Analyzing Imbalance among Homogeneous Index Servers in a Web Search System

C.S. Badue [a,*], R. Baeza-Yates [b], B. Ribeiro-Neto [a,c], A. Ziviani [d], N. Ziviani [a]

[a] *Department of Computer Science, Federal University of Minas Gerais, Belo Horizonte, Brazil*

[b] *Yahoo! Research Barcelona, Spain & Santiago, Chile*

[c] *Google Engineering Belo Horizonte Belo Horizonte, Brazil*

[d] *National Laboratory for Scientific Computing (LNCC), Petrópolis, Brazil*

Abstract

The performance of parallel query processing in a cluster of index servers is crucial for modern web search systems. In such a scenario, the response time basically depends on the execution time of the slowest server to generate a partial ranked answer. Previous approaches investigate performance issues in this context using simulation, analytical modeling, experimentation, or a combination of them. Nevertheless, these approaches simply assume balanced execution times among homogeneous servers (by uniformly distributing the document collection among them, for instance)—a scenario that we did not observe in our experimentation. On the contrary, we found that even with a balanced distribution of the document collection among index servers, correlations between the frequency of a term in the query log and the size of its corresponding inverted list lead to imbalances in query execution times at these same servers, because these correlations affect disk caching behavior. Further, the relative sizes of the main memory at each server (with regard to disk space usage) and the number of servers participating in the parallel query processing also affect imbalance of local query execution times. These are relevant findings that have not been reported before and that, we understand, are of interest to the research community.

Key words: parallel query processing, imbalance, search engines, performance analysis

1 Introduction

The tremendous success of keyword targeted advertising has fuelled Web economy to new heights. In fact, in 2004 American companies spent between 9 and 10 billion dollars in online advertising to promote their products and services (Jupiter Media, 2005). Around 40-50% of this amount is estimated to have been spent on online advertising associated with answers to user queries. As a consequence, search engines are currently a critical component of Web economy.

Additionally, fairly recent announcements of new products by major players such as Google and Yahoo indicate a rising interest in corporate search, i.e., installing search engines in the Intranets of large companies to sift through the vast amounts of data produced internally. In this case, the interest of the corporation is on compiling and organizing the information it generates regarding its own business, using information on the business as a de facto asset. Again, search engines are expected to play a major role in this context.

When a user query reaches a search engine, the query processing is split into two consecutive major phases (Barroso et al., 2003). The first phase consists of retrieving from the storage devices documents that contain each query term, executing a conjunction of the sets of these documents, and finally ranking the selected documents using some relevance metric. Search engines that deal with huge document collections perform a partial evaluation in this first phase instead of a full one. The second phase consists of taking the top ranked answers of the first phase, typically 10, and generating snippets, title, and URL information for each of them. For this, the search engine needs to examine the full texts of the top ranked documents. Both phases play important roles in the performance of modern search engines. Nevertheless, as the size of a search engine increases, the processing cost of the second phase remains basically constant, whereas the processing cost of the first phase increases—larger lists of documents have to be read from the disks and processed. Therefore, the performance of the first phase is crucial for maintaining the scalability of modern search engines that deal with an ever-increasing amount of Web documents.

In this work, we analyze the performance of the first phase of the query pro-

* Corresponding author address: Department of Computer Science, Federal University of Minas Gerais, Av. Antônio Carlos 6627, 31.270-010, Belo Horizonte, Brazil; Phone: +55 31 3499 5860; Fax: +55 31 3499 5858.

Email addresses: claudine@dcc.ufmg.br (C.S. Badue), ricardo.baeza@upf.edu,rbaeza@dcc.uchile.cl (R. Baeza-Yates), berthier@dcc.ufmg.br,berthier@google.com (B. Ribeiro-Neto), ziviani@lncc.br (A. Ziviani), nivio@dcc.ufmg.br (N. Ziviani).

cessing task, i.e., the retrieval of the most relevant documents for a given user query. In our architecture, the whole collection of documents is partitioned among a set of index servers organized in a computational cluster, such that each server stores its own local subcollection. Upon the arrival of a given user query, the task of retrieving the most relevant documents for this query is then shared among the index servers so that each index server performs the retrieving task for the query only on its partition of the document collection. A broker is responsible for merging the partial ranked answers from the servers to produce the final ranked answer.

In this architecture for parallel query processing, characterized by a local partitioning of the document collection, the response time of a query is determined by the execution time of the slowest index server. As a consequence, imbalance in execution times among index servers increases the response time of a query executed by the cluster of servers. Therefore, it is critically important to avoid imbalance among index servers if higher performance is to be achieved.

A common counter-measure against imbalance is to distribute the whole collection of documents among homogeneous index servers in a balanced way, such that each server handles a similar amount of data for processing any given query. At a first glance, as a consequence of having similar data volumes handled at each server for a given query, one would expect that execution times at the homogeneous index servers would also be approximately balanced. Indeed, this idealized scenario of balanced execution times is a usual assumption taken by theoretical models for web search systems (Tomasic and Garcia-Molina, 1993; Ribeiro-Neto and Barbosa, 1998; Cacheda et al., 2004; Chowdhury and Pass, 2003). However, in a real case scenario, correlations between term frequencies in the query log and the sizes of the corresponding inverted lists lead to imbalances in query execution times.

In this paper, we carefully investigate and analyze the imbalance issue in a computational cluster composed of homogeneous index servers. As a major contribution, we verify that the idealized scenario of balanced execution times at homogeneous index servers with similar data volumes is unlikely to be found in practice. Our results are derived from experiments in an information retrieval testbed fed with real data obtained from a real-world search engine. This is an important experimental result because our findings shed light on a usual assumption that is obliviously taken as valid by previous theoretical models, whereas imbalance masks possibilities for performance improvements. Moreover, we identify and fully analyze the main sources of imbalance: the use of disk caching, the size of main memory in the homogeneous index servers, and the number of servers in the cluster.

This paper is organized as follows. Section 2 discusses research related to ours. Section 3 presents the architecture of our system, the cluster of index servers,

the index organization, and the parallel query processing technique. Section 4 characterizes imbalance in the execution times of homogeneous index servers, describing the workload used in the experimental analysis, our experimental setup, the sources for the verified imbalance, and finally the execution times of queries in our query collection for our cluster of homogeneous index servers. Our conclusions follow in Section 5.

