Server Selection for Search/Retrieval in Distributed Library Systems

Miroslav Zarić*, Branko Milosavljević*, Dušan Surla**
* University of Novi Sad/Faculty of Technical Sciences, Novi Sad, Serbia
** University of Novi Sad/Faculty of Science, Novi Sad, Serbia
{miroslavzaric, mbranko, surla}@uns.ac.rs

***Abstract*—This paper presents one approach to solve server selection problem during search in distributed library systems.**

Information retrieval systems are aimed at providing infrastructure and means of finding specific documents in a collection, such that they satisfy specific information need. Distributed information retrieval systems aims at providing the same capabilities, but in an environment of distributed and heterogeneous document repositories. In such an information retrieval system, an important step is server selection i.e. selection of servers that will be included in a search operation, and to whom the query will be sent. Other problems that are specific to distributed information retrieval systems are query formatting for different servers, and results merging and ranking. These are special class of problems, and are not the subject of this paper. One of the first institutions to massively adopt information retrieval systems were libraries. Currently, almost every library has an online search capability. Using these search capabilities, a client application can perform search across a network of library servers. This paper is focused on a method for server selection in such search scenario.

I. INTRODUCTION

Information retrieval systems are nowadays regularly used for almost every search we perform on the Internet, and in various business information systems. An information retrieval system needs to provide efficient processing of large collections of documents, an efficient search algorithm, and result ranking. Each information retrieval system implements some specific document representation model and search model [1].

Distributed information systems, essentially represents a group of computer hardware components working together to fulfill a desired goal. In distributed information systems, typically, no hardware resources are shared, and each computer is managing its own resources, while cooperation is achieved on a logical level by entangling software components running on different computers (nodes). Distributed information systems are meant to be transparent for users, i.e. user should not be aware whether his request is handled by local computer, or by distributed system.

Distributed information retrieval systems are focused on providing information retrieval capabilities over a vast network of document servers – servers that holds document collections, as well as some tools for access, search and retrieval of documents in that collection – practically an implementation of a standalone information retrieval system.

But existence of large number of document servers presents new challenges. To successfully perform search over a network of available document servers, user should know, at least, access URL to every relevant document collection, and furthermore the details of its operations (such as query language implemented). Distributed information systems provides a new component, usually called *search broker*, that is meant to work as an interface point between users and remote document servers. This system works as a specialized client module, receiving a user query in one notation, transforming it into appropriate language for each document server, and collecting results from them, and representing it to the user.

It is evident that any distributed information retrieval system needs to solve at least following problems:

- Document server representation and server selection during queries
- Result retrieval, and duplicate removal
- Consolidated ranking

Each of these tasks presents a distinct research area.

More on distributed information retrievals methods can be found in [2].

This paper concentrates on the first problem, specifically in an environment of library servers. Libraries have traditionally been one of the first adopters of information retrieval systems. Although the task of search and retrieval is similar, library information systems have some specifics:

Instead of holding full text documents library information systems usually contain library records, most recently in MARC 21 [3]. These records contain all relevant data about some library item (such as bibliographic data, location data). Searching in library information systems is performed over a collection of these records. These records have well defined structure that enables some more guided search. In modern days, and with the advent of digital libraries, these distinctions are blurred, since they contain library records as well as a full-text, electronic versions of documents.

Although there are different implementations, most library information systems use standard Z39.50 protocol [4] for search and retrieval. But its use does not guarantee general compatibility, the problem that will be discussed later. In a recent period an approach to adapt this commonly use protocol to new, internet environment have given rise to SWR/SRU protocol, also implemented by increasing number of libraries.

But even in such an environment, of library servers, where good and broadly adopted standards exists, there is

a need to perform the same, previously defined, steps as in any other distributed information retrieval system, although some problems will be less intensive (usage of common query language and communication protocol simplifies communication from query broker to specific server).

II. Server Selection in Distributed Information Retrieval Systems

Most server selection methods operate as methods for ranking information gathered about the servers. Usually distributed systems adopt one of the methodologies for document ranking, and adopt it to ranking servers. There is, obviously, a need to define how information about the servers is represented, and to adopt chosen document ranking model to that representation.

In an analogy to document ranking, here we perform server ranking for every performed query. After an initial ranking is established, further searches, based on the query criteria are routed only to those servers that are expected to return valuable result for the query. Usually, the search query is passed to *n* best ranked servers in the list. This approach can greatly improve performance of overall system, lower the network traffic, as well as other resource requirements, while producing the same or equivalent amount of relevant results.

Best known algorithms for server selections are CORI [5], CVV [6], bGlOSS, vGlOSS [7], and weak sampling techniques [8].

