data fragments distributed in a computer cluster. We demonstrate our architecture extends the scalability of a native XML query limited in a single machine and improves the performance for some queries.

1 Introduction

With the popularity of computer communication networks, there have been many efforts to share and exchange information between resources on the Net. There are two main approaches to search on networks—searching over structured data or searching over unstructured data. An archetypal example representing structured data is a relational database, while information retrieval represents search over the unstructured data. Extending these approaches to the Internet environment uncovers new research areas. The theories of SQL queries over structured data give way to XML queries—searching over semistructured data—while newly developed Web search engines are based on information retrieval technologies.

Our hybrid keyword search paradigm lies between those two approaches. The hybrid search is fundamentally based on keyword search for the unstructured data, and adds supplemental search on metadata attached to each unstructured document. We adopt XML—the de facto standard format for information exchange between machines—as a metalanguage for the metadata. We demonstrated the practicality of the hybrid keyword search in [8, 9], but we also experienced the scalability limitations of a single-machine implementation. Particularly, the native XML database we used had very limited scalability. In this paper, we will adopt a distributed strategy to improve the scalability of hybrid

keyword search. We will experimentally illustrate performance improvement and larger scalability of the new architecture.

The rest of this paper is organized as follows. In the next section we describe our architecture and compare to a grid system. Section 3 describes our search architecture on distributed databases. We illustrate a query processing architecture on a distributed database in Section 4. In Section 5, we evaluate our hybrid search over a distributed database. We summarize and conclude in Section 6.

2 Hybrid Keyword Search on Distributed Databases

A distributed database is a collection of several or many databases stored in different computers and *logically interrelated* through a network. A distributed database management system (DDBMS) can be defined as a software application that manages a distributed database system so that to users it seems like a single machine—it provides transparency. Let us describe our distributed architecture according to the transparency criteria of [11].

- Data Independence: the hybrid searches on each machine are logically and physically independent each other. The data of each database have the same schema and the schema do not change. Each database has its own management system and only returns query results against the user inquiry. So our architecture has physical data independence due to data encapsulation in each machine.
- Network Transparency: the network connection of our system depends on message-oriented middleware or peer-to-peer middleware, and the middleware administrator or agent is only concerned with the connection. The end-user does not perceive the detailed network operation of the distributed hybrid search inquiry.
- Replication Transparency: we assume our distributed architecture is restricted to a computer cluster, and no replication exists in the cluster. Simply: our system is replication transparent. Data replication is usually necessary to increase the locality of data reference in a distributed environment. Actually, the locality is guaranteed in a clustering architecture.
- Fragmentation Transparency: our experiments with hybrid search on a distributed database will show that the data can be partitioned into each database within the limitation of the local database. Full partition is the easiest case for the distributed database management system. The data are partitioned into a chunk of XML instances with their associated unstructured data, and this type of fragmentation is *horizontal*.

We can summarize that in our architecture each database is totally independent. The query result for the distributed databases is the collection of query results from individual database queries.

Data independence and horizontal fragmentation features make our architecture different from other general architectures for federated database systems

surveyed in [13]. If data are not independent and user inquiries require joining query data from different machines with various schemas, we can consider a distributed querying framework based on recently emerging Grid infrastructure [5, 4]. OGSA-DQP (Open Grid Services Architecture; Distributed Query Processing) [1] is an example of such framework.

3 Distributed Database Architecture

The architecture of the hybrid search service on each local machine depends on the local service provider who joins the distributed database, and we demonstrated such architectures in [8, 9]. One is utilizing an XML-enabled relational DBMS with nested subqueries to implement the combination of query results against unstructured documents and semistructured metadata. The other is based on a native XML database and a text search library. To associate metadata with unstructured documents, we assign the file name for the document as the key of the metadata. A hash table is used for a temporary storage of metadata query results and the keyword search maps to the table subsequently for joining.

The remaining issue for organizing the distributed database is the computer network technology that connects the databases, and provides the network transparency to the user. Tanenbaum and Steen [14] suggested that message-oriented middleware is one of best application tools for integrating a collection of databases into a multidatabase system. We utilize a message-oriented middleware implementation—NaradaBrokering [12, 6]. It includes JMS compliant topic-based communication, which meets the minimum requirements for the network transparency in a distributed database. NaradaBrokering also provides a cooperating broker network for increased network scalability.

