Performance Analysis of Distributed Architectures to Index One Terabyte of Text

Fidel Cacheda[1], Vassilis Plachouras[2], and Iadh Ounis[2]

[1] Departament of Information and Communication Technologies, University of A Coruña
Facultad de Informática, Campus de Elviña s/n, 15071 A Coruña, Spain
fidel@udc.es
[2] Department of Computing Science, University of Glasgow
Glasgow, G12 8QQ, UK
{vassilis, ounis}@dcs.gla.ac.uk

Abstract. We simulate different architectures of a distributed Information Retrieval system on a very large Web collection, in order to work out the optimal setting for a particular set of resources. We analyse the effectiveness of a distributed, replicated and clustered architecture using a variable number of workstations. A collection of approximately 94 million documents and 1 terabyte of text is used to test the performance of the different architectures. We show that in a purely distributed architecture, 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 large number of query servers is used, mainly due to the reduction of the network load.

1 Introduction

Retrieval systems based on a single centralized index are subject to several limitations: lack of scalability, server overloading and failures [6]. Therefore, given these facts, it seems more appropriate to turn to the distributed Information Retrieval (IR) systems 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. 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 [13]). 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 [13]). The study in [15] indicates that the local inverted file organization uses system resources effectively, provides good query throughput and is more resilient to failures.

From the database point of view, a distributed information retrieval system could follow a single database model or a multi-database model [4]. In the single database model, the documents are copied to a centralized database, where they are indexed and made searchable. In a multi-database model, the existence of multiple text

S. McDonald and J. Tait (Eds.): ECIR 2004, LNCS 2997, pp. 394–408, 2004.

databases is considered explicitly, and at the same time, each database could have the inverted index distributed.

This work is a case study of different architectures for a distributed information retrieval system on a very large Web collection. The SPIRIT collection (94,552,870 documents and 1 terabyte (TB) of text) [10] 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 timings obtained depend on the specific system simulated [1], the trends and conclusions should be independent of the system used. In this way, our study works out the required resources and the best architecture to handle a very large collection of data, like SPIRIT. We believe that this work is a step along the recent trend in building very large collections for Web IR, like the TREC Terabyte track[1] initiative.

The improvements in the performance of a single database model are examined in a distributed and replicated system. Also, the effects of a multi-database model are tested through a clustered system.

We start by presenting the related work. In Section 3, we describe the simulation models used (analytical, collection and distributed models). Next, we describe the simulations performed for the different architectures: distributed, replicated and clustered system, and the results obtained. Finally, the main conclusions are presented.

2 Related Work

The work on architecture performance is the most directly related to this paper. Several articles [2], [5], [12] analyze the performance of a distributed IR system using collections of different sizes and different system architectures. Cahoon and McKinley in [3] describe the result of simulated experiments on the distributed INQUERY architecture. Using the observed behaviour for a mono-server implementation, they derive the performance figures for a distributed implementation, proving it to be scalable.

The previous work for distributing the inverted index over a collection of servers is focused on the local and global inverted files strategies [13], [15], 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 performance evaluation of several distributed architectures using a massive cluster of workstations (up to 4096) and identifying the limitations of each model. The novelty of this work relies on the size of the collection represented (1TB of text) and the large number of workstations simulated. This work is especially related to [3] and [13], but it differs mainly in three points. First, the probabilistic model is considered and therefore, disjunctive queries are used in the system (without the reduction in the answer set provided by the conjunctive operations). Second, a simple analytical model is developed initially for a single-database/single-server environment (similarly to [13]), and this will be the basis for the simulation of the distributed IR systems, composed of multiple query servers (similarly to [3]). Third, initially the results of the analytical model are tested using

[1] http://www-nlpir.nist.gov/projects/terabyte/

the TREC WT10g collection and the set of the topic relevance queries from TREC10 [7]. Next, a document collection of 1TB and its queries are modelled 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 [11].

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 its set of real queries 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 1TB 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 [1], using the WT10g collection and the set of queries used for TREC10 [7]. This analytical model is similar to the one described in [13].

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.
- tc_2: second coefficient for the time to compare two identifiers and swap them.
- ti: initialisation time, including memory allocation and output display.
- ts: average seek time for a single disk.
- tr: average time to read the information about one document in an inverted list and do its processing (seek time is excluded).
- t_i: total time (in milliseconds) to complete processing of the 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. Previous works [13] have modelled this using a linear relationship, but from our experiments, a logarithmic model seems to fit more accurately as the number of results increases (coefficients tc_1 and tc_2). Hence, the time to merge and sort n results

(tc) is calculated as: $tc = tc_1 \times n + tc_2 \times \ln(n)$ (versus the linear model used in [13] $tc = tc_1 \times n$).