CORI system was one of the first systems to introduce server selection. The server selection is based on adaptation of well known *tf-idf* norm for ranking documents. In this case a document frequency *df* and inverse collection frequency *icf* are used to rank the collections.

bGlOSS, and vGlOSS were introduced as models for ranking servers implementing Boolean and vector based retrieval model, respectively. Document frequency and collection size are used as a basis for calculating server ranking. Since the exact collection size is usually unavailable, some estimation methods are used to evaluate the size of collection.

CVV (Cue Validity Variance) evaluates query terms, in such a manner that terms that are better discriminatory values for servers gain a higher weight. Therefore, using weight of such term the importance of the server is pondered for query containing the term.

With weak sampling method, for any complex query, a short two-term sample query is sent to the server. Results of this sample queries are used to calculate server rank. This method assumes that server will provide additional data, alongside result, to allow for runtime rank calculation.

Server ranking can be enacted in:

a) cooperative environment (when each queried server response contains not only relevant documents, but also additional information, such as document ranking, total number of hits – information that can be readily use for server ranking)
b) non-cooperative environment, when only information available at run time is list of results obtained from a specific server.

There are also some server selection methods that are based on *query clustering* or *relevant document distribution* model. These methods are trying not to rank the servers itself, but to predict the number of relevant documents on each server.

Server selection problem is not only inherent to information retrieval system, but exists in other commonly used distributed systems, such as P2P networks, IPTV networks etc.

III. Performing Search In Distributed Library Systems

Search is a feature most commonly used in library information systems. Most of the library information systems provide an online access point through which the search of its catalogue can be performed. Library systems traditionally contain only library records used to describe and locate holdings in the library. In a classic library, user will perform search and get information whether some item exists or not in a chosen library. Some library systems are enhanced to support item reservations through some online tools. Digital libraries allow users not only to gather information about specific item, but in some cases to download electronic version of document.

Generally there are two distinct types of users using library information system search capabilities. Ordinary users, using the search to locate if an item exist and whether it is available. Library staffs use the search capabilities to perform various catalogue-maintenance related tasks. One of these tasks is very important for overall performance of library information system – cataloguing task. It is important since it affects search capabilities for other users.

As the main intentions of these users are different, so are the query they perform. While an ordinary user, searching for a book, may be well served by searching the local catalogue (after all, the user will primarily be interested to find an item in a local library), librarian, performing cataloguing duty, will not be served well at all if the search is confined to local catalogue only. During the cataloguing process, librarian already has a copy of an item (a book for example) and knows that it is a new entry to the catalogue – that needs to be properly described by an associated MARC record. In order to reduce amount of time needed to populate MARC records, it is highly beneficial if the librarian can get a hold of an existing MARC record describing the same item, presumably from another library system. In this case local search will yield no relevant result, and search over a network of library catalogues should be performed. Such an operation – of using existing library record to amend it and incorporate in local catalogue is called copy-cataloguing. Apart from time saving, this approach has additional benefit of increasing the overall completeness of records. The quality of the MARC records is a debatable issue, discussed in many papers, and also part of the research conducted in [9].

Since there are many library servers available, librarians will tend to use those server that are providing the most complete records, and most librarians will in

time develop their preferred list of target servers from which to retrieve records. However, initially, and later in some occasions when rare or new book is catalogued, there is a need to perform search over larger number of library servers in order to obtain some records.

Z39.50 protocol is used as a standard for communication between library information systems. It provides facilities to initialize communication link, perform search, and for presenting results to the client. The protocol allows for use of different query languages, with query language Type 1 as mandatory. As an alternative, newer systems also support use of SRU protocol [10] which uses CQL as a query language. This protocol is also standardized under the guidance of Library of Congress.

The system presented in this paper is part of continuous development of BISIS library system. As an integral part of the system, a client application for search and retrieval has been developed [11]. It allowed users to perform search to one remote library system at a time using Z39.50. In later upgrading, described in this client application was enhanced to allow simultaneous use of both Z39.50 and SRU protocols, and to allow parallel querying of multiple servers. This change required for query adaptation from Type1 query for Z39.50 compatible servers to CQL for SRU servers. This module is described in details in [12].

