ARTICLE IN PRESS

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Information Processing and Management xxx (2004) xxx–xxx

INFORMATION PROCESSING & MANAGEMENT

www.elsevier.com/locate/infoproman

A case study of distributed information retrieval architectures to index one terabyte of text

Fidel Cacheda [a,*], Vassilis Plachouras [b], Iadh Ounis [b]

[a] *Department of Information and Communication Technologies, Facultad de Informática, University of A Coruña, Campus de Elviña s/n, 15071 A Coruña, Spain*

[b] *Department of Computing Science, University of Glasgow, Glasgow G12 8QQ, UK*

Received 14 November 2003; accepted 3 May 2004

Abstract

The increasing number of documents to be indexed in many environments (Web, intranets, digital libraries) and the limitations of a single centralised index (lack of scalability, server overloading and failures), lead to the use of distributed information retrieval systems to efficiently search and locate the desired information. This work is a case study of different architectures for a distributed information retrieval system, in order to provide a guide to approximate the optimal architecture with a specific set of resources. We analyse the effectiveness of a distributed, replicated and clustered architecture simulating a variable number of workstations (from 1 up to 4096). A collection of approximately 94 million documents and 1 terabyte (TB) of text is used to test the performance of the different architectures. In a purely distributed information retrieval system, the brokers become the bottleneck due to the high number of local answer sets to be sorted. In a replicated system, the network is the bottleneck due to the high number of query servers and the continuous data interchange with the brokers. Finally, we demonstrate that a clustered system will outperform a replicated system if a high number of query servers is used, essentially due to the reduction of the network load. However a change in the distribution of the users' queries could reduce the performance of a clustered system.
© 2004 Elsevier Ltd. All rights reserved.

Keywords: Distributed information retrieval; Performance; Simulation

1. Introduction

The number of documents available on the Web is increasing daily, from several thousands in 1993 to more than 4 billion indexed by Google (2004) in 2004. There are several challenges faced by Web

[*] Corresponding author.
E-mail addresses: fidel@udc.es (F. Cacheda), vassilis@dcs.gla.ac.uk (V. Plachouras), ounis@dcs.gla.ac.uk (I. Ounis).

0306-4573/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ipm.2004.05.002

Information Retrieval (IR) systems in this highly dynamic environment. One major challenge is how to efficiently locate the desired information into all available data. Recent research in Web IR has led to the development of highly effective search engines that allow users to locate relevant or useful documents. A second challenge at the system level is how to design retrieval engines that can process a massive number of documents, while handling a considerable number of queries simultaneously.

Retrieval systems based on a single centralised index are subject to several limitations: lack of scalability, server overloading and failures (Hawking & Thistlewaite, 1999). Therefore, given these facts, it seems more appropriate to turn to the Distributed Information Retrieval (DIR) approach for the storage and search processing.

In a distributed search environment, there are usually two basic strategies for distributing the inverted index over a collection of query servers (Ribeiro-Neto & Barbosa, 1998; Tomasic & Garcia-Molina, 1993). One strategy is to partition the document collection so that each query server is responsible for a disjoint subset of documents in the collection (called *local inverted files* in Ribeiro-Neto & Barbosa (1998)). The other option is to partition based on the index terms so that each query server stores inverted lists corresponding to only a subset of the index terms in the collection (called *global inverted files* in Ribeiro-Neto & Barbosa (1998)). In the local inverted file organisation, a search term will be broadcasted to all the query servers, each of which would return a disjoint list of relevant documents. In the global inverted file organisation, only some query servers will receive some query terms, each of which would return a list of relevant documents for each term. Performance studies described in (Tomasic & Garcia-Molina, 1993) indicate that the local inverted file organisation uses system resources effectively, provides good query throughput in most cases and is more resilient to failures.

From the collection point of view, a distributed information retrieval system could follow a single-collection model or a multi-collection model (Callan, 2000). In the single-collection model, the collection of documents is indexed and distributed as a whole. In a multi-collection model, the whole collection of documents is divided into sub-collections that are indexed independently, where each sub-collection could have the inverted index distributed. A central server stores the descriptions of each sub-collection, and a sub-collection selection service will identify the most relevant sub-collection(s) for each query. The query is sent to the most relevant sub-collections and the results are merged considering that different rankings schemes could have been used.

This work is a case study of different architectures for a distributed information retrieval system. The SPIRIT collection (94,552,870 documents and 1 terabyte (TB) of text) (Jones et al., 2002) is used for the simulation of the distributed IR system. We partition the collection of documents using a local inverted file strategy, and we test the response times for different configurations. Although the obtained timings depend on the specific system simulated (>Plachouras, Ounis, Amati, & van Rijsbergen, 2002), the trends and conclusions should be independent of the system used. In this way, we intend to provide a guide to estimate the optimal architecture of a distributed IR system to index and search a very large collection like SPIRIT.

The improvements in the performance of a single-collection model are examined in a distributed and replicated system. Moreover, the effects of a multi-collection model are tested through a clustered system. Using the query topics provided in (Spink, Jansen, Wolfram, & Saracevic, 2002), we define a clustered system and compare its performance with a replicated system in order to measure the effect of changes in the query topics through time.

We start by presenting the related work in Section 2. In Section 3, we describe our simulation model, starting with the analytical model for a basic, not distributed, IR system. Next, a generic collection model is described and the key components of a distributed IR system are analysed. In Section 4, we describe the simulations performed for the different architectures: distributed, replicated and clustered system, and the results obtained. Finally, the main conclusions and the future work are presented in Section 5.

2. Related work

Related work for distributed IR systems includes the evaluation of the architecture performance, data partitioning, caching and multiprocessor systems. However, the work on architecture performance is the most directly related to this paper.

Harman, Mccoy, Toense, and Candela (1997) showed the feasibility of a distributed IR system, by developing a prototype architecture and performing user testing. However, they used a very small text collection (less than 1 GB data), and did not analyse efficiency issues.

Burkowski (1990) described a simulation study which measures the retrieval performance of a distributed IR system, using also a small collection. The performed experiments explore two strategies for distributing the workload across the servers. The first strategy distributes the collection among all servers uniformly. In the second strategy, the servers are divided into two groups: query evaluation and document retrieval. Burkowski concluded that the second strategy could provide better response times than the uniform approach under certain conditions, although the uniform strategy would perform generally better.

Lin and Zhou (1993) implemented a distributed IR system on a network of workstations, showing large speedup improvements due to parallelisation. The implemented retrieval model uses a variation of the signature file scheme to encode documents and to map the signature file across the network.

Coevreur et al. (1994) analysed the performance of searching large text collections (more than 100 GB data) on parallel systems. They used simulation models to investigate three different hardware architectures (a mainframe system, a collection of RISC processors and a special purpose machine architecture) and search algorithms. The focus of their work was to analyse the tradeoff between performance and cost, where the mainframe configuration was found to be the most effective.

Hawking (1997) designed and implemented a parallel IR system on a collection of workstations doing experiments with a maximum of 64 workstations. The basic architecture of the implemented system uses a central process to check for user commands and broadcast them to the servers in each workstation. The central process also merges the partial results before sending the final answer set to the user.

Cahoon and McKinley (1996) described the result of simulated experiments on the distributed INQUERY architecture. Using the observed behaviour for a mono-server implementation, they estimated the performance figures for a distributed implementation, proving it to be scalable. They experimented with collections up to 128 GB using a variety of workloads, and investigated how different system parameters could affect the performance and scalability of a distributed IR system, using up to 128 servers. Their results show that with a simple distributed architecture, the system maintains scalable performance at higher workloads.

Lu and McKinley (2000) analysed the effects of partial collection replication to improve the performance in a collection of 1 TB, simulating up to 33 servers. They concluded that the performance of partial replication with a connection broker exceeds that of a client-side caching or server-side caching.

The previous work for distributing the inverted index over a collection of servers is focused on the local and global inverted files strategies (Martin, Macleod, & Nordin, 1986; Ribeiro-Neto & Barbosa, 1998; Tomasic & Garcia-Molina, 1993), showing that the local inverted file is a more balanced strategy and a good query throughput could be achieved in most cases.