2 Related Work

Tomasic and Garcia-Molina (1993) compare the performance impact on query processing of various physical organizations for inverted lists. Based on results derived from simulation, they show that the index organization depends heavily on the access time of the storage device and the communication bandwidth. Ribeiro-Neto and Barbosa (1998) study how query performance is affected by the index organization, the network capacity, and the disk transfer rates, using a simple analytical model coupled with a small simulator. Cacheda et al. (2004) present a case study of different architectures for a distributed information retrieval (IR) system, in order to provide a guide to approximate the optimal architecture with a specific set of resources. Using a simulator based on an analytical model for query processing similar to the one described in (Ribeiro-Neto and Barbosa, 1998), they identify two main bottlenecks in a distributed and replicated IR system: the brokers and the network. These simulation-oriented performance studies relate to ours in identifying the disk access times as a key factor for query performance. Nevertheless, the analytical models presented therein assume that execution times are balanced if index servers manage a similar amount of data when processing a query.

Chowdhury and Pass (2003) introduce an approach based on queueing theory for modeling and analyzing architectures for search systems in terms of their operational requirements: throughput, response time, and utilization. To model the service time of an index server, they compute the mean service time as a function of the number of documents in the index, using a set of queries against their search index. If the search index has n documents, then the mean service time of a single index server is equal to $f(n)$; if the collection is partitioned into 2 pieces, each one with $n/2$ documents, then the mean service time for the 2 index servers is equal to $f(n/2)$. Thus, their queueing model also assumes a perfect balance among the execution times of index servers that process an equal number of documents per query.

MacFarlane et al. (2000) and Badue et al. (2001) investigate the performance impact on parallel query processing of two distinct types of index organizations (term or document partitioning) using a real case implementation. They conclude that performance of parallel query processing in a cluster of index

servers is impacted by disk access time, network communication time, and query concurrency level. Orlando et al. (2001) present the architecture of a parallel and distributed search engine on which they explore two main parallelization strategies: a task parallel strategy, by which queries are executed independently by a set of homogeneous index servers, and a data parallel strategy, that we refer to as a document partitioning strategy, by which each query is processed in parallel by index servers accessing distinct partitions of the database. They have conducted real experiments that highlighted the better performance of a hybrid task plus data parallelization due to a good exploitation of memory hierarchies, in particular of the buffer cache which virtualizes the access to the disk-resident posting lists. This analytical work presents a similar analysis to ours in terms of identifying the disk caching operations as a factor for accelerating disk access times. In fact, these approaches inherently consider any imbalance on execution times among index servers in their results because these results are derived from real experimentation. Nevertheless, these experimentation-based approaches fail to be aware of the imbalance and to characterize its impact on the performance of a web search system, as we do in this paper.

Based on these considerations, we observe that no matter the previous approach—be it based on simulation, modeling, or experimentation—, they all fail in being aware of any imbalance in execution times. This happens because either they simply assume balanced execution times among the homogeneous servers if the data collection is distributed in a balanced way or they obliviously take the imbalance into account from real experiments, failing then to characterize such an imbalance. In contrast, we verify in this paper that this idealized scenario of balanced execution times among homogeneous index servers with similar data volumes is unlikely to be found in practice.

3 Architecture

In this section, we present the architecture of our system, the cluster of index servers, the index organization, and the parallel query processing technique.

3.1 Cluster of Index Servers

Modern search engines typically rely on computational clusters for query processing (Barroso et al., 2003; Risvik et al., 2003). Such clusters are composed of a single broker and p index servers. The broker receives user queries from client nodes and forwards them to the index servers, triggering the parallel query processing. The whole collection of documents is partitioned among the

index servers, such that each server stores its own local subcollection, i.e., p subcollections compose the document collection C. Figure 1 illustrates this architecture for a typical search engine.

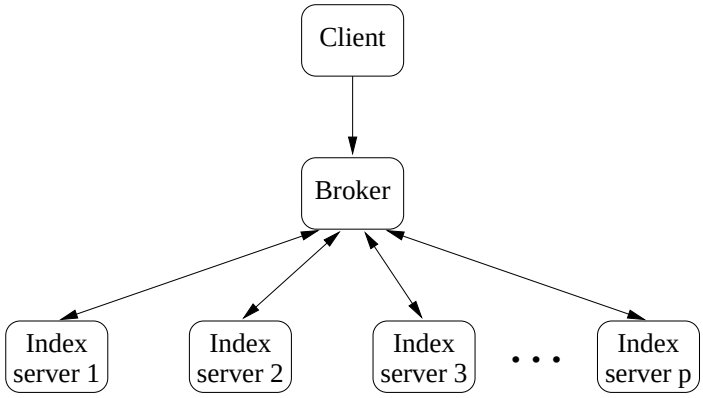


Fig. 1. Architecture of a typical search engine.

Let n be the size of the whole collection C. Assuming that the documents are evenly distributed among the p servers, the size b of any local subcollection is given by $b = n/p$.

3.2 Index Organization

An inverted index is adopted as the indexing structure for each subcollection. Inverted files are useful because they can be searched based mostly on the set of distinct words in all documents of the collection. They are simple data structures that perform well when the pattern to be searched for is formed by conjunctions and disjunctions of words (Baeza-Yates and Ribeiro-Neto, 1999; Witten et al., 1999).

The structure of our inverted indexes is as follows. It is composed of a *vocabulary* and a *set of inverted lists*. The vocabulary is the set of all unique terms (words) in the document collection. Each term in the vocabulary is associated with an inverted list that contains an entry for each document in which the term occurs. Each entry is composed of a document identifier and the within-document frequency $f_{t,d}$ representing the number of occurrences of term t within the document d. Furthermore, the inverted lists are sorted by decreasing within-document frequencies.

The size of each local inverted index is $O(b)$. This type of index organization, hereafter referred to as a *local index organization* (Ribeiro-Neto and Barbosa, 1998; Badue et al., 2001), is currently the de facto standard in all major search engines.

To avoid imbalance among index servers, we opt for balancing the distributions of the sizes of the inverted lists that compose the local inverted indexes. To achieve this we simply assign each document to an index server randomly. A random distribution of documents among index servers works well because it naturally spreads documents of various sizes across the cluster. As a result, the distributions of document sizes in the index servers become similar in shape, thus leading to inverted lists whose size distributions are also similar. Our motivation is to balance the storage space utilization at the different index servers and, as a consequence, reduce imbalance in execution time at the index servers (Badue et al., 2005), thus minimizing the effects of this possible source of imbalance.

