EXPERIMENTS ABOUT PERFORMANCE AND IMBALANCE IN DISTRIBUTED INFORMATION RETRIEVAL SYSTEMS

Fidel Cacheda Seijo, Diego Fernández Iglesias and Rafael López García
Department of Information and Communication Technologies, University of A Coruña
Facultad de Informática, Campus de Elviña s/n, 15071 A Coruña, Spain

ABSTRACT

There are several articles which suggest certain improvements in the performance of Web searches by means of techniques such as index distribution. Other articles analyze which are the bottlenecks of this performance (imbalance among query servers, network load when results are being sent, reordering of these results at the broker, etc.). Nevertheless, most of these works have only made use of simulation methods in order to perform the experiments, but the results have not been checked in a more realistic environment.
In the current article we submit an example of an information retrieval distributed system and we show the results of some experiments performed by means of the aforementioned system in order to check that it can be used to find out if, in practice, the techniques analyzed in several studies improve the performance in Web search and how certain factors (broker, the slowest query server, etc.) have an influence on the system.

KEYWORDS

Distributed information systems, information retrieval.

1. INTRODUCTION

Nowadays users expect search systems to retrieve the requested information in an almost immediate way. However, if that systems are centralized, they have to index bigger and bigger collections and, in general, they have to support more and more queries upon those collections in a simultaneous way. These two points make the aforementioned systems not to be scalable and to overload [11].

As a solution to these problems, we chose distributed search systems due to the fact that their resources can be shared, they are more concurrent and scalable, the load received by each node is smaller than in centralized ones, etc. In this sort of systems the functionality is split in two roles: brokers and query servers (QSs). Brokers are in charge of receiving queries from the users and distribute them among the different QSs, whereas QSs take charge of processing the queries and sending their partial results to brokers. Finally, brokers also combine and reorder those results in order to generate a final result which will be returned to the user.

Index files can be distributed among different QSs using two classical strategies: the first one is called *term partitioning* or global inverted files, and the second one is known as *document partitioning* or local inverted files [17][18]. The second strategy is more often used in the commercial solutions, since it is very easy to distribute the document collections uniformly among all the QSs and to generate local indexes there. When processing queries, each QS receives all the terms and it selects the resulting file set, which is disjoint regarding the other sets that have been generated by the remaining QSs.

The previous articles by Cacheda et al. [7][6][8] identified two bottlenecks in distributed systems with local inverted files: the broker and the communications network. This work is based on simulations where an ideal environment is assumed.

However, Badue et al. [4] conclude that if the environment is not ideal (and in practice often it is not), the response time of each QS can differ so much among them, although the QSs spread the files in an homogeneous way. This variation is called imbalance in the performance of the QSs.

The aim of this article is to show the main features of our system as well as to show by means of a little experiment that it is possible to use it to check that the different times taken down in the tests performed via simulation by Cacheda et al. remain in a realistic environment. Likewise, the system is going to be used to corroborate the existence of imbalance in the time spent by each QS to solve the queries, even though the files have been distributed uniformly among the QSs and these ones share the same technical features.

This article is organized in the following way: in Section 2 we mention the most important works about the subject; in Section 3 we describe the environment used for the experiments; in Section 4 we explain the experiments performed in our real environment and in Section 5 we discuss the results we have obtained. Finally, in Section 6, we show our conclusions and the ideas which can give rise to future jobs.

2. RELATED WORK

Our interest in Information Retrieval (IR) area focuses on distributed systems, so we now submit the most important works about this subject.

First of all, the scalability of this sort of systems have been studied by means of simulation by Cahoon and McKinley [9], getting some encouraging conclusions in order to a possible practical demonstration.

As regards the index distribution strategies, it is possible to find a comparative analysis in works like the ones by Tomasic and García-Molina [20] and even in more recent ones like MacFarlane et al. [14] and Badue et al. [5]. On the other hand, Moffat et al. submit in a more recent article [15] some techniques to improve the performance of distributed systems based on term partitioning. The work by B. Ribeiro-Neto and R. Barbosa [17] evaluates these strategies, but it also shows other parameters which affect the query resolution on distributed systems.