IV. Server Representation And Ranking

Introduction of support for simultaneous queries to different servers has introduced several problems that, although foreseen, need to be addressed. One of the most important is different level of support from different servers regarding different attributes (for example, some servers may support query by ISBN or ISSN, others do not). We will concentrate on Z39.50 servers since they represent a vast majority of servers available for querying. If we do not know which *use* attribute is available on which server, sending the same, complex query containing different attributes, to multiple servers usually produce large quantity of erroneous connections, i.e. the response from server was an error message stating that server is not capable of processing request on some attributes. In order to obtain information which server is supporting which attribute we have several choices:

- Incremental gathering of server description data. Initially we send all simple (one attribute) queries indiscriminately to all servers. If the server responded without an error – that attribute is supported on the server.
- Using Explain service of Z39.50 as long as servers are implementing it.
- Use an existing Z39.50 target directory, to get information about server capabilities.

	Explain Category	#Targets supporting
1.	TargetInfo	173 (8.43%)
2.	DatabaseInfo	173 (8.43%)
3.	CategoryList	172 (8.38%)
4.	AttributeDetails	169 (8.23%)
5.	AttributeSetInfo	134 (6.53%)
6.	RecordSyntaxInfo	120 (5.84%)
7.	SchemaInfo	34 (1.65%)
8.	TagSetInfo	34 (1.65%)
9.	Processing	26 (1.26%)
10.	TermListInfo	2 (0.09%)

Table 1. Support for Explain facility

First option is simple enough, but requires large number of queries to multiple servers before we can produce knowledge base for further selection.

Second option allows for automatic reconfiguration of client application to fit with capabilities of target server. That would enable that some attributes are automatically disabled if target server is incapable of processing it. Although *Explain* service of Z39.50 is exclusively intended to transmit information about server capabilities to the client side, the downside is that it is not required. So, a number of servers simply does not implement *Explain* service. Table 1 gives an example of support for *Explain* feature among 2052 servers listed in *The Z39.50 Target Directory (http://irspy.indexdata.com/)*

Third option – existing directory of available Z39.50 targets, is good starting point since they represent an aggregated list of server descriptions, with different level of details. There are several public directories of library servers available, such as:

- The Z39.50 Target Directory
  http://irspy.indexdata.com/
- LOC – Gateway to Library Catalogs
  http://www.loc.gov/z3950/
- MUN Libraries Library Z39.50 destinations - Queen Elizabeth II Library Memorial University of Newfoundland
  http://staff.library.mun.ca/staff/toolbox/z3950hosts.htm
- Directory of Z39.50 Targets in Australia
  http://www.nla.gov.au/apps/libraries?action=ListTargets
- Z39.50 Hosts Available for Testing
  http://www.loc.gov/z3950/agency/resources/testport.html

For the development of the system presented in this paper, the first directory has been used, since it provided an XML format for server descriptions (the same format that would be provided by *Explain* facility), and has more than 2000 servers listed. The list is regularly updated. Additionally, alongside basic server description data, this directory also has a *host connection reliability* measure. This measure is calculated as a percentage of successful connections to target servers in last three months.

Previous search sessions can be a source of valuable data about each server performance. Even erroneous connections provide valuable input that can be used to calculate usability of the server for future searches. Paper [13] takes into account network parameters for accessing each server.

Analysis of previous search sessions can provide following data:

- Total number of queries in which server connection was invoked
- Total count of successful connections
- Total number of errors
- Mean query response time of the server

- Total number of results returned
- Total number of results from server that has been selected for further usage by the user

We can assume that N queries have been submitted in total, and that M servers are available.

For each server i one can record the total number of invoked communication sessions n_i. Initially, without prior knowledge, we can assume that all queries will be sent to all servers. As number of search queries grows, librarian will tend to restrict server list to those server that provided valuable information in prior sessions. Hence, the total number of requested communications can be used as an indirect measure of importance one user gives to specified server, regarding his common queries. Even if server is automatically excluded from some search sessions, due to unsupported attributes, this notion of importance still holds its place, since selected query attributes also represent user's habits or preferences when forming the query. We can create a measure of importance given to this server by specific user $imp_i=n_i / N$.

Using the number of requested communications n_i and number of erroneous responses e_i from any server i the measure of overall relative reliability can be calculated as $rel_i=1-e_i / n_i$. As a starting reliability measure for servers, the one obtained from server directory list is used.

Since, without prior knowledge, we can not estimate the size of each collection, we can create an indirect relative measure based on total count of results returned from an i^{th} server (r_i), and cumulative total count of results from all servers R. Calculated value $r_{set,i}=r_i / R$ represents relative contribution of given server to total result set.

If we want to take into account a response time of the server, we can measure total time $t_{uk,i}$ it took the server i to complete n_i search queries. Mean response time of the server in that case can be calculated as $t_{sr,i}=t_{uk,i}/n_i$. To put this measure into relation with other servers, we can compare it to cumulative mean response time calculated as $T_{sr} = \frac{\sum_{i=1}^{M} t_{uk,i}}{\sum_{i=1}^{M} n_i}$.