Our work is focused on the simulation and performance evaluation of several distributed architectures using a massive cluster of workstations (up to 4096) and identifying the limitations of each model. This work is especially related to Cahoon and McKinley (1996) and Ribeiro-Neto and Barbosa (1998), but it mainly differs in three points. First, the probabilistic model is considered and, therefore, disjunctive queries are used in our system (without the reduction in the answer set provided by the conjunctive operations). Second, a simple analytical model is developed initially for a single-collection/single-server environment (similarly to Ribeiro-Neto & Barbosa (1998)), and this will be the basis for the simulation of the distributed IR systems, composed of multiple query servers (similarly to Cahoon & McKinley (1996)). Third, initially the results of the analytical model are tested using the TREC WT10g collection and the set of the topic-relevance queries from TREC10 (Hawking & Craswell, 2001). Next, we model a document collection of 1 TB and a set of queries, in order to obtain more generic results.

3. Simulation model

To explore the performance of different architectures for a distributed IR system, we implemented a discrete event-oriented simulator using the JavaSim simulation environment (Little, 2001).

The simulation model defined in this work is divided into three parts. Initially an analytical model has been developed for the simulation of a simple IR system based on the WT10g collection and the queries from the topic-relevance task of TREC10 (Hawking & Craswell, 2001), using a single server. Next, a collection model is defined to simulate, in general, the behaviour of any collection of documents and in particular, a new collection composed of 94 million documents and 1 TB of text. Finally, the basic IR model is extended to a distributed IR model defining the behaviour of a local area network of computers and modelling the tasks of the query brokers and the query servers.

3.1. Analytical model

In this section, we describe a simplified analytical model for the querying process in the IR system described in (Plachouras et al., 2002), using the WT10g collection and the set of queries used for TREC10 (see (Bailey, Craswell, & Hawking, 2003) for more details about WT10g). This analytical model is similar to the one described in (Ribeiro-Neto & Barbosa, 1998) and despite its simplicity, this model captures the key factors that determine the query processing performance for the studied IR system.

A series of experiments were carried out to identify and estimate the basic variables and critical parameters of the analytical model. The notation for these variables and parameters is provided next:

q_i	vector of keywords for the *i*th query
k_i	number of keywords in query q_i
d_k	number of documents of the inverted list for keyword k
r_i	number of results obtained in query q_i
tc_1	first coefficient for the time to compare two identifiers and swap them (merging process)
tc_2	second coefficient for the time to compare two identifiers and swap them (sorting process)
ti	initialisation time, including memory allocation and output display
ts	average seek time for a single disk
tr	average time to read and process the information about one document in an inverted list (seek time is excluded)
t_i	total time (in ms) to complete the processing of query q_i.

Once the query server receives the query vector q_i for processing, it reads from disk the inverted lists associated with the k_i keywords, whose length is given by d_k. Then the inverted lists are merged and sorted to form the answer set whose length is given by r_i. The coefficients tc_1 and tc_2 are used in the calculation of the time to merge and sort the results. Ribeiro-Neto and Barbosa (1998) have modelled this process using a linear relationship, but from our experiments, a logarithmic model seems to fit more accurately as the number of results increases. Hence, the time to merge and sort n results (tc) is calculated as:

$$tc = tc_1 \times n + tc_2 \times \ln(n)$$

The processing of a query is divided into four phases: an initialisation phase (P_1), seeking disks (P_2), reading the inverted lists from disk and processing the weights (P_3) and obtaining the answer set and ranking the results (P_4). Therefore, the processing time for a query q_i is t_i, given by:

$$\begin{aligned} t_i = ti + & \qquad (P_1)\\ k_i \times ts + & \qquad (P_2)\\ \sum_{k \in q_i} d_k \times tr + & \qquad (P_3)\\ tc \times r_i & \qquad (P_4) \end{aligned}$$

In this analytical model, the parameters d_k and r_i have to be estimated accurately. We evaluate the accuracy of the estimation by processing the topic-relevance queries from TREC10's Web track with a real IR system (Plachouras et al., 2002). In this way, we obtain the exact number of results for each query and the exact number of documents associated with each of the inverted lists retrieved by the system.

To compare the accuracy of our analytical model with the real IR system, we measured the required time to process, on a single computer, the queries 501–550 used for the TREC10 topic-relevance task with the WT10g collection. This computer had 2 AMD Athlon processors at 1.4 GHz and 2 GB RAM. The processing time of the same 50 queries was also evaluated for the proposed analytical model. Both Mann–Whitney and Kolmogorov–Smirnov two sample tests confirm the correspondence between the real and predicted processing times, with p-values of 0.956 and 1.0, respectively. In Fig. 1, we show the cumulative response times for the real system and our analytical model.

3.2. The spirit collection model

In a simulation study, the documents and queries can be represented using two different techniques. One is to use a real collection of documents and actual queries. The second technique consists of generating synthetic databases and queries, from the probability distributions that are based on real statistics (Tomasic & Garcia-Molina, 1993).

The former is more realistic and it allows the comparison between the simulation model and the real system (as described in the previous section), but it is limited to a particular application and domain. The use of synthetic data provides more flexibility for studying a wider range of configurations.

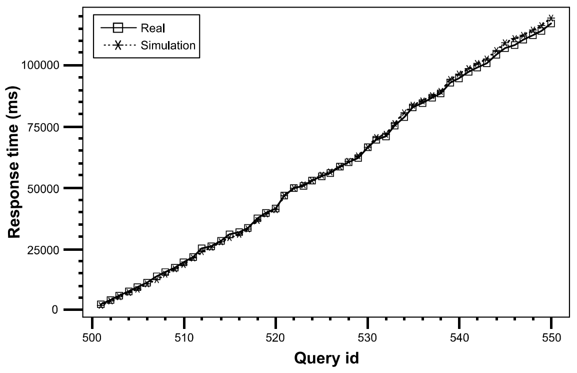

Fig. 1. Comparison between real and estimated execution times for 50 WT10g queries. Parameter values: $ti = 1400$ ms, $ts = 0.03$ ms, $tr = 4.0208$ µs, $tc1 = 0.00013068$, $tc2 = 0.000096$.

As a consequence, the basic analytical model defined for the WT10g collection will be extended to work with synthetic databases and queries. The objective is to simulate the so-called SPIRIT collection, composed of approximately 94 million Web documents and 1 TB of text, although no queries and relevant assessments exist for the moment (Jones et al., 2002). We divide this collection in 72 sub-collections and we use the statistical information (vocabulary size, document size, etc.) of one of them to model the whole collection.

3.2.1. Document model

For the document model, we first study the main parameters of one of the sub-collections, and using this as a basis, the values for the whole SPIRIT collection are estimated. In Table 1, we provide definitions for the considered parameters.

The first column describes the parameters that represent a database of documents. The database consists of a collection of D documents. Each document is generated by a sequence of W independent and identically distributed words. Each word is uniquely identified by an integer w in the range $1 \leqslant w \leqslant T$, where $T = |V|$. The probability distribution F describes the probability that any word appears and, for convenience, is arranged in decreasing order of probability.

The second column of the table represents our base case scenario and the values are obtained from one fraction of the SPIRIT collection. To define a specific probability distribution Z_1 of F, a distribution is fitted to the rank/occurrence plot of the vocabulary, and then normalised to a probability distribution. Fig. 2 shows the log–log graph of a linear and quadratic model fit to some of the 500,000 most frequent words.

Table 1
Parameters for the document model

Parameter	Real values (SPIRIT sub-collection)	Estimated values (whole SPIRIT collection)	Description
D	1,221,034	94,552,870	The number of documents
W	456	456	Average words per document
V	–	–	The vocabulary
T	4,301,776	73,689,638	Total words in V, i.e. $T = \|V\|$
$F(w)$	$Z_1(w)$	$Z_2(w)$	Pr(word = w)

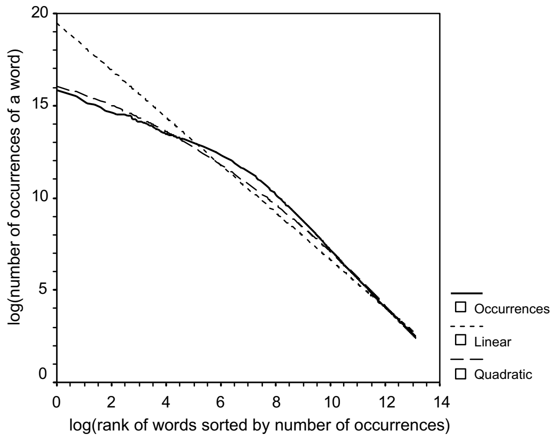

Fig. 2. Linear and quadratic fit to vocabulary occurrence data.