As we have previously mentioned, several articles by Cacheda et al. [7] [6] [8] identify two bottlenecks on this sort of systems: the broker and the communications network. Their works also propose a set of solutions to improve the performance of both matters, such as the reduction of partial results or the distributed brokers.

Besides, in Badue et al. [4] there is an analysis of the imbalance which can be found in distributed IR systems to prove that in real environments it also affects the performance. The three basic factors which cause this imbalance are the number of QSs, their amount of memory and the use of disk cache. This imbalance forces the system to decrease its speed in accordance with the slowest QS.

Finally, Lin and Zhou [13] and Hawking [10] are some of the first researchers who have published works based on experiments in non-simulated environments.

The main differences among our article and the previously mentioned ones are: the experiments we describe have been carried out in a test environment with real hardware, we use public data and query collections (the query collection is also free), and we pay a special attention to some bottlenecks identified in the works by Cacheda et al. [7] [6] [8] (broker and communications network) and Badue et al. [4] (imbalance).

3. TEST ENVIRONMENT

3.1 The Information Retrieval System

The distributed IR system we have designed and implemented in order to perform the experiments is an extension of the Terrier search engine [18], which has been developed in the University of Glasgow [22]. Among the most important features of Terrier we have to mention that it is OpenSource, it was written in Java, it supports a conventional query language, it has a large number of document weighting models and it can work with the most common file formats and with data collections which have been especially created for research, like the ones of TREC Terabyte Track [21].

At the same time as it is essential to maintain most of the features of Terrier, it is also important to fulfil two objectives in the design which extends the system: to modify the lowest part of the original source code and to make easy the addition of new functional qualities to the distributed environment.

In order to fulfil the first objective, we chose to use an index distribution in accordance with the local inverted files strategy, since the global inverted files strategy would force us to modify the index structure of Terrier. The only modification that the original code of Terrier has suffered was the necessary in order to measure the performance.

The second objective has been fulfilled by means of a layered design (figures 1 and 2). On each layer we can find different implementations that solve the same problem, and they only differ in several features. A factory takes charge of instantiating the most appropriated implementation, so the user can choose it simply changing a configuration parameter. As the most remarkable example, we will point out that it is possible to choose the transport protocol used by the distributed system, so the user chooses between TCP and UDP, and in the second case it is possible to take advantage of Multicast or to refuse this possible advantage.

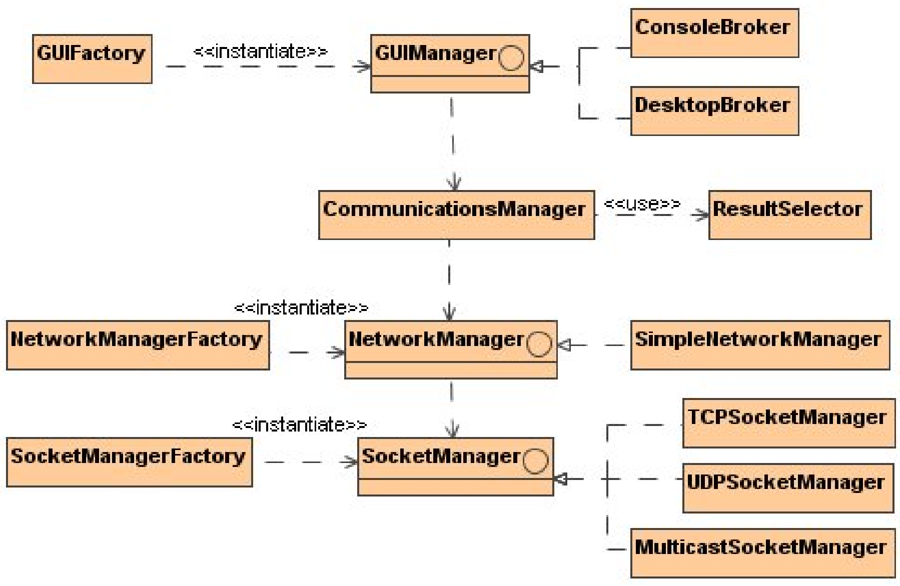

Figure 1. Broker architecture.