Figure 2 illustrates the probability mass function (PMF) [1] of the size of the inverted lists that compose the 7 local inverted indexes in our cluster with 7 index servers. We observe that the distribution of storage use is very similar in shape throughout the different servers—actually, they overlap each other in Figure 2—, indicating that the random assignment of documents to servers works fine to balance storage use among servers.

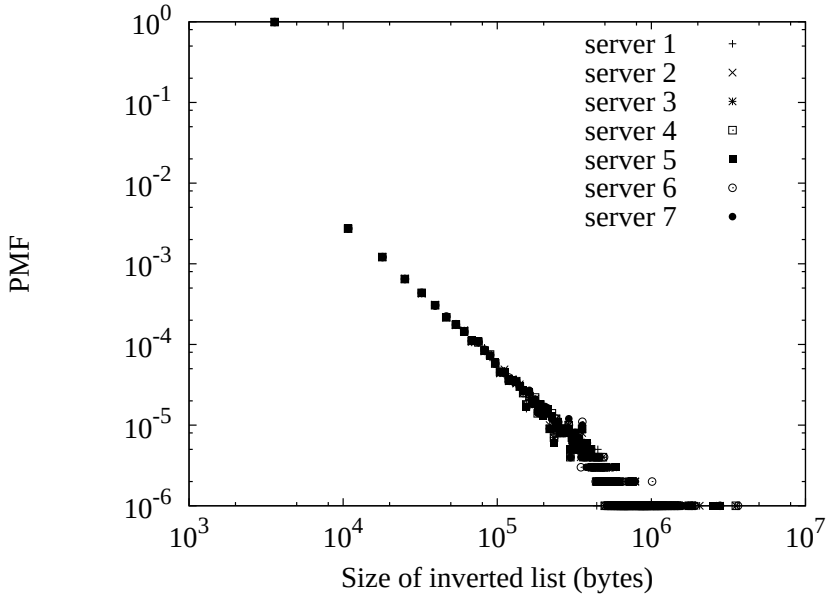


Fig. 2. PMF of the sizes of inverted lists.

[1] For discrete random variables, such as the size of inverted lists, we use a probability mass function (PMF). For continuous random variables described later, such as the execution time of queries, we use a probability density function (PDF).

In this paper, we use the standard vector space model (Salton and McGill, 1983) to rank the selected documents. In this model, queries and documents are represented as weighted vectors in a t-dimensional space, where t is the number of terms in the vocabulary of the collection. Each pair term-document is weighted by the frequency $f_{t,d}$ of term t in document d (the term frequency *tf*) and the inverse document frequency (*idf*) of the term t among the documents in the whole collection. The rank of a document with regard to a user query is computed as the cosine of the angle between the query and document vectors. Using the *idf* weight implies global knowledge about the whole collection to be available at the index servers. This could be accomplished if servers exchange their local *idf* factors after the local index generation phase. Each index server may then derive the global *idf* factor from the set of local *idf* factors (Ribeiro-Neto and Barbosa, 1998).

In our experiments, a client machine submits queries to the broker. This broker then broadcasts each query to all index servers. Once each index server receives a query, it retrieves the full inverted lists relative to the query terms, intersects these lists to produce the set of documents that contains all query terms (i.e., the conjunction of the query terms), computes a relevance score for each document, and sorts them by decreasing score—this results in a partial ranked answer to be sent by each index server to the broker. Query terms are processed by decreasing *idf*, i.e., by increasing order of the number n_t of documents in the whole collection containing the term t, thus leading to a significantly more efficient conjunction of their inverted lists. As soon as the ranking is computed, the top ranked documents selected at each index server are transferred to the broker machine. The broker is then responsible for combining the partial ranked answers received from the index servers through an in-memory merging operation. The final list of top ranked documents is then sent back to the client machine.

In our tests, we evaluate the full inverted lists. If partial evaluation of inverted lists is adopted—meaning shorter inverted lists—imbalance would be expected to be smaller. Nevertheless, partial evaluation of a huge document collection may cause a similar load as the one in our full evaluation case. Alternatively to the adopted standard ranking method, link information might be used to improve the relevance evaluation of retrieved documents, i.e., the documents resulting from the intersection of inverted lists related to the query terms. Traditional algorithms to compute link information are PageRank (Brin and Page, 1998) and HITS (Kleinberg, 1998). The computation of link information for each document in the collection is typically performed offline, during the index generation phase. Afterwards, the computed link information for each document remains available in main memory for the query processing phase.

Note that document ranking occurs after the intersection of inverted lists related to the query terms, the main focus of this paper, and deals only with straightforward computation of document scores based on information already available in main memory. Therefore, this document ranking—be it content-based, link-based, or a combination of both—can be fully carried out using main memory, thus not generating imbalance because the main sources of imbalance are related to disk operations, as further detailed in Section 4.

The broker is not a bottleneck in this architecture of local document partitioning. Our experimental results show that the average execution time per query at the broker is quite small (0.03 milliseconds in our cluster with 7 index servers). There are two fundamental reasons for this. First, broker's operation is fully carried out using main memory. Second, all the tasks the broker executes are simple tasks that do not take much CPU time. It should be noted that the broker does not have to make ranking computations and does not have to execute algebraic operations, other than comparing document identifiers. Preliminary results on this issue are presented in (Badue et al., 2005).

Notice that in this architecture, characterized by a strategy of local document partitioning, the response time of a particular query basically depends on the execution time of the slowest index server to produce the corresponding partial answer set. Therefore, the higher the imbalance in execution times of index servers, the larger tends to be the response time of a query processed by the cluster of servers. Thus, it is critically important to avoid imbalance. If the document collection is partitioned among a certain number of homogeneous index servers in a balanced way, such that all of them manage a similar amount of data when processing a query, it would be expected that execution times were also balanced. This idealized scenario of supposing balanced execution times as a direct consequence of a uniform collection distribution among servers is indeed a usual assumption taken by theoretical models in the literature to simplify the modeling task. Nevertheless, such an idealized balance is unlikely to be found in practice as we point out in this paper. Such an observation is based on the experimental analysis described in the following.