The X axis represents the words in the vocabulary, ranked by the number of occurrences in decreasing order. The Y axis represents the number of occurrences of each word.

A regression analysis was performed and confirmed that the quadratic model fits the real distribution ($R=0.99770$) better than the linear model, which corresponds to the Zipf's law ($R=0.98122$). The quadratic model is similar to Zipf's, although in previous works (Tomasic & Garcia-Molina, 1993), it has proved to match the actual distribution better.

Given the quadratic fit curve, the form of the probability distribution $Z_1(w)$ is obtained from the quadratic model, and then divided by a normalisation constant, to make the probabilities sum to 1 (Tomasic & Garcia-Molina, 1993):

$$Z_1(w) = \frac{w^{-0.046648\times\ln w-0.432720}\mathrm{e}^{16.068777}}{4.284081\times 10^8}$$

The third column of Table 1 shows the estimated values of the parameters for the whole SPIRIT collection. The number of documents in the collection is 94,552,870. The average number of words per document is supposed to remain stable. Therefore, the same value as the base case is chosen.

The size of the vocabulary of a collection of documents is supposed to match the Heaps law (Heaps, 1978), which states that the vocabulary size of a collection of documents with n terms is given by:

$$V = Kn^{\beta}$$

where K and β are two parameters that depend on the collection. To estimate these parameters, the values of V and n were measured at several points during the indexing process of the sub-collection, and a regression analysis was performed. Fig. 3 shows the log–log graph of the real values and a linear model fit, proving a strong linear relationship ($R=0.999$). The estimated values for the parameters are: $K=4.60363$ and $\beta=0.6776$.

Therefore, to estimate the size of the vocabulary of the whole collection, the Heaps law is used for the 94 million documents and 456 words per document, obtaining an approximation of 73,689,638 unique terms for the whole collection.

Finally, a different probability distribution is provided for the whole collection. Given the quadratic fit curve previously described, a new normalisation constant is defined for the new vocabulary size:

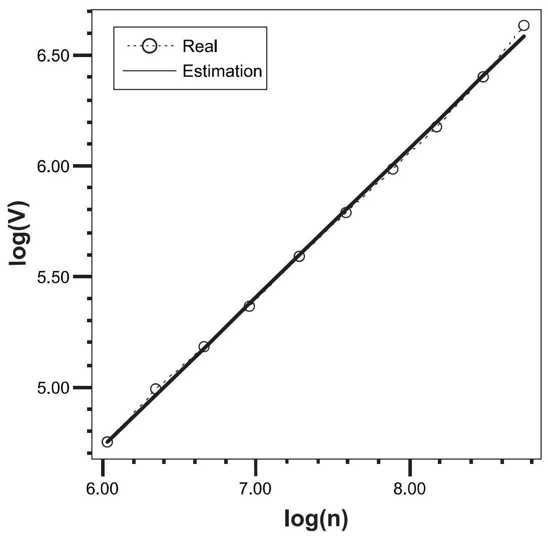

Fig. 3. Log–log plot of unique vocabulary terms vs. total terms, fit to a linear model.

$$Z_2(w) = \frac{w^{-0.046648\times\ln w-0.432720}\mathrm{e}^{16.068777}}{4.294476\times 10^8}$$

3.2.2. Query model

A query is a sequence of terms $(t_1,\ldots,t_k)$ generated from K independent and identically distributed trials from the probability distribution $Q(t)$. In our simulation study, the number of terms is selected uniformly between 1 and 4 terms per query, based on the terms used in the TREC10 topic-relevance queries. In Table 2, we present a description of each parameter and the corresponding values we used for the simulations.

There is little published data on the probability distribution Q. Tomasic and Garcia-Molina (1993) assume the uniform term distribution with the fraction parameter for the query model. However, Jeong and Omiecinski (1995) model Q, assuming that the probability of a term occurring in a query is proportional to that term's frequency in the collection. These two different query models are named uniform query model and skewed query model, respectively.

The probability distributions $Q(t)$ for the two query models are:

Uniform Query Model:

$$Q(t) = \begin{cases} \dfrac{1}{(u-s)T} & \text{if } sT\leqslant t\leqslant uT \\ 0 & \text{otherwise} \end{cases}$$

Skewed Query Model:

$$Q(t) = \begin{cases} C\times Z(t) & \text{if } sT\leqslant t\leqslant uT \\ 0 & \text{otherwise} \end{cases} \quad \text{where } 1=\sum_{i=sT}^{uT} C\times Z(i)$$

The parameters u and s affect the probability that a term appears in a query. As u decreases, the probability of choosing a word of low rank increases. Words of low rank occur frequently in the documents of the collection. Therefore, if u is too small, the queries will retrieve a large fraction of the documents. On the other hand, if u is too large, the answer sets will be too small (Jeong & Omiecinski, 1995). The parameter s is introduced to avoid the effect of the first words in the rank, i.e. stopwords, which increase excessively the number of results obtained. As s increases, more words from the top rank are considered to be stopwords, and therefore are not used as query terms.

At certain points in the simulation, we will need to know the expected size of an inverted list for a query term and the expected size of an answer set for a given query. Let us assume a query with terms $t_1,\ldots,t_k$ that is executed in a collection (or sub-collection) of documents of size *Documents*. If we are considering the whole collection *Documents* = D, but in a distributed environment, *Documents* corresponds to the number of documents covered by each of the distributed indices. Therefore, the number of documents of an inverted list for term t_i will be (Tomasic & Garcia-Molina, 1993):

$$\mathit{Documents}\times[1-(1-Z(t_i))^W]$$

Table 2
Parameters for the query model

Parameter	Value	Description
K	[1–4]	The number of terms per query
$F_q(t)$	$Q(t)$	Pr(term = t)
u		The fraction of T (in rank order of V) used in the query terms
s		The fraction of T (in rank order of V) skipped for the query terms

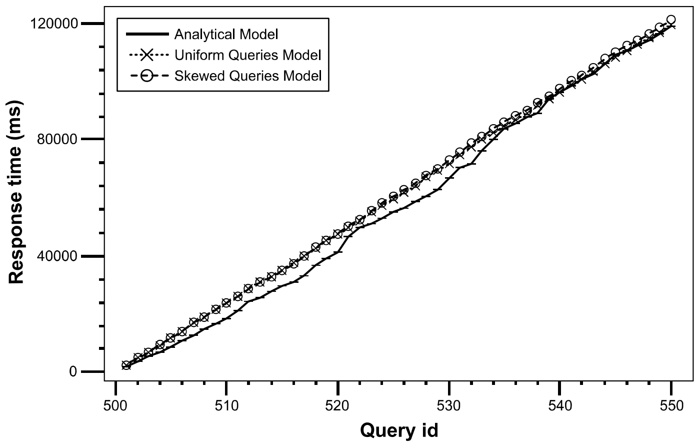

Fig. 4. Comparison of the response times for the uniform and skewed query models with the analytical model.

Consequently, the expected size of the answer set for a query with terms $t_1, \ldots, t_k$ (supposing a disjunctive query) is:

$$\mathit{Documents} \times [1 - (1 - Z(t_1))^W \times \cdots \times (1 - Z(t_k))^W]$$

In order to test the accuracy of the described SPIRIT document collection, a simulation was performed to replicate the results for the WT10g collection, with the analytical model. First, the parameters u and s were configured empirically to retrieve, on average, the same number of documents as the real set of queries (approximately, 150,000 documents per query, 9% of the collection), for both query models. For the uniform query model, the values used were: $u = 0.0009$ and $s = 0.00005$; for the skewed query model, the values used were: $u = 0.01$ and $s = 0.00007$. In both cases, u was selected to be as small as possible, in order to use a realistic portion of the vocabulary, trying to avoid uncommon terms.

