Campus-wide Distributed Library Search System

Nitin Kaulavkar
G. Sivakumar
Department of Computer Science and Engg.,
Indian Institute of Technology,
Bombay, India. Pin:400 076.
nitin@cse.iitb.ernet.in

Abstract

This paper discusses the mechanism of accessing multiple libraries geometrically distributed in an academic institute in the uniform manner. Searching of any library item in such a system may amount to blindly querying every single participating database in the system (broadcasting the query)[1]. If the information about these databases and their holdings is kept in the form of indices distributed over a group of brokers, the queries can be forwarded to a set of appropriate databases, thus avoiding a broadcast. The proposed distributed library search system is based on Client-Server Model and consists of a collection of database servers which maintain the multiple library databases in the campus, brokers which hold a variety of distributed indices, and clients. This approach helps minimize the network traffic which rises otherwise due to broadcasting, and reduces the average search time. The design approach here resembles that of a heterogeneous, bottom-up distributed database design.

1 Introduction

Nowadays a typical academic institute consists of many departments, each having various libraries associated with laboratories, research and sponsored projects, or the individual faculty members. With the advent of Local Area Networks, fast workstations, and other computing resources, it is possible to access all these libraries uniformly, and view them as one distributed library system. The design of such a system has its own difficulties. There exist various similar systems and the most important of them being ALEPH [3], a system described in [1], MULTIBASE [4] and SIRIUS-DELTA in [2].

We present the networked Client-Server model for the Campus-wide Distributed Library Search System for accessing and retrieval of the on-line library catalogues available on multiple databases in this system. The client contacts a set of appropriate database servers with the help of various brokers available in the system. A broker has a table of indices which topic-wise summarizes the holdings on the available databases in the system thus ensuring thorough yet efficient searches.

2 An Overview of the System

The proposed system has a group of clients, a group of database servers managing the library databases at the various libraries in various departments of the institute. One physical database server may manage more than one library databases. The servers are assumed to be residing on the Unix machines. There are a variety of brokers in the system having one or more indices resolving the proper database servers; so when contacted by a client with a particular type of a query, query gets forwarded to a set of appropriate database servers.

The queries where only the brokers need to be contacted are called the Meta-queries. Upon contacted the broker searches its own tables/indices for a given query or a meta-query and either gives back the list of database servers which may satisfy that query or just gives pointer to another broker which clients need to contact in order to get the list of appropriate database servers. When client gets an answer to the meta-query it either forms one more meta-query for another broker or fires the main query to the database server depending on the answer. Client allows some kind of local customization in terms of physical topology, database distribution, library locations etc. This information helps in efficient query processing as far as time and network bandwidth are concerned. When the database server receives the query, it searches its own database/catalogues to satisfy the query and sends the results back to the
3 System Architecture

This section deals with the major components of the system, namely client, server and broker.

3.1 Client

The client offers user-interface for users in the system, provides users the means of local customization, converts queries into common query format, contacts the brokers and database servers to get the results for the user queries and displays them to the users.

It constitutes of the following sub-units.

The User Interface facilitates searching and querying. The default interface would be the text-only interface with the provision for browsing through the results and saving the results so that the user can make use of them in forming the later queries.

Users can select one of the fixed number of supported queries with the fixed possible values of some of the fields. By restricting the number of possible queries we are simplifying the query analysis, processing and optimization.

The Processing Unit transforms the user query with appropriate parameters from the user interface into the common query format. Depending on the type of query, it prepares an appropriate meta-query to be passed on to the broker. Every client knows at-least one broker for at-least one of the main query items (e.g. subject broker or department location broker etc.). Client waits for the broker to give back meta-query results. The processing unit then finds out from the result the appropriate server or yet another broker to be contacted in order to know about the proper servers.

Once the results are ready, it gives back those results to the user interface with appropriate translation of the results in the form that is acceptable to the user interface.

The Communication Unit is responsible for sending formatted query or meta-query duly prepared by the Processing Unit to the destination, either to a database server or a broker. It also receives the results from brokers or database servers.

The client program, as such, is a single-user program and so its send-request and receive-response are blocking in nature, in the sense, the next send from the client gets blocked till the response to the previous send reaches. As we are dealing only with the higher layers of the network, we do not have to bother about the byte-order, mismatch in the internal format across the machines etc. The Processing Unit has to resolve heterogeneity (if any) only at the database schema level.

3.2 Broker

The broker serves as a distributed index or a distributed catalogue which summarise the holdings on various database servers in the system.

All brokers have one or more of the following indices.

1. Department Location Index: This table keeps a list of departmental library brokers or department's main brokers for each and every department in the system. If this broker does not have the actual address/id of the departmental library brokers then it contains the entry to the next higher up broker which contains that information. So each entry is a pair of either <Dept, Departmental Library Broker> or <Dept, Pointer-to-next-suitable-broker>. Note that all component units in the system can be uniquely identifiable. In Table 1, Flag indicates whether the entry shows actual server or a pointer to another broker, shown by ↑.

2. Departmental Library Index: Each of the departmental library brokers would have index containing a list of the libraries in the entire department. So each entry has the structure <Dept, List-of-libraries>. The sample index for one department is shown below.

3. Topic or Subject Index: This is the table of the actual database servers with the various topics as the keys. There would be some topics for which this broker would know the exact

Table 1: Sample Department Location Index

<table>
<thead>
<tr>
<th>Dept</th>
<th>Flag</th>
<th>Server or a pointer to broker</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSE</td>
<td></td>
<td>kailash</td>
</tr>
<tr>
<td>CESE</td>
<td>↑</td>
<td>EE's Dept Location Index</td>
</tr>
<tr>
<td>EE</td>
<td></td>
<td>telecom</td>
</tr>
</tbody>
</table>
details about the database servers, in the rest of the cases it would contain the pointers to the next level of brokers. There will also be a provision for a default next-level broker for every broker. So this contains the entries of type <Topic, Actual-database-server> for which this broker is the actual broker, <some-topics, some-other-broker> for some topics and finally the catch-all <default-topics, default-broker>.