Relative response time of a server (server speed), compared to group of servers, can now be computed as $t_{rel,i}=t_{sr,i} / T_{sr}$. We can further normalize this value to bring it to the range [0,1]. For this purpose, a sigmoid function can be used.

Finally we can take into account the information about number of records that has been copied into local system (retrieved records for copy cataloguing). If the total number of retrieved records is R_r, and number of records retrieved from server i is $r_{r,i}$ than relative contribution of server i to total set of retrieved records is $r_{ret,i}=r_{r,i}/R_r$. This measure is also an indirect measure of "quality" of records from that server, from a user point of view (user will tend to pick those records that are similar to its cataloguing needs).

Based on these measures each server i can be represented with a performance vector

$si=\{imp_i, rel_i, r_{set,i}, t_{rel,i}, r_{set,i}\}$.

We can create a vector of "maximum performing" server, and compare all other servers to it. Standard measure of cosine similarity can be used, or any other method used in vector based models. As more and more search queries is performed, best performing servers will be ranked better.

V. Implementation And Testing

The proposed system is implemented in a client part of BISIS application. An XML configuration file is used to form the list of available servers. This XML file contains all relevant server data, such as server name, URL, port, supported access points (attributes that can be used for query terms matching). This XML file also contains data representing all information gathered about the server from all previous search/retrieve sessions. Table 2 displays servers support for use of different attributes in search queries. Since server list provides information about supported attributes, the client module has been altered so that servers are now automatically excluded from available server list if a not supported attribute has been selected for the given query. This feature represents a server list filtering based on a query formulation.

To test the effect of server filtering, series of 50 queries on different attributes (and combinations) has been run, without server filtering and with server filtering. The list of 120 servers has been compiled from a full list of 2000+ available servers. The same servers were used in both run. The results are given in Table 3. Although server capabilities are taken into account, it did not completely removed errors. The actual cause of errors may be different, and not restricted to supported attributes. It may be that server is temporarily unavailable, or that given address is no longer accessible. However since the main goal of the server filtering (and ranking based on proposed criteria) is to compile a list of best performing servers, for future use, the actual cause of remaining errors were not further investigated, but existence of these errors were taken into account when calculating each server performance vector.

Additionally, a low level change in communication module has been introduced. Original version of the communication module fires simultaneous connections to multiple servers. In altered version, new configurable property *maxActiveConnections* has been introduced.

#	Attribute	Name	# Targets
1	4	Title	1523 (73.18%)
2	21	Subject heading	1494 (71.79%)
3	1003	Author	1487 (71.45%)
4	7	ISBN	1316 (63.23%)
5	8	ISSN	1261 (60.59%)
6	5	Title series	1245 (59.82%)
7	1016	Any	1227 (58.96%)
8	1	Personal name	1196 (57.47%)
9	12	Local number	1175 (56.46%)
10	13	Dewey classification	1088 (52.28%)
11	2	Corporate name	1074 (51.6%)
12	31	Date of publication	1067 (51.27%)
13	3	Conference name	1048 (50.36%)
14	54	Code--language	994 (47.76%)
15	6	Title uniform	952 (45.74%)
16	1007	Identifier--standard	926 (44.49%)
17	33	Title key	922 (44.3%)
18	16	LC call number	896 (43.05%)
19	1004	Author-name personal	892 (42.86%)
20	9	LC card number	853 (40.98%)

Table 2. Server support for different access points (search by specific attribute)

use attribute	Errors (server filtering off)	Errors (server filtering on)
ISBN	46%	35%
Personal name	40%	24%
Title	33%	24%
Author	34%	21%
Subject heading	30%	20%
Author+Title	45%	30%

Table 3. Effect of a server filtering on number of erroneous server responses

In case when all 120 servers have been included in the search, the original version would consume following resources:

- Total number of threads: 131
- Peak memory utilization: 66132 KB
- Mean memory utilization: 58544 KB

The altered version, with *maxActiveConnections* set to 10 consumes following resources:

- Total number of threads: 26
- Peak memory utilization: 51896 KB
- Mean memory utilization: 51004 KB

The total time required to perform search and present a result to the user has not been noticeably changed. With this optimization in place a minor reduction in number of errors has been registered. On average there were 3 errors less. This suggests that minor number of errors were produced by too many communication threads running simultaneously.

After initial sets of 50 queries on given attributes have been run, the server statistic is already formed so ranking of servers could be taken into account. To test if ranking is presenting relevant servers at top of the list additional queries were run.