The comparison of the response times of both query models and the analytical model is shown in Fig. 4. Here, it is clear that the simulations using the query models produce on average similar response times as the baseline. Although the fluctuations of the real queries are not present in the query models (probably due to the uniform selection of the terms per query in our query models), these models can be considered as quite accurate for the average response times.

On the other hand, there is no clear difference between the response times for the uniform query model and the skewed query model. However, the effect of the parameters u and s on the words potentially used in the queries is quite important. The uniform query model is limited to only 2463 different words, while the skewed model could use more than 28,000 different words from the vocabulary. Note that in both cases, we used a vocabulary of 2,898,203 terms to retrieve on average 150,000 documents per query. [1] Therefore, the skewed model is considered to represent better the real queries.

3.3. Distributed model

In a distributed IR system, the queries are stored in a global queue, which is controlled by one or more *central brokers*. Each broker will take one query and, depending on the implemented index organisation

[1] In the uniform model, all the terms have the same probability of being selected. Therefore, only those terms with a frequency close to the average number of retrieved documents per query could be used. In the skewed model, the probability is directly related to the term frequency. Therefore, a wider range of terms could be used, producing more realistic queries.

(local or global), will send it to a different number of query servers through a network (Ribeiro-Neto & Barbosa, 1998).

In this work, we do not evaluate the impact of the query arrivals on the performance of an interactive system. We assume that the system is operating in batch mode and that there are always enough queries to fill a minimum-size query processing queue. Initially, following the TREC practice, we assume that this queue has a size equal to 50, i.e. a batch of 50 queries is processed for each experiment.

Moreover, this work is focused on a local index organisation (see Fig. 5). In this organisation, a broker takes a query out of the queue and sends it to all query servers in the network. Each query server then processes the whole query locally, obtains the answer set for that query, ranks the documents, selects a certain number of documents from the top of the ranking and returns them to the broker. The broker collects all the local answer sets and combines them into a global and final ranked set of documents. Once a broker finishes with one query, it is ready to process the next one from the queue. If more brokers are available, then more than one query could be sent to the query servers, which will process them sequentially.

This process is repeated until the user queries queue is emptied. The total response time starts when the first query is taken from the queue and finishes when the last query has been processed.

The analytical model previously described is now extended to support the definition of a distributed IR system, with local index organisation. Some new parameters are defined:

$d_{k,j}$	number of documents of the inverted list for keyword k on query server j
$r_{i,j}$	number of results obtained for query q_i on query server j
$tr_{\max}$	maximum number of top ranked documents returned as the local answer set (1000 by default)
$tr_{i,j}$	number of documents from the top ranking in query q_i returned as the local answer set for query server j
$t_{i,j}$	total time (in ms) to complete the processing of query q_i at query server j
$rq_{i,j}$	time to receive the query q_i for the query server j
$ra_{i,j}$	time to receive the local answer set for query q_i from the query server j

As a consequence, the time for the query server j to process the query q_i is given by:

$$\begin{aligned} t_{i,j} = rq_{i,j} + & \\ & ti + && (\mathrm{P}_1) \\ & k_i \times ts + && (\mathrm{P}_2) \\ & \sum_{k \in q_i} d_{k,j} \times tr + && (\mathrm{P}_3) \\ & tc \times r_{i,j} && (\mathrm{P}_4) \end{aligned}$$

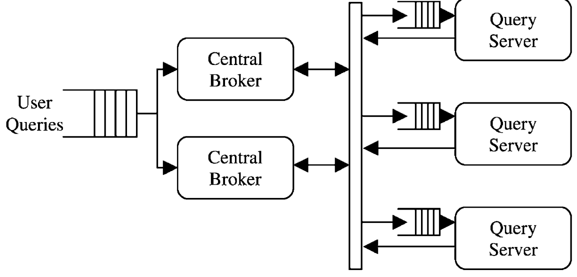

Fig. 5. Distributed IR model.

The parameters $d_{k,j}$ and $r_{i,j}$ are estimated using the collection model described in the previous section.

As soon as the broker has received all the local results from all the query servers, it must combine them to obtain the final answer set. Therefore, the total processing time for query q_i could be given by:

$$\begin{aligned} t_i = {} & \max(t_{i,j}) + \\ & \max(ra_{i,j}) + \\ & \sum_j tr_{i,j} \times tc \end{aligned}$$

The problem is that the parameters $rq_{i,j}$ and $ra_{i,j}$ can not be estimated using an analytical model as they directly depend on the network load of each moment. Therefore, it is necessary to capture the behaviour of the network to represent accurately the response times of a distributed IR system.

In our case, the system will contain a single LAN that will be simulated by a single FCFS infinite length queue. This LAN will manage all the messages sent by the brokers to the query servers and the answers from the query servers to the brokers. The service time for a request is calculated by the equation:

$$LANOverhead + RequestLength \times \frac{1}{LANBandwith/8} \times 1000$$

In Table 3, we describe the parameters used in the simulation of the network and show their corresponding values. The above *RequestLength* parameter depends on the type of message sent. If a query is sent to the query servers, the value of the *QuerySize* parameter will be used. If the local answer set for query q_i is sent from query server j to the broker, then the length of the packet will be: $tr_{i,j} \times DocAnswerSize$.

4. Simulation results

This section describes the results of several experiments, in which we used the simulation model described in the previous section. The objective is to investigate different approaches for the distribution and replication of the collection using a cluster of query servers, and compare the performance between the different configurations.

All the simulations are based on the 1 TB SPIRIT collection model. We model the queries with the skewed query model and, following a worst case scenario, we assume that each query would approximately retrieve 8.4 million documents (about 9% of the whole collection). A batch of 50 queries is used to test the performance, and for each different configuration, five different simulations (with distinct initial seeds) are run, and the average values for the execution times are calculated for each query.

The experiments are designed to test the performance of different architectures that distribute or replicate in several ways the collection. Initially, a purely distributed system is examined. Next, the effects of the replications are analysed and then, we examine possible configurations of a clustered system (based on an asymmetric distribution and replications).

Table 3
Parameters for the distributed model

Parameter	Value	Description
LANOverhead	0.1 ms	Network overhead for each packet sent
LANBandwidth	100 Mbps	Network speed (in bits per second)
QuerySize	100 bytes	Number of bytes sent from the broker to the query servers for each query request
DocAnswerSize	8 bytes	Number of bytes per document sent in the local answer sets to the broker

4.1. Distributed system

In this set of experiments, the collection of documents is distributed using the local index organisation over N query servers, where $N = 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 768$ and 1024. We simulate a distributed IR system with 1, 2, 3 and 4 brokers respectively. The results are displayed in Fig. 6.

The best performance is obtained when the query servers are continuously processing queries, without any idle period while waiting for new queries. If only one broker is used, the query servers will have long idle periods, especially as the number of query servers increases. As a consequence, the processing time in the broker for each query also increases (see Fig. 6).

The optimal performance is achieved when two or more brokers are used. In fact, with less than 512 query servers, two brokers are able to provide queries to the servers, continuously and, therefore, the performance of the system is maximised. However, there is still a bottleneck with 768 or 1024 query servers, with inactivity periods that will reduce the throughput. Three brokers will provide the maximum throughput, and no further benefit is obtained if we increase the number of brokers.

The bottleneck in the brokers is due to the number of local answer sets received from all the query servers that must be sorted. Increasing the number of query servers will benefit the processing time in the query servers, as each server stores a smaller inverted index. On the other hand, the brokers will receive more local answer sets to be merged into the final result set, as the number of query servers increases. This results in increasing the processing time in the brokers and eventually, leads to inactivity periods in the query servers while waiting for new queries from the brokers. In fact, if the number of query servers is high enough, the performance will start deteriorating at a certain point, independently of the number of brokers used.