The processing of a query is divided into four phases: an initialization phase, seeking disks, reading the inverted lists from disk and assigning weights to documents, and obtaining the answer and ranking the results. Therefore, the processing time for a query q_i is t_i, given by:

$$t_i = ti + k_i \times ts + \sum_{k \in q_i} d_k \times tr + tc \times r_i \ .$$

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 [1] in order to obtain the exact number of results for each query and the exact number of documents associated with each of the inverted lists retrieved.

The accuracy of the analytical model was confirmed comparing the response times of the real IR system for the queries number 501 to 550 from the WT10g collection (the parameter values used are: ti=1400ms, ts=0.03ms, tr=4.0208μs, tc_1=0.000131, tc_2=0.000096; q_i, k_i, d_k, r_i depend on the collection modelled or simulated, as described in the next section).

3.2 The Spirit Collection Model

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 1TB of text, although no queries and relevant assessments exist for the moment [10]. 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.

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 the definition for the parameters considered.

Table 1. Parameters for the document model. The real values were obtained from a SPIRIT subcollection and the estimated values represent the whole SPIRIT collection

Parameter	Real values	Estimated values	Description
D	1,221,034	94,552,870	The number of documents
W	456	456	Average words per document
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)

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. V represents the vocabulary, where each word is uniquely identified by an integer w in the range $1 \leq w \leq T$, 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 normalized to a probability distribution. The regression analysis performed confirms that the quadratic model fits better the real distribution ($R = 0.99770$) versus the linear model representing the Zipf's law ($R = 0.98122$). The quadratic model is similar to Zipf's, although in previous works [15], 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, divided by a normalisation constant [15].

The third column of Table 1 shows the values for the parameters of 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 was chosen.

The size of the vocabulary of a collection of documents matches the Heaps law [8], with $K = 4.60363$ and $\beta = 0.6776$. Therefore, an approximation of 73,689,638 unique terms for the whole collection is obtained.

Finally, a different probability distribution is provided for the whole collection. Given the quadratic fit curve previously described, a new normalization constant is defined for the new vocabulary size ($\Theta = 4.294476 \times 10^8$).

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 $Q(t)$. Actually, in our simulation study the number of terms is selected uniformly between 1 and 4 terms per query. In Table 2 a description of each parameter and the base values chosen are presented.

The most realistic query model is the skewed query model [9], where Q is modelled assuming that the probability of a term occurring in a query is proportional to that term's frequency in vocabulary. The probability distribution $Q(t)$ for the skewed query models is: (where C represents a normalization constant)

$$Q(t) = \begin{cases} C \times Z(t) & \text{if } sT \leq t \leq (u-s)T \\ 0 & \text{otherwise} \end{cases} \qquad \text{where } 1 = \sum_{i=sT}^{(u-s)T} C \times Z(i) \text{ and } u \geq 2s\,.$$

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 high rank increases. Words of high rank are highly repeated 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 [9]. 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. In all our simulations the values for these parameters are: $u = 0.01$ and $s = 0.00007$.

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 used in the query terms
s		The fraction of T skipped for the query terms

At certain points in the simulation, we will need to know the expected size of an inverted list and the expected size of an answer set. 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. So, the number of documents of an inverted list for term t_i will be [15]: $Documents \times [1 - (1 - Z(t_i))^W]$.

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

$$Documents \times [1 - (1 - Z(t_1))^W \times \ldots \times (1 - Z(t_k))^W].$$

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. The results showed that the simulations using the skewed query model produce on average similar response times as the real queries. Although the fluctuations of the real queries are not present in the skewed query model, this model can be considered quite accurate for the average response times.

3.3 Distributed Model

In a distributed IR system, the queries are stored on a global queue, which is controlled by one or more *central brokers*. Each broker will take one query and will send it to all the query servers through a network, in a local index organization [13]. 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.

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 (by default, 50 queries).

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

Table 3. Parameters for the distributed model

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

$d_{k,j}$: 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 (we consider the top 1000 documents only).

$tr_{i,j}$: 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 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.