4 Characterizing Imbalance

We define the *imbalance of a given query* as the ratio between the maximum execution time and average execution time of index servers participating in the parallel processing for this particular query. This imbalance metric equals 1 in a perfectly balanced scenario that yields the maximum execution time exactly matching the average execution time. As the imbalance metric progressively gets higher than 1, there is a stronger indication that the query response time is dominated by a much larger execution time of a single server.

In Section 4.1, we characterize the workload used in our experiments. In Section 4.2, we describe our experimental setup, including the homogeneous cluster of index servers and the uniform distributions of sizes of inverted lists across index servers. Based on this experimental study, we verify the presence of a significant level of imbalance in execution time among the servers in despite of the collection being uniformly distributed among these same servers. Moreover, we identify and analyze the main sources for this imbalance: the use of disk caching in Section 4.3, and the size of main memory and the number of index servers in the cluster in Section 4.4. Finally, in Section 4.5 we characterize the execution times of queries in our query collection for our cluster of homogeneous index servers.

4.1 Workload Characterization

The test collection, referred to as WBR03, is composed of 10 million Web pages collected by the TodoBR [2] (TodoBR, 2003) search engine from the Brazilian Web in 2003. The inverted index for the whole collection occupies roughly 12 GB. The query set used in our tests is composed of 100 thousand queries, extracted from a partial log of queries submitted to the TodoBR search engine in September 2003.

The distribution of text terms in both queries and documents follows a Zipf distribution, as shown in Figure 3. The plots show the normalized frequency of text terms in documents and the normalized frequency of text terms in queries. The x-axis shows the resulting rank of each text term when these are sorted in decreasing order of occurrence in documents or in queries. Therefore, the frequency in documents (or in queries) that are expected for the x most frequent term is given by

$$f(x) = O(x^{-b}), \qquad b > 0. \tag{1}$$

Fitting a straight line to the log-plot of the data presented in Figure 3, we can estimate the value of the parameter b. We identify two different regions in the distribution of text terms in queries and three different regions in the distribution of text terms in documents. For the distribution of terms in queries, the values for parameter b are 0.617531 and 1.9918 for the first and second regions, respectively. For the distribution of terms in documents, the values for parameter b are 0.301089, 0.773434 and 1.58112 for the first, second, and third regions, respectively.

[2] TodoBR is a trademark of Akwan Information Technologies, which was acquired by Google in July 2005.

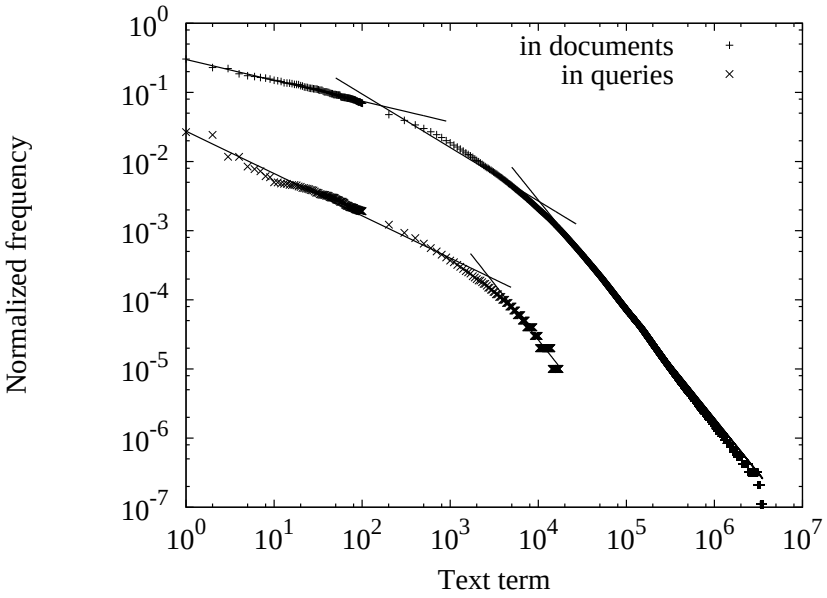


Fig. 3. Frequency of text terms in documents and in queries.

The query vocabulary has 21 552 terms and the text vocabulary has 3 541 678 terms. Common terms between both collections are 17 468. Figure 4 shows the normalized frequency of text terms in the document collection as a function of the normalized frequency of text terms in the query collection, thus considering only the 17 468 common terms between both distributions. Comparing the normalized frequency of text terms in documents and in queries, we observe that—even if dealing with rare query terms—it is likely that query terms are mentioned in a large number of documents. This is important because this indicates that such a query set consistently generates a significant query processing load in our system.

In fact, there are some very rare terms in our collection, thus leading to small inverted lists. As a consequence, when we partition the collection among the index servers, some of them may not store any portion of the inverted lists related to rare terms. In the case of queries concerning such rare terms, the imbalance is calculated as the ratio between the maximum execution time and average execution time of index servers that have inverted lists for the query terms and effectively participate in the parallel query processing. The number of unparticipating servers tends to increase with the total number of servers. In our test collection, this case occurs in only 2% of our queries and does not significantly impact the overall performance.

It is important to investigate if there is a uniform distribution of document partitions among index servers because otherwise this would be an expected source of imbalance. Consider our cluster with 7 index servers (detailed in Section 4.2), such that documents are randomly distributed in 7 subcollections.

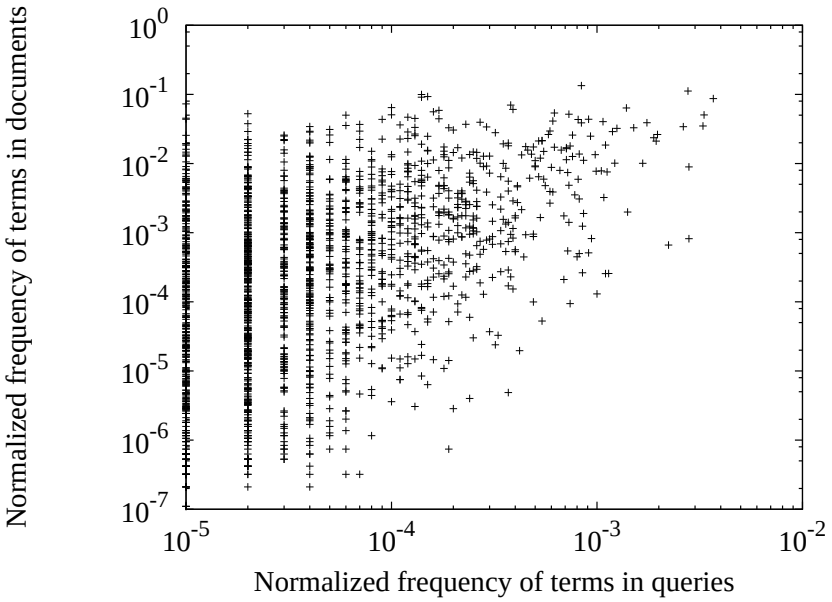


Fig. 4. Relationship between the frequency of terms in queries and in documents.