The load on the brokers could be lowered in two ways. The first one is to reduce the number of documents included in the local answer sets by the brokers, with a possible repercussion on the precision and recall of the system, which must be analysed. The second way is to reduce the number of local lists received by the brokers, which can be achieved by designing a distributed protocol for the brokers, in order to filter and reduce the answer sets gradually.

In a system with three brokers, the throughput tends to be stabilised around 0.64 queries/s with 512 query servers, with minor improvements as the number of servers increases (0.66 queries/s with 1024 query servers).

Working with an optimal configuration of three brokers, Table 4 provides an estimation of the expected time in minutes to process 50 queries with a distributed IR system, using from 1 to 1024 query servers.

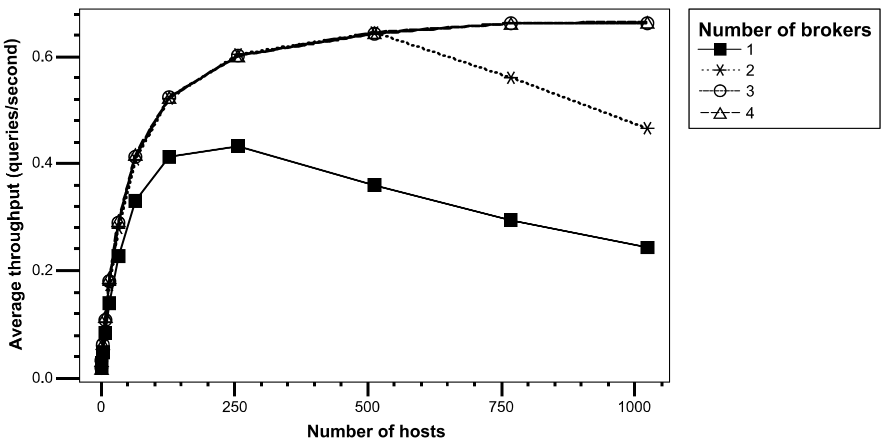

Fig. 6. Throughput for the simulation of a distributed IR system with local index organisation.

Table 4
Estimated time (in min) to process 50 queries by the distributed IR system with 3 brokers

Query servers	Time (mm:ss)	Query servers	Time (mm:ss)
1	46:01	*64*	02:00
2	24:40	*128*	01:35
4	13:20	*256*	01:23
8	07:37	*512*	01:17
16	04:36	*768*	01:15
32	02:53	*1024*	01:15

4.2. Replicated system

A replicated system is composed of one or more distributed IR systems. Each distributed system indexes the whole collection, and all the distributed systems that have been replicated, have the same number of query servers. In this case, the distributed system previously described could be seen as a replicated system, with only one replica.

In a replicated system, the brokers must decide initially which replica will process the query, and then broadcast the query to all the query servers in the replica. The objective of the selection of the replicas is to balance the load through all the replicas to obtain an optimal performance for the whole system. In our case, a round robin policy is used to distribute the queries to the replicas. Each broker will select a different initial replica and for each following query the next replica is selected.

Firstly, we analyse the optimal number of brokers required in a generic replicated system. To study this, we simulated a set of replicated systems, changing the number of brokers used. A summary of the results is provided in Table 5.

Initially, a system with two replications is simulated, using a variable number of brokers. With only four brokers, there is a reduction in the performance, following the pattern of the basic distributed system with two brokers (decreasing with 768 or 1024 hosts per replica). Using five brokers, a nearly optimal

Table 5
Throughput (queries per second) for different replicated IR systems

Query servers	$R=1$	$R=2$			$R=3$			$R=4$		
	$B=3$	$B=4$	$B=5$	$B=6$	$B=6$	$B=7$	$B=8$	$B=8$	$B=9$	$B=10$
1	**0.02**	0.03	**0.03**	0.03	0.05	**0.05**	0.05	0.06	**0.06**	0.05
2	**0.03**	0.05	**0.06**	0.06	0.08	**0.09**	0.08	0.11	**0.11**	0.11
4	**0.06**	0.1	**0.11**	0.11	0.14	**0.15**	0.16	0.19	**0.19**	0.2
8	**0.11**	0.18	**0.2**	0.2	0.27	**0.27**	0.29	0.35	**0.36**	0.36
16	**0.18**	0.3	**0.34**	0.35	0.47	**0.47**	0.52	0.61	**0.64**	0.63
32	**0.29**	0.5	**0.53**	0.55	0.73	**0.77**	0.8	0.98	**0.99**	0.99
64	**0.41**	0.75	**0.78**	0.81	1.1	**1.18**	1.1	1.46	**1.48**	1.47
128	**0.52**	1	**0.98**	1	1.46	**1.42**	1.4	1.95	**1.93**	1.9
256	**0.6**	1.17	**1.16**	1.18	1.72	**1.7**	1.73	2.2	**2.18**	2.26
512	**0.64**	1.19	**1.24**	1.27	1.6	**1.78**	1.81	1.95	**2.05**	2.13
768	**0.66**	0.96	**1.24**	1.29	1.17	**1.42**	1.46	1.26	**1.45**	1.47
1024	**0.66**	0.85	**0.99**	1.07	1.06	**1.08**	1.11	1.11	**1.13**	1.13

The "*Query servers*" column represents the number of servers per replica. Each column indicates the number of replications (*R*) and the number of brokers used (*B*).

throughput is achieved. A further increase in the number of brokers will only slightly improve the performance (and simultaneously, the network load).

The case of the systems with three and four replications is quite similar. With six and eight brokers respectively, there is a decrease in the performance for more than 512 hosts, reproducing the behaviour of one unique distributed system. As in the previous case, one more broker is sufficient to avoid the bottleneck and serve properly all the query servers.

Generally, for the simulated configurations, the number of brokers necessary to achieve near optimal performance for a generic replicated system, with R replicas, is given by: $2R+1$. With $2R$ brokers, there is still a bottleneck when the number of query servers is high, and this extra broker will reduce the idle times in the hosts. If more brokers are added only slight improvements can be achieved, especially with more than 256 query servers. If the number of replications is further increased, more extra brokers would be necessary to maintain throughput at the same levels.

Another important point in the replicated systems is the relation between the throughput and the number of replicas. If a basic distributed system has a throughput of T queries/min, then the theoretically maximum throughput for a system with R replicas will be $T * R$.

This is consistent with the results obtained in Table 5, and especially when there are fewer than 128 query servers per replica. In this case, the throughput obtained for the different replicated systems, with the optimal number of brokers (or more), is slightly below the theoretical value. This is due to the round robin distribution policy used in the brokers, as it can lead to some small periods of inactivity at certain replicas. In the future, we will analyse other distribution policies, similar to the one used in (Lu & McKinley, 2000), in order to improve the throughput up to the optimal theoretical value.

In addition, it is important to stress that if more than 256 (or 512 for some cases) query servers are used per replica, the performance of the system decreases rapidly (see shaded areas of Table 5).

Fig. 7 shows the throughput of each replicated system (configured with the optimal number of brokers) versus the total number of query servers in the whole system. If the number of query servers in the system is less than 1000, the performance improves with each new replica added. However, if the number of query servers is over this limit, the performance decreases, especially as more replicas are added to the system. In fact, a system with 4 replicas of 1024 query servers has a lower throughput than a system with 4 replicas of 64 servers each.

This decrease in performance is due to the network. Each replica adds more hosts to the network, which is used intensively to send the results back to the brokers. As a consequence, the network latency greatly

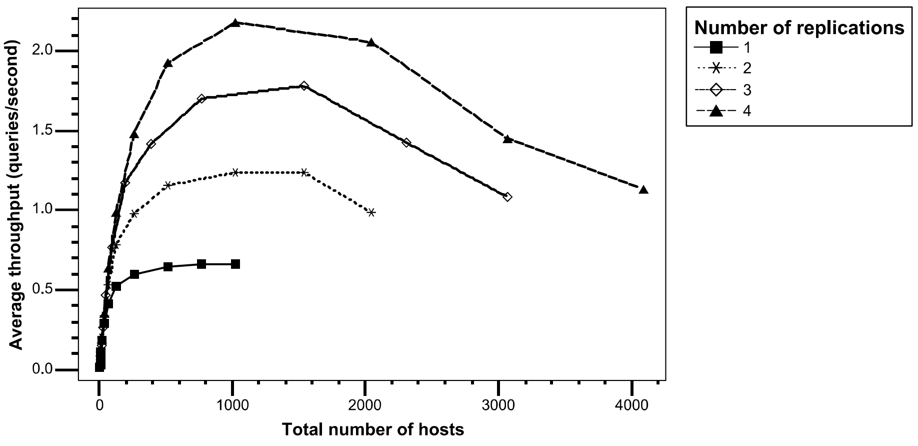

Fig. 7. Throughput (queries per second) for different replicated IR systems versus the total number of query servers in the system.

