RESEARCH ARTICLE OPEN ACCESS

Enhancement of Information Retrieval in Distributed Systems

R.Senthilkumar[1], Dr. M. Ramakrishnan [2]
Research Scholar[1], Professor & Head in IT [2]
Anna university[1], Velammal Engineering College[2]

Abstract:
The distributed system enables multiple, simultaneous connections between clients and Inquery servers. The different components of the system communicate using a local area network. Each component may reside on a different host and operates independently of the others. In this section, we describe the functionality and interaction between the clients, the connection server, and the Inquery servers. The clients are lightweight processes that provide a user interface to the retrieval system. Clients interact with the distributed IR system by connecting to the connection server. The clients initiate all work in the system, but they perform very little computation. The clients can issue the entire range of IR commands but, in this paper, we focus on inquery, document retrieval commands and query evaluation measurements. A client sends query commands to the connection server.
Keywords : Simulation model, Query retrieval measurements, Document retrieval measurements, Distributed retrieval.

I. Introduction

A query command consists of a set of words or phrases (terms). The command either specifies the list of Inquery servers to search or the client allows the connection server to determine the appropriate collections to search. Clients may also retrieve complete documents by sending a document retrieval command to the connection server. The command consists of a document identifier and collection identifier. In response, the connection server returns the complete text of the document from the appropriate Inquery server. A client issues a command and waits for the connection server to return the results before it issues another command. Users issue queries and document commands. The clients and Inquery servers communicate via the connection server. The connection server is a lightweight process that keeps track of all the Inquery servers, outstanding client requests, and organizes responses from Inquery servers. The connection server continuously polls for incoming messages from clients and Inquery servers. The connection server handles outstanding requests from multiple clients. A client sends a command to the connection server which forwards it to the appropriate Inquery servers. When the connection server receives an answer from an Inquery server, it forwards the next command on the corresponding queue to the Inquery server. The connection server maintains intermediate results for commands specifying multiple Inquery servers. When Inquery servers return results, the connection server merges them with other results. After all the Inquery servers involved in a command return results, the connection server sends a final result to the client. Only query and summary commands may specify multiple Inquery servers. For a query command, each Inquery server sends its top *n* responses back to the connection server. The connection server maintains a sorted list of the overall top *n* entries until all the Inquery servers respond. The connection server merges new results with the existing sorted list. The Inquery server uses the Inquery retrieval engine to provide IR services such as query evaluation and document retrieval. Inquery is a probabilistic retrieval model that is based upon a Bayesian inference network .Inquery accepts natural language or structured queries. Internally, the system stores text collections using an inverted file

II. Simulation Model

We present a simulation model for exploring distributed IR system architectures. Simulation techniques provide an effective and flexible platform for analyzing large and complex distributed systems. We can quickly change the system configuration, run experiments, and analyze results without making numerous changes to large amounts of code. Furthermore, simulation models allow us to easily create very 6 large systems and examine their performance in a controlled environment fig.1. Our simulation model is simple, yet contains enough details to accurately represent the important features of the system. We model the clients, Inquery servers, and connection servers as different processes. Processes simulate the activities of the real system by requesting services from resources. The simulator is driven by empirical timing measurements obtained from our prototype. Our technique for designing an environment for studying distributed information retrieval architectures [2]. A distributed object-oriented database system while our work focuses on IR systems

using simple command. To accurately model an IR system, we analyze the prototype of the distributed Inquery system.

III. Query Evaluation Measurements

The simulator uses a simple, yet accurate model to represent query evaluation time. Based upon our measurements on Inquery using our query sets, evaluation time is very strongly related to the number of terms per query and the frequency of each of the terms. Our query evaluation model is a function of the number of terms per query and the frequency of the individual query terms plus a small overhead where n is the number of terms in the query and term I is the term[2] ,[3]. We model eval term time as an increasing linear function of the term frequency.

IV. Document Retrieval Measurements

We measure Inquery to determine the amount of time it takes to retrieve a document. For our text collections, the retrieval time is variable and there is not a strong correlation between document size and retrieval time. The low correlation is due to the size of the documents in our text collections which are not very large so retrieval occurs very quickly. In our collections, the average size of a document in the TIPSTER 1, Congressional Record, and the CACM is 2.3 KB, 11.7 KB, and 0.5 KB, respectively. The simulator represents the document retrieval time for an Inquery server as a constant value, 0.31seconds, which is the average document retrieval time for 2000 randomly selected documents from the TIPSTER 1 collection. The connection server time consists of two values, the processing time for handling a message and the time to merge results. We obtain the message handling time by measuring the prototype connection server. When the connection server receives a message from either a client or Inquery server, the simulator uses a constant value, 0.1 CPU seconds, to represent the message processing time. The time to merge query results depends upon the number of answers an Inquery server returns.