Figure 1 summarizes our general architecture. The search client is a publisher for a query topic, and a group of search services subscribe on the same topic. When the client publishes the query, the query message is broadcast to the search service subscribers. The query results from the search services are returned back to the client by publishing the message on a dynamic virtual channel—a *temporary topic* whose session object was attached to the query message originally delivered to the search service.

The architecture can be grown to a cooperating broker network. One or more heterogeneous search services could be attached to each broker and each broker can relay messages to other brokers. The cooperative network usually follows a hierarchical structure, for better performance.

4 Query Processing

The query processing of each database (fragment) on our distributed architecture depends on the local database or the search library. Due to the full partition in our database distribution, the query processing in the DDBMS is simple, and query propagation and result collections are the only interesting processes.

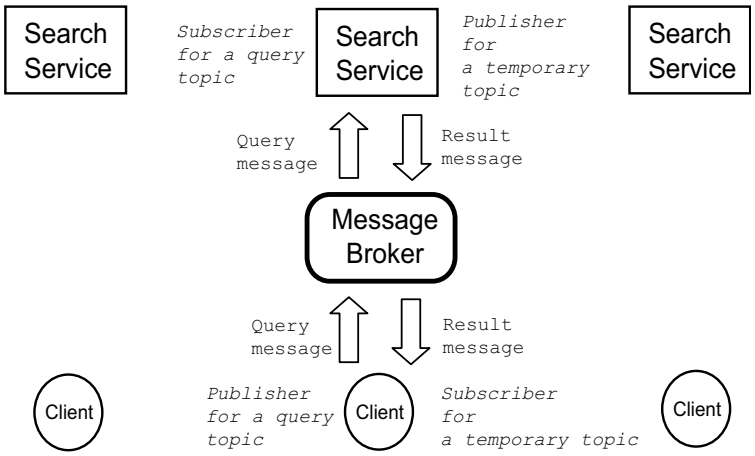


Fig. 1. Scalable Hybrid Search Architecture on Distributed Databases

A query processing architecture on a distributed database is shown in figure 2. The search client is a query publisher and the search service is a query subscriber. The user types query parameters through a user interface, and they are amalgamated into a message along with a job property and a temporary topic. The job property in this case is an integer property assigned to the "QUERY" final variable. The JMS `MapMessage` message type is used for the query message in our Java program.

The content of this message is a set of *name-value* pairs. The name is always a string and the value can have one of several allowed types. We used a string and an integer type. Those values could be empty for the string, and zero for the integer, if there is no user input. The `JMSReplyTo` property is a message header that contains a temporary topic.

The message broker delivers query messages to search services that have already subscribed to the same topic. The listener in the search service captures the query message, which includes a property header filled with a temporary topic. The extracted query parameters are passed to the local query processing service, which produces query results.

Those query results are returned back only to the client that published the query message. This works because the temporary topic is unique to this client. The inquirer client listens for the returned messages, and displays query results to the user.

5 Experimental Performance

In this section, we evaluate our hybrid search over a distributed database. A cluster of 8 computers forms a distributed environment, and all computers are located within a local network whose network bandwidth is very high.