Therefore, the time for the query server j to process the query q_i is given by:

$$t_{i,j} = rq_{i,j} + ti + k_i \times ts + \sum_{k \in q_i} d_{k,j} \times tr + tc \times r_{i,j} .$$

The parameters $d_{k,j}$ and $r_{i,j}$ are estimated through the collection model parameters d_k and r_i respectively, 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:

$$t_i = \max(t_{i,j}) + \max(ra_{i,j}) + \sum_j tr_{i,j} \times tc .$$

The problem is that the parameters $rq_{i,j}$ and $ra_{i,j}$ can not be estimated using an analytical model as they depend directly 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:

$$LANOverhead + RequestLength \times \left(LANBandwitdh / 8\right)^{-1} \times 1000 .$$

The values and description for the parameters used in the simulation of the network are described in Table 3. The *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 for query q_i set is sent from query server j to the broker, then the length of the packet will be: $tr_{i,j} \times DocAnswerSetSize$.

4 Simulation Results

This section describes the results of several experiments developed using the simulation model described in the previous section. The objective is to determine different approaches for the distribution and replication of the collection using a bunch of query servers and compare the performance between the different configurations.

All the simulations are based on the 1TB SPIRIT collection model [10]. The queries have been modelled using the skewed query model and following a worst case scenario: each query will retrieve on average approximately 8.4 million documents (9% of the whole collection). A batch of 50 queries is used to test the performance, and for each different configuration, 5 different simulations (with distinct initial seeds) are run, and the average values for the execution times are calculated for each query.

Initially, a purely distributed system is examined. Next the effects of the replications are analyzed and then, we examine possible configurations of a clustered system (based on an asymmetric distribution and replications).

4.1 Distributed System

In this set of experiments, the collection of documents is distributed using the local index organization over *N* query servers, where *N = 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 768 and 1024*. Initially, the number of brokers for the distributed IR system is set to one, and next is increased to 2, 3 and 4. The results are displayed in Fig. 1.

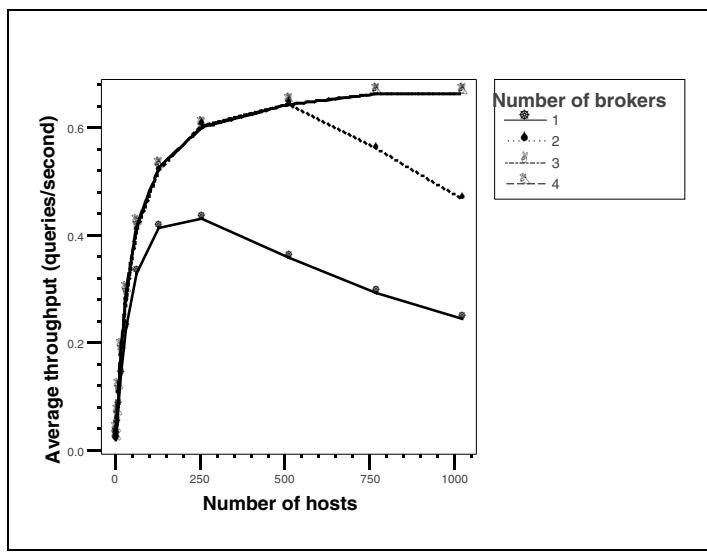

Fig. 1. Throughput for the simulation of a distributed IR system with local index organization

Table 4. Estimated time (mm:ss) to process 50 queries by the distributed system (3 brokers)

Query servers	Time	Query servers	Time
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

The optimal performance is achieved when two or more brokers are used (see Fig. 1). In fact, with less than 512 query servers, two brokers are able to provide continuously queries to the servers, and so, 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 benefit is obtained if the number of brokers is increased.

The bottleneck in the brokers is due to the number of local answer sets received from all the query servers that must be sorted. Therefore, increasing the number of query servers will benefit the processing time in the query servers as each server reduces the size of its index. On the other hand, the brokers will receive more local answer sets to be merged in the final result set. In fact, if the number of query servers is high enough, the performance will start descending at a certain point, independently of the number of brokers used.

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

Without any improvements, the throughput tends to be stabilised around 0.64 queries/second with 512 query servers, with minor improvements as the number of servers increases (0.66 queries/second with 1024 query servers).

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

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

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

Initially, a two replications system is simulated, testing different 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). While, with five brokers the maximum throughput is achieved, an increase in the number of brokers will slightly increase 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, 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 servers.

Generally, for the configurations simulated, the number of brokers necessary to operate 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 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/minute, then the expected throughput for a system with R replicas will be $T*R$. This is coherent with the results obtained in Table 5, especially considering 128 or less 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 future works, some other distribution policies can be analysed in order to improve the throughput up to the optimal theoretical value, similar to the one used in [12].