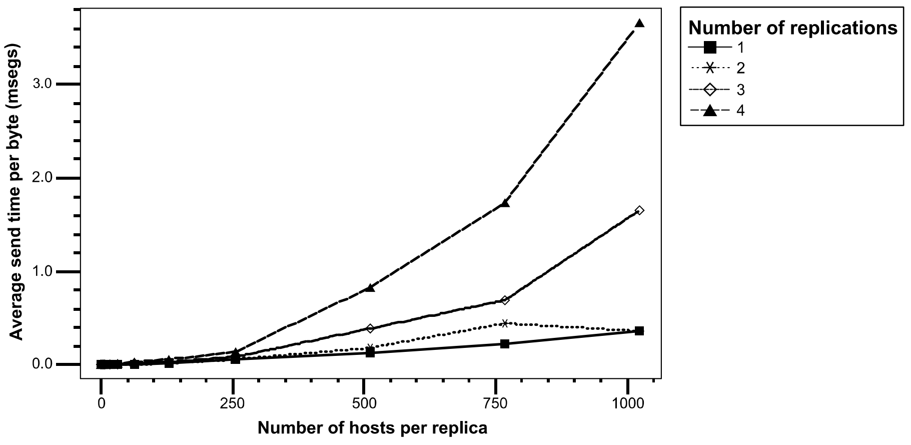

Fig. 8. Network load for different replicated IR systems versus the number of query servers per replica.

increases with each new replica added (see Fig. 8), making the network a bottleneck for the whole system. In a system with one replica and 1024 query servers, each byte will reach its destination in 0.36 ms on average. However, in a system with four replicas and 1024 query servers per replica, the time each byte needs to reach its destination increases 10 times. Hence, all the messages sent through the network are highly delayed producing inactivity periods on both query servers and brokers.

The effect of this bottleneck could be reduced in three different (and independent) ways. First, we can reduce the number of results included in the local answer sets sent from the query servers, although this may also affect the precision and recall of the IR system. Second, we can reduce the size of the local answer sets using compression techniques. Third, we employ different network configurations. In this work, we have only simulated a basic 100 Mbps LAN, although several improvements could be achieved with more complex configurations, using switches, routers and other network technologies apart from Ethernet, i.e. ATM or Gigabit Ethernet.

In the next section, we explore the benefits obtained with the second option. The remaining solutions require a deeper analysis and will be considered in future experiments.

4.2.1. Compression to reduce network congestion

The described network congestion is mainly due to the answers sent by the query servers to the brokers. Nearly all query servers of each replica will send their local results at the same time and above all, the amount of information sent is quite high.

As described in Section 3.3, each server will send a maximum of $tr_{\max}$ results (with a default value of 1000) and the information about each document is codified using *DocAnswerSize* bytes (currently, 8 bytes). This means that each server will send 8000 bytes to the broker, and, in the worst case (1024 servers per replica), the broker will receive 8000 Kbytes of answers for each query.

To study the benefits of reducing the information sent from the query servers to the brokers, a basic distributed model (without replication) was simulated, compressing the local answer sets (no compression and decompression times were considered). This experiment is also equivalent to a reduction in the size of the local answer sets, although this could affect the retrieval effectiveness of the IR system. We have made the hypothetical assumption that the local answer sets could be compressed to: 75%, 50%, 25% and 10%. Table 6 describes the expected latency of the network and the new achieved throughput, as the information sent to the brokers is being compressed. The study is centred on 256 or more query servers.

Table 6
Average throughput (*T*) and average send time per byte in seconds (*S*), using different compression ratios, in a distributed IR system (one replica)

Query servers	100%		75%		50%		25%		10%	
	T	*S*	*T*	*S*	*T*	*S*	*T*	*S*	*T*	*S*
256	0.601	0.055	0.602	0.056	0.603	0.056	0.604	0.058	0.604	0.062
512	0.644	0.127	0.646	0.127	0.648	0.131	0.65	0.135	0.651	0.14
768	0.662	0.234	0.666	0.222	0.669	0.19	0.672	0.213	0.674	0.225
1024	0.662	0.359	0.667	0.322	0.671	0.295	0.675	0.277	0.678	0.294

On the contrary to what we were expecting, the average time to send one byte does not decrease significantly as the compression ratio is increasing. The same seems to happen to the throughput obtained in the system: there is a slight improvement as the data are being compressed, but not relevant. In fact, with a compression ratio of 10% (sending 800 bytes, instead of 8000), the throughput improves by less than 2.5%.

The same tests were repeated for the replicated systems and the results are displayed in Fig. 9. The stronger line represents the throughput of the replicated systems with a 10% of compression, while the thinner line is the baseline (no compression). As was described previously, there is no significant improvement in a system with only one replica (both lines are superimposed). But, for two, three and four replicas, the effect of the compression is more important, avoiding the reduction in the performance with a high number of hosts. In these three cases, the performance of the baseline systems decreased when more than 1000 servers were simulated. However, the compression of the local answer sets to 10%, resulted in higher throughput. In fact, the behaviour of each of the replicated systems is analogous to a distributed system, with the performance increasing proportionally to the number of replicas defined.

These results underline the importance of the volume of information sent from the query servers to the brokers. In a distributed system, the network is not the bottleneck (at least, with the number of hosts simulated) and therefore, there is little improvement in the throughput even by reducing up to a 10% the volume of information. However, on a larger system, where the network is the bottleneck, this reduction is lightening the load of the network and permits the system to achieve its best performance.

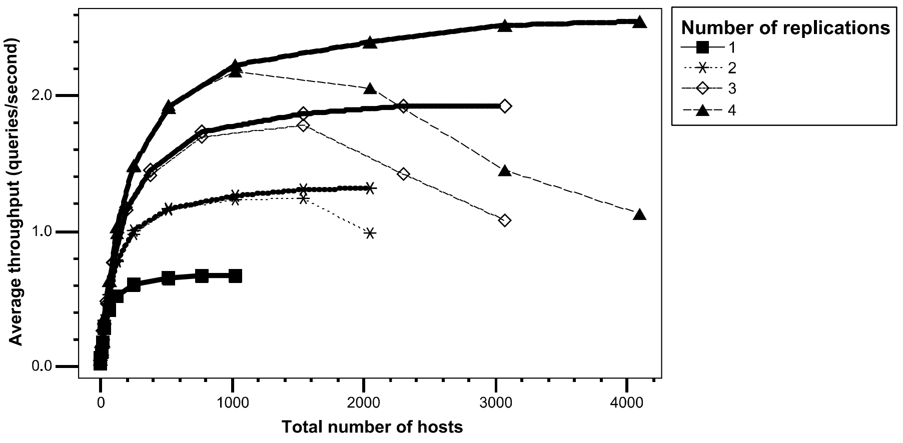

Fig. 9. Throughput for several replicated IR systems, with a 10% of compression ratio. The stronger line represents a 10% compression and the thinner line is the baseline (without compression).

4.3. Clustered system

A clustered system is divided into groups of computers, where each group operates as an autonomous distributed and replicated IR system. Each cluster can be composed of a different number of query servers. We assume that each cluster is responsible for one disjoint part of the whole collection of documents, and each cluster could use distribution and replication to store its respective index.

The brokers are global for the whole IR system. First, a broker must determine the appropriate cluster for each query and then, it should broadcast the query to the selected clustered system. If the cluster supports replication, then the broker will also decide to which replica the query will be sent (e.g. by using the round robin policy, as described previously).