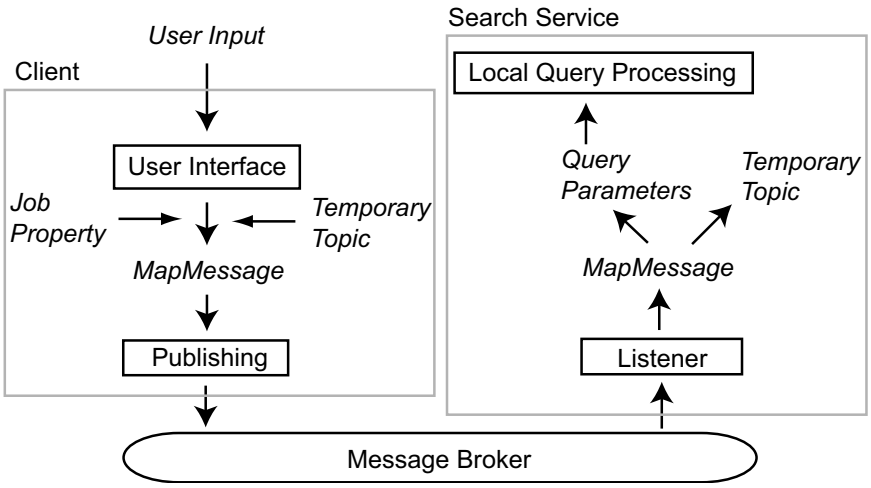


Fig. 2. A Query Processing Architecture on a Distributed Database

In these experiments we use 100,000 XML instances extracted from the *DataBase systems and Logic Programming* (DBLP) XML records [10] and 100,000 abstract text files from the TREC collection, OHSUMED [7]. We select article XML instances only, and non-ASCII characters are converted to ASCII characters. We cut off embedded metadata in the collection and extract the abstract part only. The data set is horizontally partitioned into 8 fragments, and each fragment on each machine has 12,500 XML instances and 12,500 text files. All eight computers have the same specification—2.4 GHz Intel Xeon CPU with 2 GB of memory, running a Linux 2.4 kernel and Java Hotspot VM 1.4.2. Those machines are connected with each other by 1 Gbps links, but the switch for the outside connections has 100 Mbps bandwidth. Apache Xindice 1.1b4 [3] is used as a native XML database, and Jakarta Lucene 1.3 [2] is utilized for text management.

We use two different communication middlewares—JXTA version 2.3 and NaradaBrokering version 0.96rc2—for the performance comparison, and three experimental architectures for 8 nodes as follows:

- JXTA: 1 node acts a rendezvous node and all other 7 nodes are connected to the rendezvous node. All of the nodes are located in the same subnet and all the query propagations are broadcast only within the subnet. This is because the current JXTA implementation does not meet the specification for selective query propagation from the rendezvous peer.
- Single NaradaBrokering: 1 node has a server and a search service, and the other nodes only have search services. All the search services are clients for the communication.
- Multiple NaradaBrokering Cluster: 1 node has a root server and the other nodes have second-level servers. Each node has a NaradaBrokering server

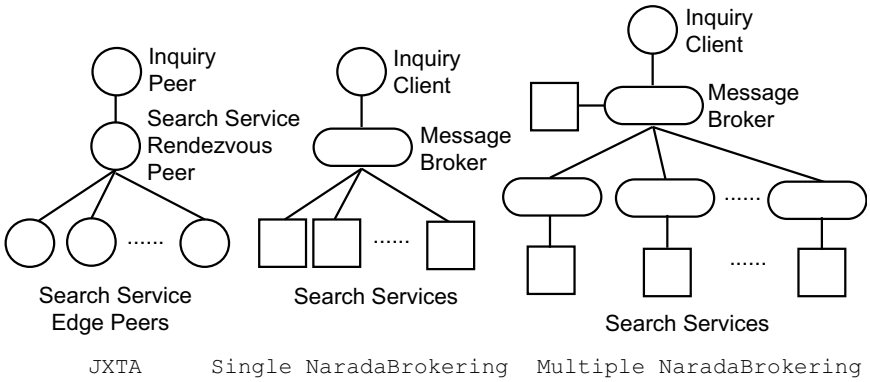


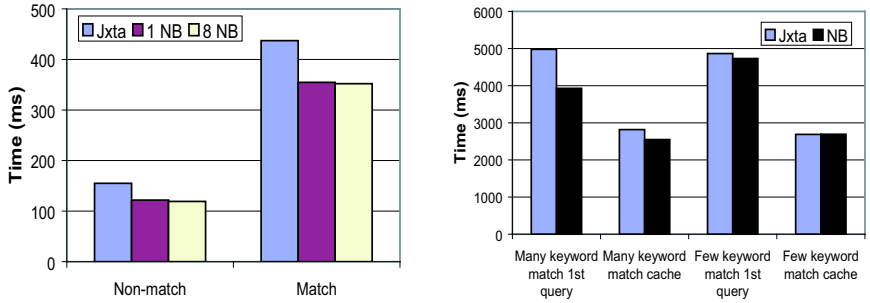
Fig. 3. Examples of communication middleware architectures

and a search service. This architecture gives us an idea of the performance difference between a cooperative network and a single message broker.

Examples of these architectures are shown in figure 3.

Figure 4(a) shows the average response time for an author exact match query over 8 search services using the three approaches. We choose 8 queries that combine an author and a keyword, and the databases are indexed against the author element in the XML instances and keywords for the unstructured data. Each query matches to only one of eight search services. The number of matches is between 1 and 3. We present the graphs separately for the average response time of no match and match result cases. We can interpret the difference between matched and non-matched query time to mean that the additional overhead for processing matched results is more than half of the total query time. The local processing time for non-matched query is very short as a result of the indexing against the author name. But the matched query takes long for joining the query results, and it is larger than the communication time. The query time of NaradaBrokering connections is shorter than that of JXTA connections. The time for eight connections of NaradaBrokering is a little shorter than 1 broker connection. But considering standard deviations—more than 70 ms—the two NaradaBrokering cases are statistically indistinguishable. NaradaBrokering uses Java NIO—the new I/O for better performance provided in JDK 1.4. JXTA version 2.3 does not use the Java NIO features.