We represent network time as sender overhead, receiver overhead, and network latency. The sender and receiver overhead is the CPU processing time for adding and removing a message from the network. The network latency is the amount of time the message spends on the network itself. These times depend upon the size of the message and the bandwidth of the network.

V. Distributed Retrieval

In a multi-server distributed text retrieval system, there are several independent mono-servers, or librarians. Each is responsible for some component of the collection, for which it maintains an index, evaluates queries, and fetches documents. Separate from the librarians are one or more receptionists, which interact with the users of the system and communicate user requests to the librarians. Each receptionist may be resident on the same physical machine as one or more librarians, or may be quite separate. In the models of computation we consider, the receptionists may have available global information about each librarian, such as the total number of documents or perhaps a partial (or even full) copy of its index information. A receptionist is essential to ranked query evaluation because it is necessary to collate the results of ranking each sub collection.

In this model, queries evaluation is as follows.

1. A user lodges a query with a receptionist or users [3], which examines any global information it has access to and passes the query, with perhaps some of the global information, to a selected set of librarians.
2. Each selected librarian evaluates the query and, making use of any transmitted global information and its own local information, determines a ranking for the local collection a list of document identifiers and similarity scores.
3. Each ranking is returned to the receptionist, which waits for all the nominated librarians to respond and then merges their rankings to obtain a global collection wide ranking and identify the top *k* documents. During the merging process the receptionist may again make use of global information.
4. Each selected librarian is given a list of document identifiers within its domain and is requested to return the text of the corresponding documents to the receptionist for display to the user. As an optional initial step, the receptionist may converse with the librarians to establish parameters. In this generic description of the model we have specified neither how the rankings are merged nor the internal structure of the librarians and receptionists.

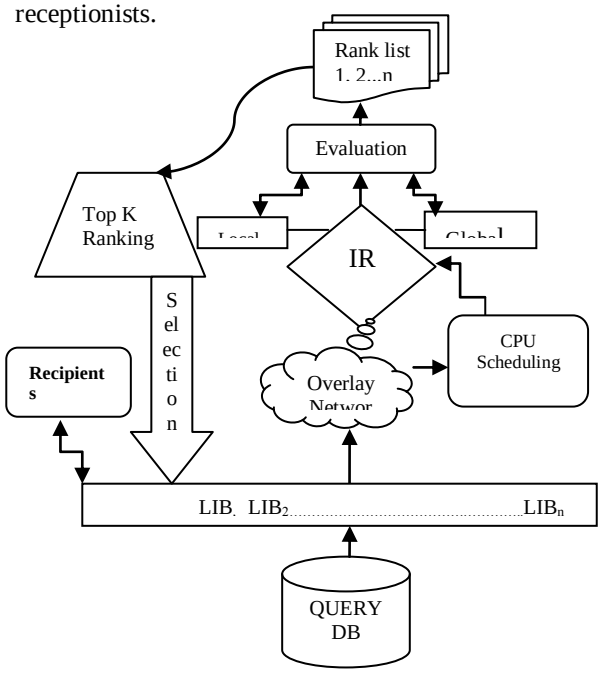

Fig1: Architecture

To evaluate performance we need to consider three factors. One is effectiveness. Another is response time the delay between issuing the query and return of answers, which depends on the amount of processing involved, the volume of network track, and the number of handshaking steps used. The last factor is use of resources CPU time at the receptionist and librarians, volume of data needed by librarians and transmitted over the network, and disk space required by the receptionist. Response time and resource usage are linked, but only loosely. In particular, response time measures the minimum delay a user will experience, even on a lightly loaded system, whereas resource use is an indication of the all queries throughput possible with the system when it is operating at capacity, with multiple users and queries competing for resources[4]. A key facility that we strived for was transparency; it should be possible for any set of collections to be queried as a single database. We require that each of the sub collections can be accessed without recourse to any central information and that any sub collection can be a logical component of databases managed by several divergent receptionists.