Different commercial Web IR systems claim to use a clustered system adapted to the distributions of the queries received, e.g. AllTheWeb (Risvik & Michelsen, 2002). Therefore, the objective of these experiments is to test if the performance of a replicated IR system could be improved using a clustered system fitted to a distribution of queries, and how the changes of this distribution will affect the performance.

In the work by Spink et al. (2002), a set of real queries of Web users is categorised into 11 different topics. Moreover, the variations in the percentage of queries for each topic are analysed in three different years: 2001, 1999 and 1997. Table 7 provides a summary of the 11 topics and the percentage of queries through the different years. In the simulated systems, once a query is generated, it is automatically assigned to a topic using these distributions values. In these simulations, we increase the number of queries to 200, which are classified into one of the 11 topic categories. Moreover, the queries will retrieve 3 million documents on average to fit the size of the clusters.

In these experiments, we assume that each topic is indexed in a different cluster. The collection is divided into 11 sub-collections with an inverted file of approximately the same size, that is 8.5 million documents and, therefore, the 11 defined clusters will index the same number of documents, although using a different number of query servers. This is done for convenience; in this way, the behaviour of all the clusters will fit the same optimal throughput curve.

The base sub-collection of 8.5 million documents has been distributed over N query servers, where $N=1$, 2, 4, 8, 16, 32, 64, 128, 256 and 512. The throughput matches the previous results displayed in Fig. 6, with an optimal configuration of two brokers. The throughput values are displayed in Table 8.

Two different configurations have been tested for the clustered system. The first one has 128 query servers and the second one has 1024 query servers. Each cluster is assigned a number of query servers proportional to the percentage of queries that it is expected to receive (see Table 7).

Table 7
Distribution of queries across general topic categories, and configurations for the clustered systems simulated

Topics	2001 (%)	1999 (%)	1997 (%)	Config 1	Config 2
Commerce	24.755	24.73	13.03	8*4	63*4
People	19.754	20.53	6.43	6*4	51*4
Non-English	11.355	7.03	3.84	5*3	39*3
Computers	9.654	11.13	12.24	4*3	33*3
Pornography	8.555	7.73	16.54	5*2	44*2
Sciences	7.554	8.02	9.24	5*2	38*2
Entertainment	6.655	7.73	19.64	4*2	34*2
Education	4.554	5.52	5.33	6*1	47*1
Society	3.955	4.43	5.44	5*1	41*1
Government	2.054	1.82	3.13	3*1	21*1
Arts	1.155	1.33	5.14	2*1	12*1

Table 8
Average throughput (queries/s) for the basic sub-collection (8.5 million documents), in a distributed IR system

	Query servers									
	1	2	4	8	16	32	64	128	256	512
Throughput	0.05	0.081	0.14	0.228	0.349	0.476	0.573	0.637	0.671	0.687

The clustered systems are configured according to the most recent distribution of the topics, corresponding to the year 2001. In Table 7, columns "*Config 1*" and "*Config 2*" describe the query servers assigned to each topic, respectively. The first number corresponds to the number of distributed query servers, and the second one stands for the number of replicas in each cluster (the excepted throughput for each topic configuration can be estimated from Table 8). The total number of query servers assigned to each topic is associated with the distribution of the topics on the year 2001. For each topic, the distribution and replication of the query servers have been chosen analogously in both configurations, trying to maximise the replications for the most popular topics.

For the first configuration, the baseline is a replicated IR system, with 4 replications of 32 query servers each. On the other hand, the baseline for the second configuration is a replicated system with 4 replicas of 256 query servers each.

Fig. 10 presents the box diagram for the response time for the first 100 queries processed by the tested systems (queries 1–100). All the systems were tested for queries following the topics of years 2001, 1999 and 1997. Obviously, the performance of a replicated IR system does not depend on the type of queries (the baseline is independent of this factor), and the response times for the clustered system with 128 servers are labeled "*Config 1-2001*", "*Config 1-1999*" and "*Config 1-1997*", respectively.

The first clear conclusion is that the clustered system does not outperform a replicated system. The replicated system will process one query in 4682 ms, while the clustered system optimally configured for the 2001 queries will just process one query in 7806 ms (approximately the same performance as a system with two replicas of 64 servers). On the contrary, the clustered system reduces greatly the network load with 0.0008 ms/byte, versus the replicated system with 0.0044 ms/byte.

On the other hand, the clustered system seems sensitive to the changes in the topics of the queries through time. For the queries of the year 1999, the performance is nearly the same, 8068 ms per query,

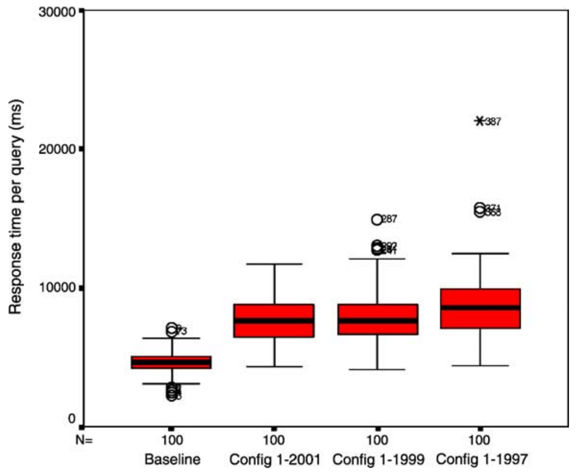

Fig. 10. Clustered IR systems vs. a replicated IR system. Configuration 1: 128 query servers.

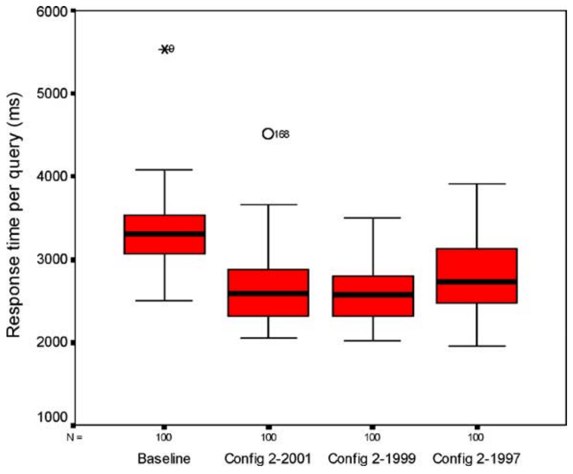

Fig. 11. Clustered IR systems vs. a replicated IR system. Configuration 2: 1024 query servers.

while for the queries from 1997 the performance drops to 9212 ms per query. In fact, the higher differences for the topic distributions are between the years 2001 and 1997 (see Table 7).

Note that the presence of atypical data reflects the effect of the changes in the topics through the time. In fact, the baseline also happens to have some atypical data but very near the confidence interval, due to the distribution of the queries over many servers. In a clustered system, with a reduced amount of hosts per cluster, some heavy queries will produce higher response times. This is more notable when the topics of the queries are changed (1997 queries), because the smaller clusters may have to cope with a higher number of queries than initially expected.

In the second configuration, the clustered system and the replicated system have processed the queries matching the 2001, 1999 and 1997 distributions. The response times for the clustered system with 1024 query servers are labeled "*Config 2-2001*", "*Config 2-1999*" and "*Config 2-1997*", respectively. Fig. 11 shows the box diagram for the response time for the first 100 queries processed for all these systems.

In this case, the clustered system outperforms the baseline, for all query distributions, through the years. The replicated system requires 3313 ms per query, while the clustered system for the 2001 queries will process one query in 2665 ms on average. Regarding the network load, while the replicated system needs, on average, 0.112 ms to send one byte, the clustered system uses only 0.007 ms per byte, on average.

This configuration is also sensitive to the changes in the topics of the queries, but to a smaller degree. For the queries from 1999, the performance is slightly better, 2630 ms per query, but for the queries from 1997, the performance drops to 2938 ms per query (still outperforming the baseline).

In this configuration, the increase in the number of query servers reduces the size of the local indexes (even for the smaller clusters) and therefore, the increase in the response times is less significant. At the same time, the different clusters can support more easily the changes in the query topics through the time. In this configuration, for the 1997 queries, the performance decreases by 9%, while with 128 query servers the throughput decreased by 14%.