We evaluate further hybrid search queries for the year match. In these queries, there are two different keyword selections. One has a few keyword matches—only 4 documents in 100,000 unstructured data documents, and the other has many keyword matches—41,889 documents out of 100,000. The year in the query is "2002" and it is hit in 7,752 out of 100,000 XML instances. We only show single NaradaBrokering case because multiple NaradaBrokering cluster had little difference. Figure 4(b) shows the average response time for the year match hybrid



(a) Author exact match (b) Year match

Fig. 4. Average response time for queries over 8 search services

query over 8 search services in the cluster. Compared with the performance experiment results on a single machine [3] running an XML-enabled commercial database, the year match queries on the distributed database show dramatic improvement in performance. The query time is improved from 82.9 seconds to less than 0.3 second for a few keyword matches, and from half an hour to less than 0.3 second for many keyword matches. This big performance improvement derives from *horizontal partitioning*, which reduces data retrieving time from disk with the relation partitioning. We divide those data into each machine by continuous ranges, and it is called *range partitioning*. The second and later repeats of the same inquiry take a shorter time than the first attempt. This is because the internal cache on the databases improves the query response time for recently answered inquiries. The query time in NaradaBrokering middleware is faster than that in JXTA, similar to the author matches.

Through experimental performance tests, we established that a distributed database can overcome the limit in target size of XML query when Xindice is run on a single machine. There was no problem up to 100,000 target XML instances in the query over the distributed database. The local native XML database could not produce accurate query results when the number of stored XML instances was over 16,000. Another contribution from the distributed query processing is the performance improvement for some queries over large target results. Our experiment was focused on the exact match, because Xindice does not provide context-based indexing on XML elements and attributes—there was no performance improvement from XML element indexing for an approximate match search.

[3] This machine was used for performance tests in [9]. We should add that since this paper was originally submitted we evaluated newer version of the commercial database in one of the cluster computers with large resource allocations. With this modified configuration, the query time was less than 1 second for the many keyword matches.

6 Conclusion

In this paper we described a scalable hybrid search on distributed databases. Those distributed architectures are mainly based on several or many computer network connections utilizing a message-oriented middleware or a peer-to-peer network framework. The hybrid search provides a total query result generated from a union of queries against data fragments in a computer cluster. The aspect of horizontal partitioning for our architecture contributed a performance improvement for some queries comparing to those on a single machine. Furthermore our new architecture extended the scalability of Xindice XML query, limited to a small size on a single machine.

References

1. N. Alpdemir, A. Mukherjee, N. Paton, and P. Watson. Service-Based Distributed Querying on the Grid. In *Proceedings of International Conference on Service Oriented Computing (ICSOC)*, December 2003.
2. Apache Software Foundation. Jakarta Lucene. World Wide Web. http://jakarta.apache.org/lucene/.
3. Apache Software Foundation. Xindice. World Wide Web. http://xml.apache.org/xindice/.
4. F. Berman, G. Fox, and A. Hey, editors. *Grid Computing: Making The Global Infrastructure a Reality*. John Wiley & Sons, 2003.
5. I. Foster and C. Kesselman, editors. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2003.
6. G. Fox, S. Pallickara, and S. Parastatidis. Towards Flexible Messaging for SOAP Based Services. In *Proceedings of International Conference for High Performance Computing and Communications(SC)*, November 2004.
7. W. Hersh, C. Buckley, T. Leone, and D. Hickam. OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th Annual ACM SIGIR Conference*, 1994.
8. 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.
9. 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.
10. M. Ley. Computer Science Bibliography. World Wide Web. http://www.informatik.uni-trier.de/~ley/db/.
11. T. Ozsu and P. Valduriez. *Principles of Distributed Database Systems*. Prentice Hall, 1999.
12. 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.
13. A. Sheth and J. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3):183—236, September 1990.
14. A. Tanenbaum and M. Steen. *Distributed Systems: Principles and Paradigms*. Prentice Hall, 2002.