Table 1 shows the coefficient of correlation between the normalized frequency of text terms in the query collection and the normalized frequency of text terms in the subcollections of documents. We observe that the correlation pattern between the query and document collection remains virtually unchanged after the partition of the whole collection among the servers. This indicates that data distribution in our experiments seems unlikely to be a significant source of imbalance in the execution time of parallel query processing.

Table 1
Correlation between the frequency of text terms in queries and subcollections.

Subcollection	Coefficient of correlation
1	0.309722
2	0.309536
3	0.309643
4	0.309901
5	0.309465
6	0.309528
7	0.309692
Whole collection	0.309645

Figure 5 shows the PMF of the sizes of queries in our query log. The size of a

query is given by the sum of the sizes of the inverted lists related to its terms. The PMF of the sizes of queries follows a Zipf distribution, where we identify three distinct regions. The values for parameter b are equal to 0.981686, 0.816511, and 1.89127 for the first, second, and third regions, respectively. It is interesting to point out that the distribution of execution times of our query log follows the same kind of distribution of sizes of queries (shown in Section 4.5).

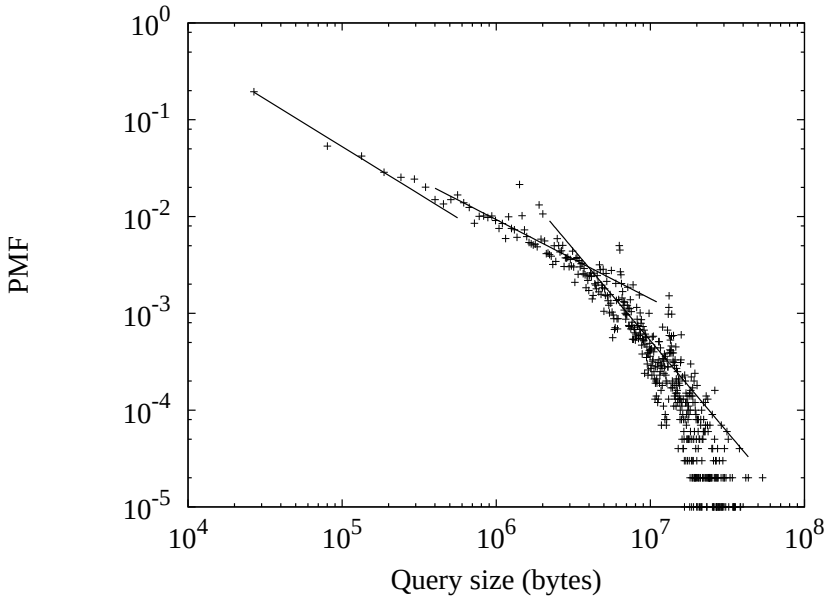


Fig. 5. PMF of the sizes of queries.

4.2 Experimental Setup

For the experiments reported in this paper, we use a cluster of 7 identical index servers. In our setup, each index server is a Pentium IV with a 2.4 gigahertz processor, 1 gigabytes of main memory and a ATA IDE disk of 120 gigabytes. The broker is an ATHLON XP with a 2.2 gigahertz processor and 1 gigabytes of main memory. The client machine, responsible for managing the stream of user queries, is an AMD-K6-2 with a 500 megahertz processor and 256 megabytes of main memory. All of them run the Debian Linux operating system version 2.6.

Our document collection is relatively small compared to the enormous collections handled by modern search engines. In order to overcome this limitation and establish a scenario to conduct our experiments where the absence of enough capacity for disk caching may happen, we maintain a bounded ratio between the size of the subcollections and the size of the main memory at each

server by limiting the latter to 200 megabytes, unless otherwise stated.

Although the utilization of disk space at index servers is balanced, as shown in Figure 2, we investigate if this balanced storage use among the subcollections reflects on balanced local execution times among index servers, or not. Figure 6 illustrates the distributions of average, maximum, and minimum local execution times per query. These statistics on execution time for a query are computed from local execution times of index servers that effectively participate in the parallel query processing in our cluster. Interval bars represent the minimum and maximum execution times for each query. To allow visual inspection, we display results for selected queries at intervals of 2 000 queries.

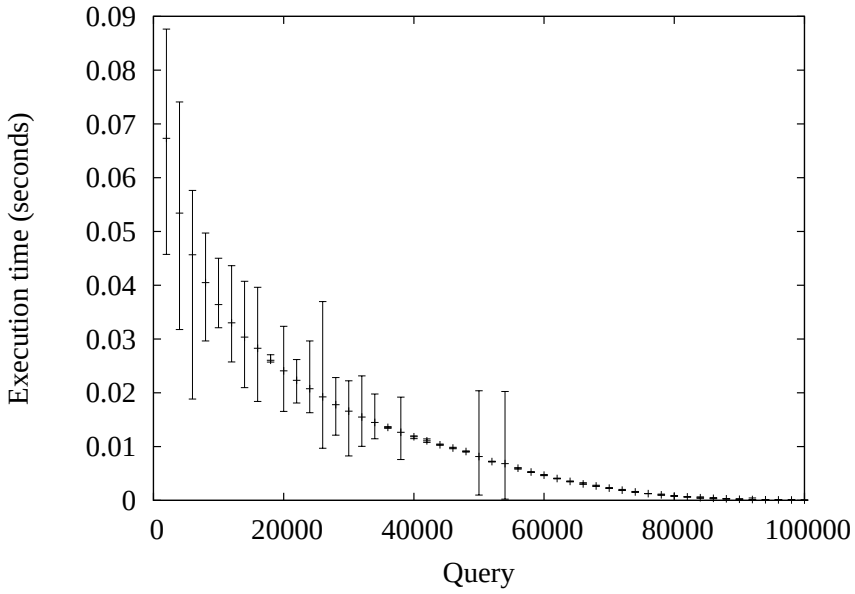


Fig. 6. Distribution of local processing times per query.