Note that if more than 256 query servers are used per replica, the performance of the system starts to decrease rapidly. If the total number of query servers in the system (considering all the replicas) is beneath 1000, the performance is improved with each new replica added. However, if the number of query servers is over this limit, the performance decreases, especially as more replicas are included in the system. In fact, a system with 4 replicas of 1024 query servers has a worse throughput than a system with 4 replicas of 64 servers each.

This loss of performance is due to the network. Each replication adds more hosts to the network, which is used intensively to send the results back to the brokers. As a consequence, the network latency is greatly increased with each new replica added, converting the network to the 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, the query servers and the brokers.

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 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).

Different commercial Web IR systems claim to use a clustered system adapted to the distributions of the queries received (e.g. AllTheWeb). 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. [14], a set of real queries of Web users is categorized into 11 different topics. Moreover, the variations in the percentage of queries for each topic are analyzed in three different years: 2001, 1999 and 1997. Table 6 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, the number of queries is increased to 200 in order to examine the whole range of topics, and 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 clusters defined will index the same number of documents, although using a different number of servers.

Table 6. 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
Goverment	2.054 %	1.82 %	3.13 %	3 * 1	21 * 1
Arts	1.155 %	1.33 %	5.14 %	2 * 1	12 * 1

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. 1, with an optimal configuration of two brokers.

Roughly speaking, 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 should receive (see Table 6).

For the first configuration, the baseline is a replicated IR system, with 4 replications of 32 query servers each. On the other side, the clustered system is configured in accordance with the distribution of the topics on the year 2001. In Table 6, the column "*Config 1*" describes the query servers assigned to each topic. The first number represents the number of the distributed query servers, and the second, the number of replicas in each cluster.

Figure 2 presents the box diagram for the response time for the first 100 queries processed by the tested systems. 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 labelled "*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 on 4682 milliseconds, while the clustered system optimally configured for the 2001 queries will just process one query in 7806 milliseconds (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, vs. 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 the time. For the queries of the year 1999, the performance is nearly the same, 8068 milliseconds per query, while for the 1997, queries the performance drops to 9212 milliseconds per query. In fact, the higher differences for the topic distributions are between the years 2001 and 1997 (see Table 6).

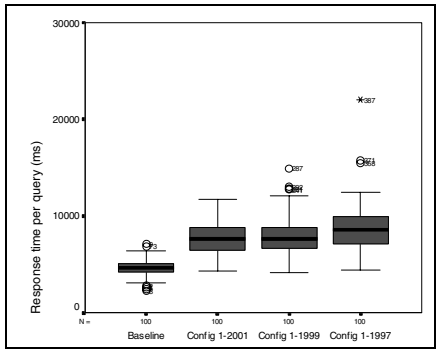

Fig. 2. Clustered IR system vs. a replicated IR system. Configuration 1: 128 query servers

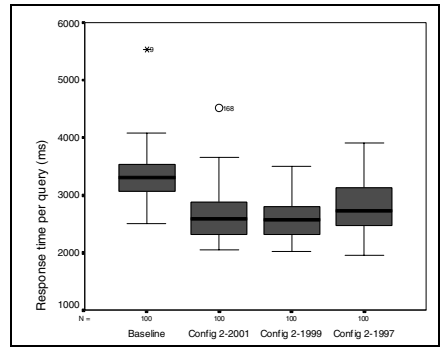

Fig. 3. Clustered IR system vs. a replicated IR system. Configuration 2: 1024 query servers

Also, 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 1024 query servers of the clustered system are assigned according to the values reflected in Table 6, on the column labelled "*Config 2*". The baseline in this case, is a replicated system with 4 replicas of 256 query servers each. As for the previous case, the clustered system and the replicated system have processed the queries matching the 2001, 1999 and 1997 distributions. Fig. 3 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 the years of the query distributions. The replicated system has a performance of 3313 milliseconds per query, while the clustered system for the 2001 queries will process one query in 2665 milliseconds on average. Regarding the network load, while the replicated system needs, on average, 0.112 milliseconds to send one byte, the clustered system uses only 0.007 milliseconds 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 1999 queries, the performance is slightly better, 2630 milliseconds per query, but with the 1997 queries the performance drops to 2938 milliseconds per query (still outperforming the baseline).