The brokers in the system may have one or many such indices but all these brokers must have at least the department location index or the pointer to the broker which keeps such an index.

3.3 Database Server

The database server interacts with the local database management system which keeps the library catalogues. Each server holds library database for one or more search items. The information about what type of search items are supported is known to the brokers. When client contacts the server with a specific query, server accepts it, verifies that it is indeed capable of resolving that query. Then it converts the common query format into the local schemas for local database management system and contacts the local database manager with this query. It gets back the results of the query and hands it over to the client.

The server constitutes of following three basic units.

- The Communication Unit is as usual responsible for listening for the queries from all clients in the system, getting the results from the Query Handling Unit and sending them over to the client.

- The Processing Unit of the server accepts the query to be processed and translates it from the common format into the format which is acceptable to the local database management system. It then hands over the query to the Query Handler which arranges to contact the local database management system.

- The Query Handler gets the query from the Processing Unit in a format acceptable to the local database management unit, contacts the local database manager with the query. It waits till it gets the result from the local DBMS. Once it gets the results, it passes them on to the Processing Unit for further processing as translation etc.

The server can be contacted by many clients at any given time so it has to be of non-blocking nature, i.e. server can accept queries from another client without waiting for the previous query to get over.

4 Updating Brokers

This system is meant for the retrieval of the library items in a distributed environment. It is not essential for such a system to incorporate changes in the indices on the brokers as soon as the database on the servers get changed. It is enough to assume that the changes will be reflected on the brokers eventually.

There are two approaches to solve this problem. The one simple approach is to just broadcast the changes to all brokers. When database managers add/delete the library items in the database, the database server just broadcasts these updates and the brokers upon receiving these changes update their own indices (add/remove entries in the tables). Another approach is to keep the list of brokers which have entries for this server on each server and send updates to only those brokers in the list. Each such broker will eventually ensure that all other brokers' tables are updated. This is possible because of the organization of brokers and the entries for other brokers in the indices of each broker.

To avoid cycles and endless updation loops, we use hop-count field and modify that on the way.

5 The Queries Supported

The proposed system is the information retrieval system for the library catalogues, mainly for books, journals and other such library items. So at least the bare minimum support for searching and querying the library items, like search by Author, Subject, Title etc should be provided at each client site.

5.1 Client Customization

This system offers queries which can have the wildcards in some of the query-fields. For example, the user can ask some query which has value of the field called Dept='Any-dept' and it should be possible to substitute the values for that field. This means that the client has to contact all the libraries of any one of the available departments in the institute. It would be logical as well as efficient if client software allows user to specify some choices which can be used as the values for such wild-card query-fields instead of selecting the random values or all possible values. It is also quite natural for example to first search the
user's own choice of departments before other departments in the institute. This option can very well be masked off by setting Dept-Choices? field to FALSE.

Users can customize the user interface, the choice-list of libraries, departments, subjects etc. Whenever there is any query which involves the fields with the values say any one or all for libraries, depts, subjects and if the user has asked the Processing Unit to refer to the local choices, the Processing Unit looks at the customisation and select the fields of the query from the choices specified.

5.2 The Format of the Queries and Responses

As discussed earlier, we offer only a certain types of queries in the system. This is not really a limitation, as this system is aimed to be used for the library search. And moreover we can always support the new types of queries by suitably modifying the format. Broadly we can specify the format of any query in the system as follows,

\[
\text{Query} = \{ \\
\quad \text{Client-Id-specific-field;} \\
\quad \text{Hop-Count; Query-type; } \\
\quad \text{Class-of-Query; } /\!* \text{Meta or standard } */ \\
\quad \text{Search-Items } \{ \\
\quad \quad \text{Dept;Subject;Lib-Id} \\
\quad \quad \text{Use-Dept-Choices?; } \\
\quad \quad \text{Use-Subject-Choices?; } \\
\quad \quad \text{Use-Library-Choices?; } \\
\quad \quad \text{Author;Title; } \\
\quad \} \\
\}
\]

Here the fields Dept, Subject, Library-Id, Author and Title can take one of the three values; either "*" (stands for all), "?" (stands for any one) or it can have the "actual" value. The fields Use-Subject-Choices?, Use-Dept-Choices?, Use-Library-Choices? have boolean values and are used to instruct the Processing Unit whether or not to consult local customisation for filling up the respective fields. Dept, Subject and Library-Id fields can have fields joined with the boolean operators, like AND and OR. The field Hop-Count is used to avoid the cycles and indefinite bouncing of queries between the brokers due to faulty index organisation.

The responses to the queries are either from the brokers or from the database servers. Brokers either respond with the full answers to the meta-query or with the pointer/s to another broker/s. Servers return only full answers as a response to clients. By examining the fields in the responses, client can find out whether the response is the answer to the query or the pointer to another broker. In some cases the response can very well be a mix of the answer and the pointers to other brokers. Responses have the following format,

\[
\text{Response} = \{ \\
\quad \text{Broker-or-server-Id; } \\
\quad \text{Hop-Count; } \\
\quad \text{Answer-or-pointer-or-both; } \\
\quad \text{Response-for-this-type-of-query; } \\
\quad \ldots \\
\}
\]

References