Different queries have been run. Most of the queries used ISBN attribute (since it is the most common query attribute used for searching the records during the copy-cataloguing). On average, if the query is run only on servers that have ranked as 100% relevant, we could get about 55% of all results returned by a full, non-filtered set of servers.

However if servers ranking as 80% relevant or higher are selected (in our case it was a total number of 39 out of 120 servers) we got, on average 92% of all results returned by non-restricted search). With servers ranked as 70% relevant or higher were used the same result set is returned as with non restricted search.

These results are promising but further analysis, on data gathered in real life usage scenarios, should be performed.

However these results show that number of communications may be significantly reduced if only best qualifying servers are used to submit the search, while result set will remain relevant to the query.

This notion was further strengthen by introducing the "quality" measurement of record. There is no prescription how to judge the record quality. Surely the completeness of the record must be taken into account, as well as its syntax correctness, but from a copy-cataloguing viewpoint the best record would be the one that requires minimal effort to bring it in concordance with local cataloguing practice. Therefore, not only overall completeness does matter, but also existence of certain fields and even the style used to enter some data. This problem has been further addressed in [9].

CONCLUSION

This paper presents one approach to solving server selection problem, a common step in performing search in any distributed information retrieval system. In this case it is implemented on client application for Z39.50 and SRU protocols, commonly used in library information systems. Tracking of different performance measures, and ranking based on these measures are proposed. Data gathered about the server capabilities and its performance during the previous search sessions are used to estimate its relevance for future searches, based on the attribute set used in the query. This approach indirectly gives an opportunity to tailor the server selection according to individual users preferences, since some of the measures are directly affected by the choices user has made on returned results. This enables the client application to be personalized to reflect the user's preferences.

Taking into account server capabilities, taken from server directory list, a filtering of the available servers can be performed, thus reducing the number of communication links that will certainly result in errors. Additionally, ability to set the number of active connections can reduce the resource usage. Server ranking can be used to limit the number of servers that needs to be queried, but still to be able to get result most relevant to the user. Furthermore, delayed start of communication threads gives an opportunity to stop the search if some predefined numbers of records are already retrieved from best ranking servers. This server ranking system is further strengthened when used in combination with record ranking algorithm.

REFERENCES

[1] Manning, Christopher D. and Raghavan, Prabhakar and Schtze, Hinrich, Introduction to Information Retrieval (New York, NY, USA: Cambridge University Press, 2008).

[2] Nicholas Eric Craswell, "Methods for Distributed Information Retrieval" (2000).

[3] Gordana Rudic and Dusan Surla, "Conversion of bibliographic records to MARC 21 format", The Electronic Library 27, 6 (2009), pp. 950-967.

[4] ANSI/NISO, "Information Retrieval (Z39.50): Application Service Definition and Protocol Specification", Library of Congress.

[5] Callan, James P. and Lu, Zhihong and Croft, W. Bruce, "Searching distributed collections with inference networks", in *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA: ACM, 1995), pp. 21--28.

[6] Gravano, Luis and García-Molina, Héctor and Tomasic, Anthony, "GlOSS: text-source discovery over the Internet", *ACM Trans. Database Syst.* 24 (1999), pp. 229--264.

[7] Budi Yuwono and Dik Lun Lee, "Server Ranking for Distributed Text Retrieval Systems on the Internet", in Rodney W. Topor and Katsumi Tanaka, ed., *DASFAA* vol. 6, (World Scientific, 1997), pp. 41-50.

[8] David Hawking and Paul B. Thistlewaite, "Methods for Information Server Selection", *ACM Trans. Inf. Syst.* 17, 1 (1999), pp. 40-76.

[9] Zarić, M. “Model za distribuirano i rangirano pretraživanje u bibliotečkim informacionim sistemima”, doctoral thesis, University of Novi Sad, Faculty of Technical Sciences, 2013 (serbian).

[10] Search Retrieval over URL, Standard, Library of Congress, available at: http://www.loc.gov/standards/sru/index.html

[11] Boberić, D., "System for retrieval of bibliographic records" (2010).

[12] Miroslav Zaric, Danijela Boberic Krsticev, Dušan Surla, "Multitarget/multiprotocol client application for search and retrieval of bibliographic records", *Electronic Library, The* Vol. 30 Iss: 3 (2012), pp. 351-366.

[13] Carter, Robert L. and Crovella, Mark E., "Server Selection Using Dynamic Path Characterization in Wide-Area Networks", in Proceedings of the INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution (Washington, DC, USA: IEEE Computer Society, 1997), pp. 1014.