In this configuration, the increase in the number of query servers augments the distribution of the local indexes 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, analyzing the optimal design and estimating the maximum performance achieved with multiple configurations (from 1 up to 4096 query servers). We have studied the performance of a simulated distributed, replicated and clustered system on a very large Web collection, and we have established the bottlenecks and limitations of each possible configuration.

Two main bottlenecks have been identified 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). Therefore, the load can be improved by reducing the number of documents included in the local answer sets by all the query servers, which can affect the precision and recall parameters. 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 high number of query servers and the continuous data interchange with the brokers, especially in a replicated IR system. The traffic over the network can be limited by reducing the number of results in each local answer set (with the additional benefit over the brokers) or compressing the local answer set before sending it.

The analysis of the clustered systems indicates that the best throughput of these systems is achieved when a great number of query servers is used, outperforming a replicated system. A clustered system will reduce greatly the network load 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, processing the final results more efficiently.

However, the clustered systems must be configured a-priori based on the distribution of the queries that the IR system will 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.

In the future, we plan to study different solutions for the brokers and network bottlenecks (e.g. distributing the brokers) and their implications in the retrieval performance. Also, these results will be used to extend the basic actual IR system to a distributed system, and in general, we believe that the results in this paper are useful to any group interested in indexing a very large collection like SPIRIT.

Acknowledgements. The work of the first author has been partially supported by Spanish CICYT, under project TIC2001-0547 and by the Fundación Caixa Galicia (grant 2002/2003 postgraduate studies).

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 also like to thank Mark Sanderson and Hideo Joho for giving us access to the 1TB dataset used for the SPIRIT Project.

References

1. Amati, G., Carpineto, C., Romano, G.: FUB at TREC-10 Web track: A probabilistic framework for topic relevance term weighting. In NIST Special Publication 500-250: The Tenth Text REtrieval Conference (TREC-2001). 2001.
2. Burkowski, F. J.: Retrieval performance of a distributed database utilizing a parallel process document server. In Proceedings of the International Symposium on Databases in Parallel and Distributed Systems, pp: 71-70. 1990.
3. Cahoon, B., McKinley, K.S.: Performance evaluation of a distributed architecture for information retrieval. In Proceedings of ACM-SIGIR International Conference on Research and Development in Information Retrieval, pp: 110-118. 1996.
4. Callan, J.: Distributed information retrieval. In W. Bruce Croft, editor, Advances in Information Retrieval: Recent Research from the CIIR, chapter 5, pp: 127-150. Kluwer Academic Publishers, 2000.
5. Hawking, D.: Scalable text retrieval for large digital libraries. In Proceedings of the 1st European Conference on Research and Advanced Technology for Digital Libraries, Springer LNCS, vol. 1324, pp: 127-146. 1997.
6. Hawking, D., Thistlewaite, P.: Methods for Information Server Selection. ACM Transactions on Information Systems, 17(1), pp: 40-76. 1999.
7. Hawking, D., Craswell, N.: Overview of the TREC-2001 Web Track. In: Information Technology: The Tenth Text Retrieval Conference, TREC 2001. NIST SP 500-250. pp.61-67.
8. Heaps, H. S.: Information Retrieval: Computational and Theoretical Aspects. Academic Press, New York, 1978.
9. Jeong, B., Omiecinski, E.: Inverted File Partitioning Schemes in Multiple Disk Systems. IEEE Transactions on Parallel and Distributed Systems, 6(2):142-153. 1995.
10. Jones, C.B., Purves, R., Ruas, A., Sanderson, M., Sester, M., van Kreveld, M., Weibel, R.: 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. ACM Press, 2002.
11. Little, M. C.: JavaSim User's Guide. Public Release 0.3, Version 1.0. http://javasim.ncl.ac.uk/manual/javasim.pdf, University of Newcastle upon Tyne, 2001.
12. Lu, Z., McKinley, K.: Partial collection replication versus caching for information retrieval systems. In Proceedings of the ACM International Conference on Research and Development in Information Retrieval, pp: 248-255. 2000.
13. Ribeiro-Neto, B. Barbosa, R.: Query performance for tightly coupled distributed digital libraries. In Proceedings of the 3rd ACM Conference on Digital Libraries, pp: 182-190. 1998.
14. Spink, A., Jansen, B. J., Wolfram, D., Saracevic, T.: From e-sex to e-commerce: Web search changes. IEEE Computer 35(3): 107-109. 2002.
15. Tomasic, Al, Garcia-Molina, H.: 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, 1993.