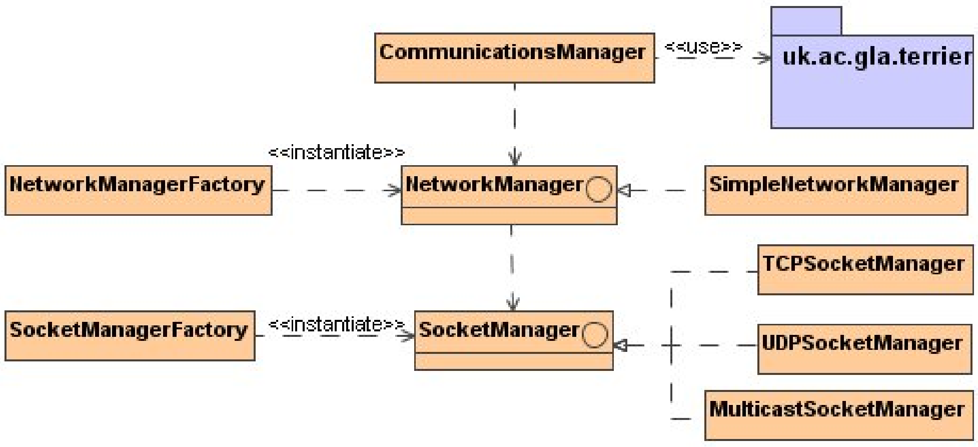

Figure 2. QSs architecture.

Figures 1 and 2 show that both parts of the system share 3 layers. The first is *CommunicationsManager,* responsible for the main logic of the application. The second is *NetworkManager*, conceived for possible network operations, like packet partitioning in the application layer, although there is only a trivial implementation at the moment. Finally, *SocketManager*, is in charge of the packet transference by sockets. The broker also has a graphics user interface.

The system allows the user to establish a broker hierarchy too. In order to do it, the user simply has to fill a configuration parameter in each node to indicate which its parent broker is. Nevertheless, it is necessary to indicate at this point that the experiments we have performed do not exploit this feature, so the different QSs are connected to the same broker. However, we hope to use this feature in future works.

3.2 Document and Query Collections

The studies by Cacheda et al. [7] [6] [8] are based on a simulation of the SPIRIT collection (94,552,870 documents and 1 TB of text) [12]. On the other hand, the study performed by Badue et al. [4] uses a WBR03 document collection, composed of 10 million of pages, collected from the Brazilian web by the TodoBR search engine [19] in 2003. The query set is composed of 100,000 queries extracted form a partial log submitted to the TodoBR search engine in September 2003.

In the case which concerns to the present article, the document collection which we have used in our experiments is .GOV [1], which is part of the TREC Terabyte Track [21], it is composed of 1,247,753 documents which belong to .gov domains and it is distributed by the University of Glasgow [22]. These documents were collected at the beginning of 2002 using the hardware and the network of the NIST [16].

As regards the queries, we have used the 2006 Efficiency Task Topics collection of TREC 2006 Terabyte Track [3]. This collection is composed of 100,000 queries but we have only considered an existing subset of 10,000 queries.

3.3 Hardware Environment

The experiments have been executed in a 100 Mbps Fast Ethernet communications network.

The broker has been installed in a PC with a 3.2 GHz Intel Pentium 4 processor with HyperThreading support, 1GB of RAM and Linux Ubuntu operating system.

The machines which support the QSs have the following technical features: 2.6 GHz Intel Pentium 4 processor with HyperThreading support, 512 MB of RAM and Linux Ubuntu operating system.