In Central Nothing (CN) system the only global information maintained by the receptionist is a list of librarians. When a query is entered every librarian is given the query and prepares a ranking of its *k* best" documents, as determined by its index and its values for parameters *ft* and *N*. When these lists have all been returned the receptionist merges them, accepting at face value all supplied similarity value sit has no basis for per turbing either the numeric values or the ordering. For *S* sub collections the result is a list of *k S* similarities. The top *k* are then extracted, and a document request list sent to each librarian. Some of the librarians may not be required in this second phase. The main advantage of CN operation is that no global information is required; the receptionist is free to choose any subset of librarians. The disadvantage is that much of the power of a ranked query is potentially lost. For example, a term might be common in one sub collection and be assigned a minimal weight, but in the context of the collection as a whole that term might be rare, and documents from the sub collection important thus the ranking will be poor. It might also be that effectiveness is dramatically compromised by the use of sub collection weights. Finally, it is possible that unnecessary calculation is performed the receptionist has no basis for excluding any sub collection, and so every sub collection processes the query in full. In principle the receptionist could pass the query terms to the librarians and the librarians then return *k* documents immediately, without the intermediate step of passing back document identifiers and similarities, much as for Boolean queries. Transmission of *k S* rather than *k* documents would severely degrade performance.

In a Central Vocabulary (or CV)[1] system the global information stored by the receptionist is the vocabularies of the sub collections, which allows the receptionist to determine for each term a collection-wide weight. This should allow better ranking, but the preprocessing stage eliminates the spur-of the-moment choice of sub collections possible in a CN scheme, and storage is required for the collection-wide vocabulary. Query processing is similar to that in a CN system, with the crucial difference that each query term transmitted to the librarians is accompanied by a weight to be used. In our implementation, the librarians still calculate a *k*-ranking, but the similarity scores computed by the various librarians are exactly the same as for the mono-server alternative. The formation of a global vocabulary means that collections can be completely avoided if they contain none or few of the query terms.

In a Central Index (CI) system, the receptionist or users has full access to the indexes of the sub collections, so it can perform all the index processing and request from each librarian the documents required to make a global ranking of length *k*. In this case the preprocessing involves merging the sub collection vocabularies and indexes, and the need for storage space on the part of the receptionist is relatively large. To save some of the central index space the receptionist can collect adjacent documents into groups and then index the groups as if they were single documents space is saved because the number of groups containing each term is less than the number of documents, reducing index size. Compared with a CV system[8], the advantage is that each librarian must consult only a fraction of its index. The potential disadvantage is that highly relevant documents that are grouped with non-relevant documents may not be retrieved. The performance questions we sought to answer in our full implementation were the size of the central index, the cost of processing the central index, the extent to which the librarians could be protected from redundant computation, and how overall costs compare to other approaches.

VI. Conclusions

We have discussed three alternative methodologies for practical distributed information retrieval, each based on a common architecture in which sub collections are managed independently by librarians and queries are brokered to librarians by Users. The methodologies are differentiated by the kind of data that must be held by the receptionist, varying from no more than a list of valid sub collections (central nothing) to a merged vocabulary (central vocabulary) to a full index of stored data (central index).

References

[1]. Bailey and Hawking, A parallel architecture for query processing over a terabyte of text. Technical Report TR-CS-The Australian National University.

[2]. Bell. C, A. Moat, I.H. Witten, and J. Zobel. The MG retrieval system: compressing for

space and speed. Communications of the ACM, April 2005.

[3]. Brumfield, J. A., Miller, J. L., and Chou, H.-T. Performance modeling of distributed object-oriented database systems

[4]. B. Cahoon and K.S. McKinley. Performance evaluation of a distributed architecture for information retrieval. In H.-Development in Information Retrieval, Zurich, Switzerland.

[5]. Callan J.P. ., Z. Lu, and W.B. Croft. Searching distributed collections with inference networks. In E.A. Fox, P. Ingwerson, and R. Fidel, editors, Proc. ACM-SIGIR International Conference on Research and Development in Information Retrieval, Seattle, Washington.

[6] N. Craswell, "Methods for Distributed Information Retrieval," Intl. Conf. (INACT) Proce..

[7]. Harman, D.W. The First Text REtrieval Conf. Information Processing and Management, July-Aug. 1999.

[8] D. Hawking and P. Thistlewaite, "Methods for Information Server Selection," ACM Trans. Information Systems (TOIS),

[9] R.M. Losee and L.A.H. Paris, "Measuring Search Engine Quality and Query Difficulty: Ranking with Target and Freestyle," J. Am. Soc. for Information Science, , 1999.