As an outcome of these experimental results, we verify in practice a consistent imbalance per query in the execution time at servers, even though the distribution of sizes of inverted lists at the various index servers are quite balanced. Motivated by this unexpected result, which contradicts the usual assumption of balanced execution times adopted by theoretical models found in the literature, we conduct a comprehensive experimental analysis to investigate the sources for the observed imbalance. As a consequence, we identify the main sources for imbalance: the use of disk caching, the size of main memory in the homogeneous index servers, and the number of servers in the cluster. We analyze the first source of imbalance in Section 4.3 and the other ones in Section 4.4.

We identify disk caching at the different servers as the major source of imbalance. To illustrate the consequences of this effect on query performance, we refer to a sample query processing observed in our real experiments described in Section 3.3, where we consider our cluster with 7 index servers and a user query q with the following execution times (in milliseconds) at servers: 31.83, 26.41, 30.12, 24.43, 5.27, 35.09, 28.18. For the same sample, the disk access times (in milliseconds) at servers is: 27.62, 22.18, 25.67, 20.25, 1.01, 30.87, 23.94, and the number of bytes retrieved from disk by the index servers is: 374 128, 375 920, 378 328, 375 712, 374 376, 373 864, 373 352. Even though index servers read from the disk a similar amount of data, the execution time of index server 5 is much smaller than the others (1.01 milliseconds). A possible explanation for this relatively small disk access time is that inverted lists were found in the disk cache of the operating system, thus accelerating disk I/O at this particular server in comparison with the disk access time observed at the other servers.

Figure 7 shows the PDF of local disk access times in our cluster. Note that the distributions of disk access time in the distinct index servers overlap each other, indicating that the behavior of disk access throughout the servers is very similar. We observe that the disk access times at all servers are basically grouped in two main regions: the first region is related to disk access times less than 4.5 milliseconds and the second region to disk access times greater than 4.5 milliseconds. We attribute the first region of smaller local disk access times to queries whose inverted lists are found in disk cache (referred to as *cache region*), and the second region of larger disk access times to queries whose lists had to be actually retrieved from disk (referred to as *disk region*). It is interesting to observe that the two vertical lines depicted in Figure 7 actually correspond to two technical characteristics of the adopted storage devices: the average rotational latency (4.5 milliseconds) and the later plus the average seek latency (13.5 milliseconds). Seek latency is the time taken to move disk heads to the right track and rotational latency refers to the waiting time until the right sector is under the read/write head.

We further analyze the relationship between the size of queries and the frequency of queries in the collection, investigating if there are any links from them to the use of disk caching. As previously explained, the size of a query is given by the sum of the sizes of the inverted lists related to its terms. Therefore, we consider separately the queries that find a certain level of correlation between the size of the inverted lists they demand and their frequency in the collection, and those that do not. To achieve this, we calculate the correlation as the ratio between the query size and the query frequency. If this ratio is greater than or equal to 0.25 and less than or equal to 4, then the size and the

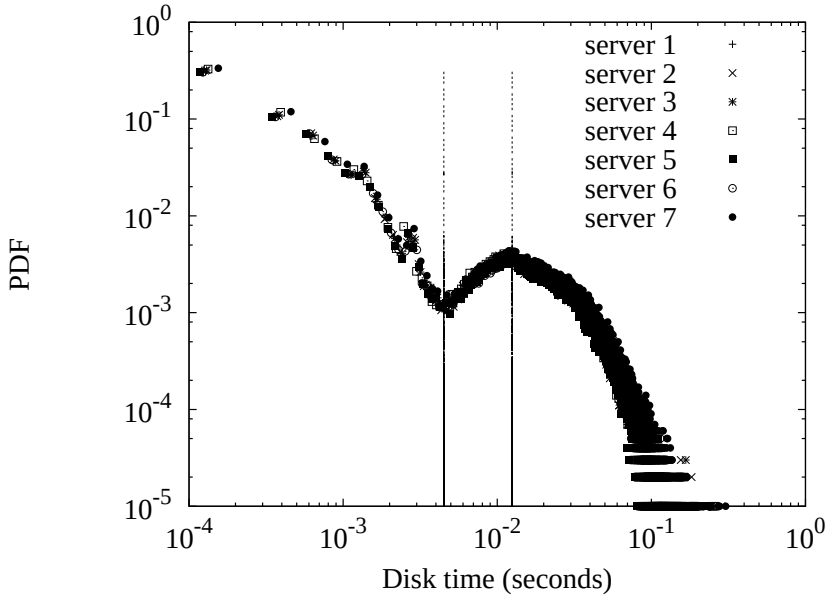


Fig. 7. PDF of local disk access times.

frequency of the query are related by a factor of 4, which we consider as representing a reasonable level of correlation between them. Therefore, queries that fall into this criterion are considered correlated, otherwise they are considered uncorrelated.

In Figure 8, we plot the normalized size of queries as a function of the normalized frequency of queries in the collection, but we make a distinction between the correlated and the uncorrelated queries. When we make this distinction, it is interesting to analyze separately three different representative regions that show up in Figure 8: (i) Region 1 is characterized by uncorrelated data where the size of queries is prevailing over the frequency of queries; (ii) Region 2 contains the correlated queries; and (iii) Region 3 is characterized by an uncorrelated region where the frequency of queries is prevailing over the size of queries.