4. EXPERIMENTS

The experiments consist of executing each one of the 10,000 queries chosen from the 2006 Efficiency Task Topics collection [3] and taking down several times at the broker and at each one of the QSs.

For each query, we assume that the distributed system not only retrieves the *n* best results with the same precision as its centralized counterpart, but in the worst case, each QS also retrieves the *n* best partial results. These results are organized in the broker so it is guaranteed that finally the *n* best results are going to be offered to the user.

In these experiments we are mainly interested in network times and in imbalance.

All this procedure will be repeated for a variable number of QSs (1, 2 and 4, respectively) and for TCP and UDP with or without Multicast.

5. RESULTS

Herewith, we show some tables containing the results of the execution of the queries with the different configurations of the system. In these tables, the symbols have the following meanings: μ is the average, σ is the standard deviation, *p10* is the percentile 10, *p90* is the percentile 90, μp is the average using percentiles, σp is the standard deviation using percentiles and, finally, *I* is the imbalance. Times are shown in milliseconds.

For each configuration, we have calculated the 10 and 90 percentiles in order to remove disproportionate results, due to JVM aspects such as on demand class loading, that may bias the evaluation of the obtained data. Therefore, a little oscillation between averages considering or not percentiles can be noticed.

Table 1. Times for 1 QS.

	TCP			UDP			Multicast		
	QS1	Broker	Network	QS1	Broker	Network	QS1	Broker	Network
μ	278.64	278.95	0.31	277.2	277.48	0.28	281.62	281.87	0.25
σ	179.48	179.49	0.61	179.27	179.28	0.57	183.52	183.53	0.42
p10	83.25	83.54	0.26	82.09	82.34	0.23	82.19	82.41	0.21
p90	527.12	527.48	0.34	525.26	525.59	0.31	535.66	536.08	0.28
μp	258.51	258.82	0.29	256.92	257.2	0.26	260.87	261.12	0.24
σp	118.2	118.2	0.02	118.13	118.13	0.02	121.27	121.27	0.01

Table 1 shows that the averages considering percentiles are greater than the averages not considering them, being the reason that most of results that have been removed were quite bigger than averages. It can be already watched in this first table that the times taken at the broker are not pure processing times, but they represent the total amount of time spent in solving the queries. They are very close to the times of the QS1 so we can deduce that the network time is quite small.

It's remarkable that the set of queries we have considered is heterogeneous, not a uniform one. Consequently, the query processing times we have obtained are very unlike, since depending on the query, not only the number of documents implied can change, but also the number of query servers where these documents are found in.

In these experiments we have only used a small set of results in order to have something from where we can keep on working. This is the reason for network time to be almost constant for all of our tests. Besides, there is not a significant discrepancy among the sizes of the packets from zero to ten results, and these packets do not exceed the Maximum Transfer Unit of the network, which could make the times slower.

Table 2. Times for 2 QSs.

	TCP				UDP				Multicast			
	QS1	QS2	Broker	Net.	QS1	QS2	Broker	Net.	QS1	QS2	Broker	Net.
μ	138.43	123.94	138.78	0.26	138.01	125	138.53	0.25	138.73	130.06	141.1	0.25
σ	95.53	84.79	95.5	0.43	95.04	84.91	94.88	0.43	94.99	84.94	93.85	0.54
p10	32.98	29.65	33.31	0.21	33.12	30.36	33.91	0.2	33.87	37.11	39.41	0.19
p90	269.48	237.93	269.72	0.28	268.37	239.11	268.61	0.27	269.45	244.06	269.93	0.26
μp	127.81	114.65	128.12	0.24	127.4	115.81	127.9	0.23	128.16	120.78	130.46	0.23
σp	63.29	56.06	63.25	0.02	62.84	56.21	62.65	0.02	62.82	55.53	61.52	0.02
I	5.43%				4.77%				2.96%			

Table 3. TCP times for 4 QSs.