5. Conclusions

In this paper, we have described different architectures for a distributed IR system, analysing the optimal design (i.e. number of brokers) and estimating the maximum performance achieved with multiple configurations (from 1 up to 4096 query servers). We have studied the performance of a distributed, replicated and clustered system, and we have established the bottlenecks and limitations of each possible configuration. Our work extends previous works with respect to both the size of the collection and the number of query servers simulated, and confirms some of the previous findings in the literature (see Section 2).

Indeed, we have identified two main bottlenecks in a distributed and replicated IR system: the brokers and the network. The load on the brokers is mainly due to the number of local answer sets to be sorted (characteristic of a distributed system), as it was also shown by Cahoon and McKinley (1996). Therefore, the load can be improved by reducing the number of documents included in the local answer sets by all the query servers, which could affect precision and recall. Another way is to reduce the number of local lists sent to the brokers, by designing more complex and elaborate distributed protocols.

The network bottleneck is due to the large number of query servers and the continuous data interchange with the brokers, especially in a replicated IR system. The network load could be alleviated by reducing the number of bytes sent over the network or by using different network configurations and technologies. The network traffic can be limited by reducing the number of results in each local answer set (with the additional benefit over the brokers), or by compressing the local answer set before sending it. The latter was tested by compressing the local answer sets up to a 10%. In this case, the network load was substantially reduced and therefore, the performance obtained by the replicated system was nearly optimal, as described in Fig. 9. Moreover, it is important to recall that we have modeled the network as a single LAN. This could have an impact on the obtained performance measures. In the future, we intend to use a more realistic network model for our distributed IR system.

The analysis of the clustered systems indicates that the best throughput of these systems is achieved when a large number of query servers is used, i.e. greater than 1000 in total. In this case, the clustered system outperforms the replicated one. A clustered system will reduce the network load substantially, as only a fraction of the query servers will process and answer each query. Therefore, in a replicated system, the network load increases (and the throughput improvements are slowed) as the number of servers increases. While in a clustered system the processing times in the clustered query servers could be slightly higher, the local answers will reach faster the broker, and the brokers will receive fewer answers, thus processing the final results more efficiently.

However, the clustered systems must be configured a priori, depending on the distribution of the queries that the IR system would receive. Therefore, to avoid negative effects on the performance, it is important to detect changes in the distribution of the queries through the time and re-configure the clusters of the system accordingly.

We also plan to study different solutions for the brokers and network bottlenecks, as well as their impact on the retrieval performance. Moreover, these results could be used to extend a basic centralised IR system to a distributed one, and then analyse the correspondence between the expected and the actual performance. In general, we believe that the results in this paper are useful to any group interested in handling a very large collection like SPIRIT, or building a large-scale search engine.

Acknowledgments

The work of the first author has been partially supported by the Comisión Interminis-terial de Ciencia y Tecnología (CICYT) of the Spanish government, under project TIC2001-0547 and by the Fundación Caixa Galicia (Beca curso 2002/2003 para Estudios de Postgrado en Universidades y en Centros de Investigación de Excelencia Académica).

The work of the second and third authors is funded by a UK Engineering and Physical Sciences Research Council (EPSRC) project grant, number GR/R90543/01. The project funds the development of the Terrier Information Retrieval framework (url: http://ir.dcs.gla.ac.uk/terrier).

We would like to thank Mark Sanderson and Hideo Joho for giving us access to the 1 TB dataset used for the SPIRIT Project. We would also like to thank Prof. David Harper for his important comments and suggestions.

References

Bailey, P., Craswell, N., & Hawking, D. (2003). Engineering a multi-purpose test collection for web retrieval experiments. *Information Processing and Management, 39*(6), 853–872.

Burkowski, F. J. (1990). Retrieval performance of a distributed database utilising a parallel process document server. In *Proceedings of the second international symposium on databases in parallel and distributed systems* (pp. 71–79). New York: ACM Press.

Cahoon, B., & McKinley, K. S. (1996). Performance evaluation of a distributed architecture for information retrieval. In *Proceedings of 19th ACM-SIGIR international conference on research and development in information retrieval* (pp. 110–118). New York: ACM Press.

Callan, J. (2000). Distributed information retrieval. In W. B. Croft (Ed.), *Advances in information retrieval: recent research from the CIIR* (pp. 127–150). Kluwer Academic Publishers.

Coevreur, T. R., Benzel, R. N., Miller, S. F., Zeitler, D. N., Lee, D. L., Singhal, M., Shivaratri, N., & Wong, W. Y. P. (1994). An analysis of performance and cost factors in searching large text databases using parallel search systems. *Journal of the American Society for Information Science, 45*(7), 443–464.

Google, (2004). Homepage. Google Inc. Retrieved 21 April, 2004. Available: http://www.google.com/.

Harman, D., Mccoy, W., Toense, R., & Candela, C. (1997). Prototyping a distributed information retrieval system that uses statistical ranking. *Information Processing and Management, 27*(5), 449–460.

Hawking, D. (1997). Scalable text retrieval for large digital libraries. *Lecture Notes in Computer Science, 1324*, 127–146.

Hawking, D., & Craswell, N., (2001). Overview of the TREC-2001 Web track. In E. M. Voorhees, & D. K. Harman (Ed.), *Proceedings of the tenth text retrieval conference* 500-250 (pp. 61–67). Gaithersburg, Maryland: NIST Special Publication.

Hawking, D., & Thistlewaite, P. (1999). Methods for information server selection. *ACM Transactions on Information Systems, 17*(1), 40–76.

Heaps, H. S. (1978). *Information retrieval: computational and theoretical aspects*. New York: Academic Press.

Jeong, B., & Omiecinski, E. (1995). Inverted file partitioning schemes in multiple disk systems. *IEEE Transactions on Parallel and Distributed Systems, 6*(2), 142–153.

Jones, C. B., Purves, R., Ruas, A., Sanderson, M., Sester, M., van Kreveld, M., & Weibel, R. (2002). Spatial information retrieval and geographical ontologies an overview of the SPIRIT project. In *Proceedings of the 25th ACM-SIGIR conference on research and development in information retrieval* (pp. 387–388). New York: ACM Press.

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

Little, M. C., (2001). *JavaSim User's Guide. Public Release 0.3, Version 1.0*. University of Newcastle upon Tyne. Retrieved 1 June, 2003. Available: http://javasim.ncl.ac.uk/manual/javasim.pdf.

Lu, Z., & McKinley, K. (2000). Partial collection replication versus caching for information retrieval systems. In *Proceedings of the 25th ACM-SIGIR conference on research and development in information retrieval* (pp. 248–255). New York: ACM Press.

Martin, P., Macleod, I. A., & Nordin, B. (1986). A design of a distributed full text retrieval systems. In *Proceedings of the 9th ACM-SIGIR conference on research and development in information retrieval* (pp. 131–137). New York: ACM Press.

Plachouras, V., Ounis, I., Amati, G., & van Rijsbergen, C. J., (2002). University of Glasgow at the Web track of TREC 2002. In E. M. Voorhees, & L. P. Buckland (Ed.), *Proceedings of the eleventh text retrieval conference* 500-251 (pp. 645–651). Gaithersburg, Maryland: NIST Special Publication.

Ribeiro-Neto, B., & Barbosa, R. (1998). Query performance for tightly coupled distributed digital libraries. In *Proceedings of the 3rd ACM conference on digital libraries* (pp. 182–190). New York: ACM Press.

Risvik, K., & Michelsen, R. (2002). Search engines and web dynamics. *Computer Networks, 39*, 289–302.

Spink, A., Jansen, B. J., Wolfram, D., & Saracevic, T. (2002). From e-sex to e-commerce: Web search changes. *IEEE Computer, 35*(3), 107–109.

Tomasic, A., & Garcia-Molina, H. (1993). Performance of inverted indices in shared-nothing distributed text document information retrieval systems. In *Proceedings of the 2nd international conference on parallel and distributed information systems* (pp. 8–17). San Diego, California: IEEE Computer Society.