Figure 9 compares the execution time for correlated and uncorrelated queries with respect to their size and their frequency in the collection. This comparison clearly shows that the well correlated queries (Region 2) have taken a better benefit of disk caching. This happens because they have the best tradeoff between the size of their inverted lists and their frequency in the collection. On the one hand, the largest inverted lists are demanded by the most frequent queries, favoring disk caching of these large inverted lists. On the other hand, rare queries, unlikely to find the inverted lists they need in cache, require the smallest inverted lists that do not demand large activate caching transfer times from the disk. For the uncorrelated data from Region 1, the frequency of queries is proportionally smaller than the size of queries. This implies that

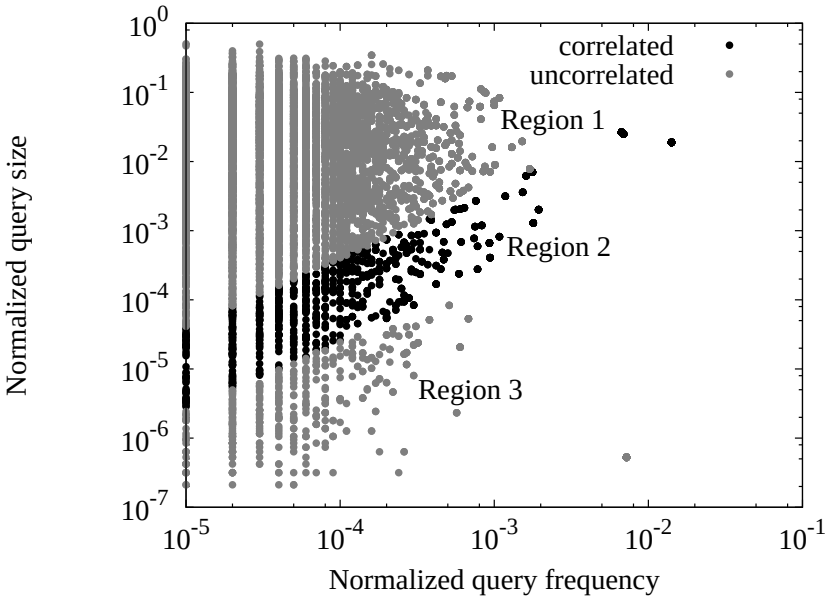


Fig. 8. Comparing query frequency and query size.

rare queries demand for large inverted lists, thus resulting in no cache use and large transfer delays. The uncorrelated data from Region 3 face the opposite: query terms impose relatively small data volumes to be retrieved in the system, thus getting small execution times either through small transfer delay or through the use of disk caching. Although these queries have small execution times they are not as numerous as the correlated ones or the uncorrelated ones in Region 1. Therefore, correlated queries prevail as a group in getting the smallest execution time. Furthermore, to corroborate this analysis it is important to notice that in Figure 9 the uncorrelated distribution mimics the size distribution (Figure 5) while the correlated one mimics the effect of disk caching (Figure 7).

Note that the disk access can be some orders of magnitude faster if the server finds the needed data in the cache region, thus avoiding the much slower actual access to the disk. For a given query q, if the local disk access time at a single index server is in the disk region and the local disk access times at the other servers are in the cache region, then the imbalance of query q might be severe.

Indeed, we verify that imbalance in execution times among index servers increases with the number of servers operating in the cache region, as shown in Figure 10. The points in Figure 10 show the imbalance for each query and the line shows the average imbalance over queries as a function of the number of servers operating in the cache region. This value is of course complementary to the number of servers operating in the disk region. For example, for a particular query being processed in our cluster of 7 servers, if Figure 10 shows that 2 of them operate in the cache region, then necessarily the other 5 are in

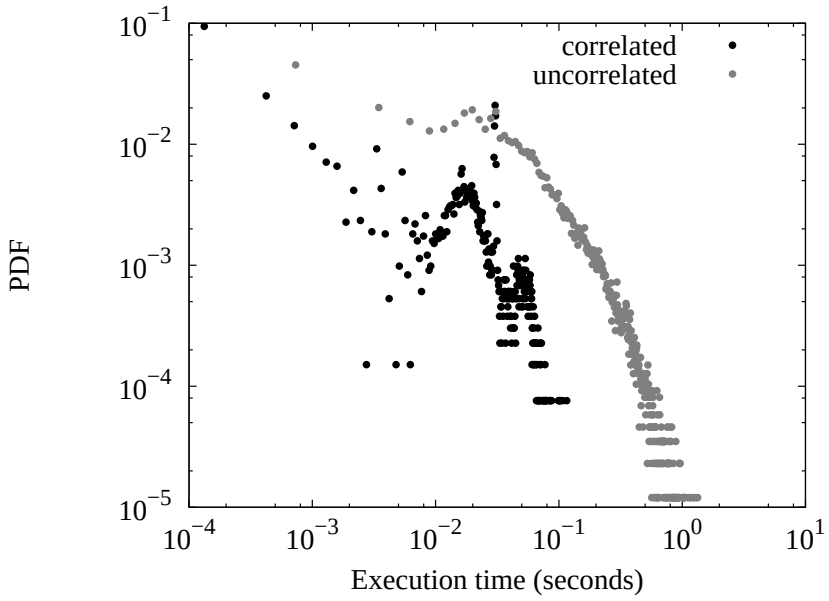


Fig. 9. PDF of execution times for correlated and uncorrelated queries.

disk region, directly influencing the imbalance magnitude.

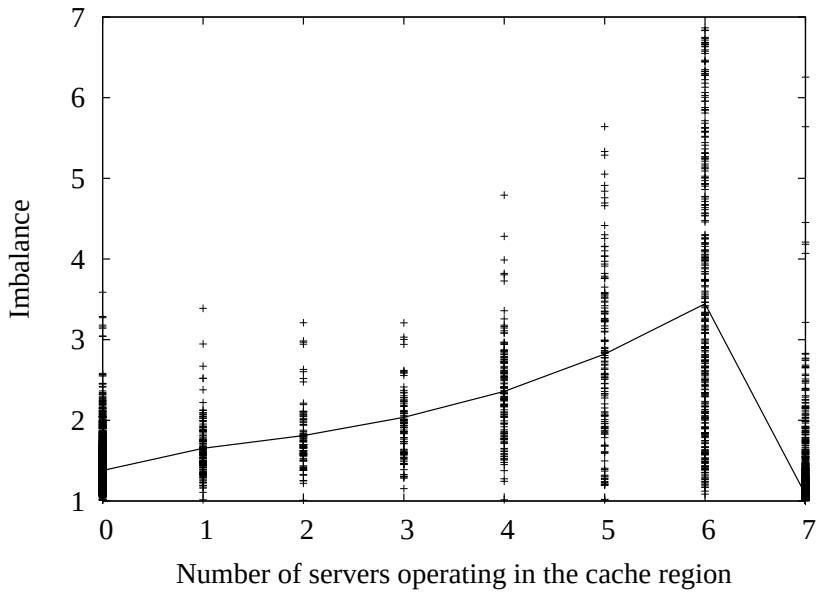


Fig. 10. Imbalance caused by the number of servers operating in the cache region.