	QS1	QS2	QS3	QS4	Broker	Network
μ	71.64	73.06	69.44	60	75.84	0.34
σ	49.14	51.82	48.4	40.68	53.18	0.8
p10	17.26	16.71	16.29	14.77	18.13	0.22
p90	139.23	146.63	136.77	114.49	151.53	0.33
μp	66.25	67.15	64	55.62	69.81	0.27
σp	32.7	34.78	32.05	26.89	35.93	0.03
I	12.07%					

Table 4. UDP times for 4 QSs.

	QS1	QS2	QS3	QS4	Broker	Network
μ	74.72	71.38	67.43	59.9	76.62	0.34
σ	53.35	50.26	46.38	40.75	54.5	0.79
p10	17.34	16.64	16.17	14.81	18.12	0.24
p90	151.09	142.28	129.15	114.26	153.72	0.34
μp	68.52	65.67	62.36	55.47	70.3	0.28
σp	35.72	33.5	30.55	26.89	36.72	0.03
I	11.95%					

Table 5. Multicast times for 4 QSs.

	QS1	QS2	QS3	QS4	Broker	Network
μ	74.7	72.25	67.6	59.92	76.71	0.31
σ	53.11	50.8	46.44	40.79	54.42	0.74
p10	17.43	16.92	16.33	14.79	18.28	0.21
p90	150.12	143.45	129.34	114.54	153.55	0.3
μp	68.54	66.51	62.53	55.49	70.42	0.25
σp	35.51	33.9	30.54	26.94	36.64	0.02
I	12.29%					

In spite of the fact that we have used only four servers, we could compare how document distribution brought about the fact that global execution times decrease as we increase the number of machines in the experiment. We can deduce this from the following table whose sizes are expressed in megabytes:

Table 6. Indexes

	1 QS	2 QSs	4 QSs
Lexicon	114.39	75.09	47.30
Direct index	441.60	223.91	106.61
Inverted index	454.09	233.71	112.76
Document index	46.17	24.37	12.55

Hence, when the size of the indexes decreases, I/O load decreases too, so we will spend less time in disk access, decreasing total time as well. However, it would be an error to conclude that whenever we increase the number of QSs, the total processing times will decrease, since if we had a big number of QSs, a bottleneck could appear (e.g. network, broker or even the I/O load).

Other point we have to emphasize is that the imbalance is accentuated as the quantity of servers grows. In this case, a cause can be that the times for the forth QS (shown in tables 3, 4 and 5) are considerably lower than in the rest of servers due to the fact that its created indexes are smaller than the remaining ones too, since it is hard to distribute the documents in a completely uniform way. On the other hand, the more QSs there are, the bigger the imbalance will be. The lower the processing times are, the more remarkable the divergences among times will be.

Finally, the generated load in a limited number of query servers does not seem to be enough to make impact on the global performance of the system, since network time remains constant, independently of the number of QSs (1, 2 or 4). This seems to be the reason whereby obtained data by means of different transport protocols are so similar.

6. CONCLUSIONS

In this article we have submitted a distributed IR system for performing different experiments of this subject. This system could be installed in a local area network or in a cluster with a good number of machines in order to corroborate by means of realistic experiments if some IR optimization techniques, like partial result sets reduction or brokers distribution, are really effective in practice and if the results of certain experiments (e.g. the works by Cacheda et al. [7][6][8]) are close to the ones taken down from a real environment.

In spite of we have used the .GOV collection in our initial experiments, we hope we will perform tests with a larger collection like .GOV2 (25,205,179 documents) in the future.

The model was also designed thinking in a possible extension and in the addition of new functionalities in order to study some other ways of optimization, like a better usage of the communications network (for example, partitioning the packets in the application layer, modifying or deleting their headers, creating an *ad-hoc* network protocol, etc.).

Nevertheless, we also have corroborated that for a low number of QSs, there is an imbalance among their execution times, and it grows as the number of QSs does, as it is shown in the previous work by Badue et al. [4].

Finally, as the experiments were performed with few QSs, we have not generated enough network load to make an impact on the obtained results, so the time spent on network operations was similar for all configurations, independently of the number of QSs or the transport protocol. We hope to study thoroughly this matter in future works, confirming that the number of partial results, the number of QSs and the transport protocol have an influence on this aspect.

REFERENCES

GOV collection. http://ir.dcs.gla.ac.uk/test_collections/govinfo.html.

GOV2 collection. http://ir.dcs.gla.ac.uk/test_collections/gov2-summary.htm.

2006 Efficiency Task Topics (TREC 2006 Terabyte Track), a.k.a. TREC Efficiency Track http://trec.nist.gov/data/terabyte/06/06.efficiency_topics.tar.gz.

Badue, C.S. et al. "Analyzing imbalance among homogeneous index servers in a web search system," in *Information Processing and Management Journal,* vol. 43 (3), pp. 592-608, 2007.

Badue, C.S. et al. "Basic Issues on the Processing of Web Queries," in *Proc. of the 28th International ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR'05)*, pp. 577–578, 2005.

Cacheda, F. et al. "A Case Study of Distributed Information Retrieval Architectures to Index One Terabyte of Text," *Information Processing and Management Journal*, vol. 41 (5), pp. 1141-1161, 2005.

Cacheda, F. et al. "Performance Analysis of Distributed Architectures to Index One Terabyte of Text," in *Proc. Of 26th European Conf. on Information Retrieval Research (ECIR'04)*, Lecture Notes on Computer Science (2997), pp. 394-408, 2004.

Cacheda, F. et al. "Performance Analysis of Distributed Information Retrieval Architectures Using an Improved Network Simulation Model," *Information Processing and Management Journal*, vol. 43 (1), pp. 204-224, 2007.

Cahoon, B. and McKinley, K.S. "Performance evaluation of a distributed architecture for information retrieval," in *Proc. of 19th ACM-SIGIR International Conf. on Research and Development in Information Retrieval*, pp: 110-118, 1996.

Hawking, D. "Scalable text retrieval for large digital libraries," *Lecture notes in computer science*, 1324, pp. 127–146, 1997.

Hawking, D. and Thistlewaite, P. "Methods for Information Server Selection," *ACM Transactions on Information Systems*, vol. 17 (1), pp. 40-76, 1999.

Jones, C.B. et al. "Spatial information retrieval and geographical ontologies an over-view of the SPIRIT project," in *Proc. of the 25th ACM-SIGIR Conf. on Research and Development in Information Retrieval*, pp. 387-388, 2002.

Lin, Z. and Zhou, S. “Parallelizing I/O intensive applications for a workstation cluster: a case study,” *ACM SIGARCH Computer Architecture News*, 21(5), pp. 15–22, 1993.

MacFarlane, A. et al. “Parallel Search using Partitioned Inverted Files,” in *Proc. of the 7th International Symposium on String Processing and Information Retrieval (SPIRE’00)*, pp. 209–220, 2000.

Moffat, A. et al. “A pipelined architecture for distributed text query evaluation,”. *Information Retrieval*, published on-line, 2006.

National Institute of Standards and Technology (NIST). http://www.nist.gov.

Ribeiro-Neto, B. and Barbosa, R. “Query performance for tightly coupled distributed digital libraries,” in *Proceedings of the 3rd ACM Conference on Digital Libraries*. New York: ACM Press, pp. 182-190, 1998.

Terrier (TERabyte RetrIEveR). http://ir.dcs.gla.ac.uk/terrier/.

TodoBR (2003). http://www.todobr.com.br.

Tomasic, A. and García-Molina, H. “Performance of inverted indices in shared-nothing distributed text document information retrieval systems,” in *Proc. of the 2nd International Conf. on Parallel and Distributed Information Systems*, pp. 8-17, 1993.

TREC Terabyte Track. http://www-nlpir.nist.gov/projects/terabyte/.

University of Glasgow. http://www.gla.ac.uk/.