To better understand how disk caching directly impacts imbalance, it is important to take a careful look at the average imbalance in Figure 10 for some representative scenarios: no caching, the worst case, and the best case. In the no caching scenario (i.e., 0 in the x-axis of Figure 10), all servers actually access the disk to retrieve the needed data, obtaining the lowest imbalance

(1.38) among the cases where there is at least one server operating in the disk region. The worst case for imbalance (i.e., 6 in the x-axis of Figure 10) presents a much higher imbalance (3.45) because a single server has a much larger execution time than the corresponding execution times at all remaining servers, thus leading to a high imbalance value. This happens because there is one single server that has a large execution time and a set of other index servers that have much smaller execution times because they retrieve the needed information from the disk cache. In contrast, the best case to avoid imbalance (i.e., 7 in the x-axis of Figure 10) results in an average imbalance of 1.08 and is achieved when all servers operate in the cache region, thus resulting in a small imbalance value due to the relatively small and similar disk access times throughout the cluster of servers.

The best case and the no caching scenarios present the two smallest imbalanced results, a consequence of having all servers operating in the same (cache or disk) region, thus providing no abrupt difference among the execution time of the participating servers. Nevertheless, the no caching scenario still yields a significantly higher imbalance with respect to the best case, which can be explained by the higher variance found in direct disk access when compared with memory access. Besides having the lowest imbalance, the best case also provides the fastest response time since all needed data to process a query is found in the disk caches at the servers. These results on the influence of disk caching on imbalance also suggest that the more memory available for disk caching at the servers, the lower the imbalance, and the larger the number of servers in the cluster the higher the imbalance, as we will discuss in the following.

4.4 Influence of Main Memory Size and Number of Servers

The results from Section 4.3 indicate that other source of imbalance is the size of the main memory of index servers because this affects the availability of data in the cache region at servers and, as a consequence, the imbalance. Therefore, we investigate in this section how the main memory size at the servers actually influences on the imbalance in parallel query processing.

Figure 11 shows the average imbalance as a function of the number of index servers in our cluster, while varying the size of the main memory at each server. We observe that the average imbalance in execution time of index servers increases as the size of main memory decreases, as would be expected. For the average imbalance shown in Figure 11, the best fitting we found was a logarithm growth of the number of servers given by $O(\log^{1.23783}(x))$, $O(\log^{1.03861}(x))$, $O(\log^{1.07531}(x))$, $O(\log^{1.20753}(x))$, $O(\log^{1.26731}(x))$, for 200, 300, 400, 500, and 600 megabytes of main memory, respectively.

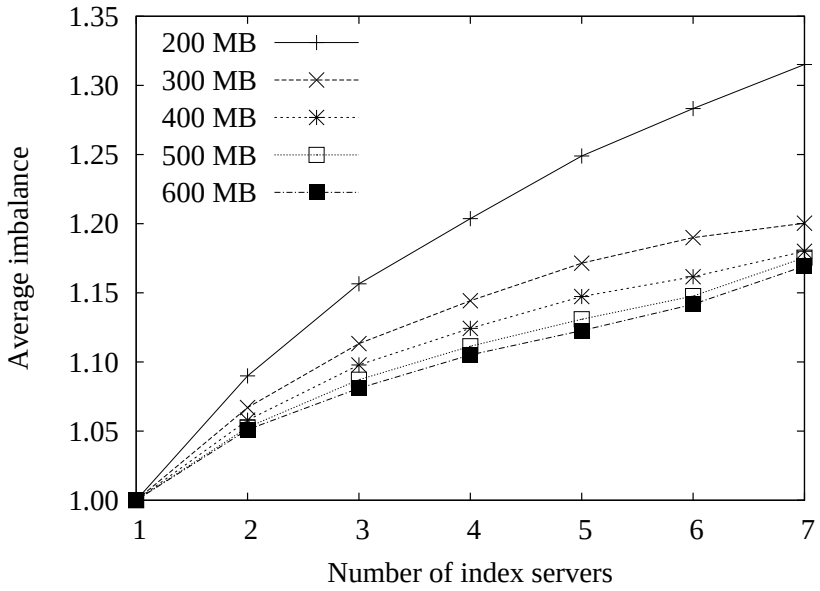


Fig. 11. Average imbalance as a function of the main memory size at servers.

On the one hand, when the main memory size is relatively large as compared to the size of the local index stored at the servers, there is more memory capacity available for the operating system to perform disk caching operations. This implies that local disk access times at all servers fall into the cache region for a high percentage of queries in our collection and this is exactly the best case scenario that produces the smallest imbalance (see Section 4.3). On the other hand, considering a relatively small main memory available for disk caching, index servers need to actually retrieve the inverted lists from the disk. In this scenario, the queries are more susceptible to imbalance as some disk blocks might be found in the cache of a few servers and not be found in the cache of the remaining servers. We also point out that there is a diminishing return in terms of imbalance while the RAM memory capacity increases.

The results presented in Section 4.3 also indicates that another source of imbalance is the number of index servers in the cluster, because the probability to occur variance among local execution times increases with the number of servers participating in the parallel query processing. We observe in Figure 11 that, for a fixed size of main memory, the average imbalance in execution times of index servers increases with the number of servers participating in the parallel query processing. We have already discussed in Section 4.3 that the average imbalance increases with the number of servers operating in the cache region, as shown in Figure 10. Therefore, this indicates that the larger the number of index servers participating in the parallel query processing, the higher the probability of increasing the ratio between the number of servers operating in the cache region and those in the disk region. As a consequence, this leads to larger imbalance in execution times per query in the cluster.

We now characterize the execution times of queries in our query log for our cluster of index servers. Figure 12 shows the PDF of execution times of queries in our cluster with a single index server. We observe that this PDF follows a Zipf distribution, where we identify three regions. The values for the parameter b are 0.754743, 0.369147 and 1.75183 for the first, second and third regions, respectively. It is interesting to note that the PDF of execution times of queries in Figure 12 resembles the PDF of sizes of queries shown in Figure 5. As expected, the larger the inverted lists related to the terms of a query, the larger the execution time to answer that query.

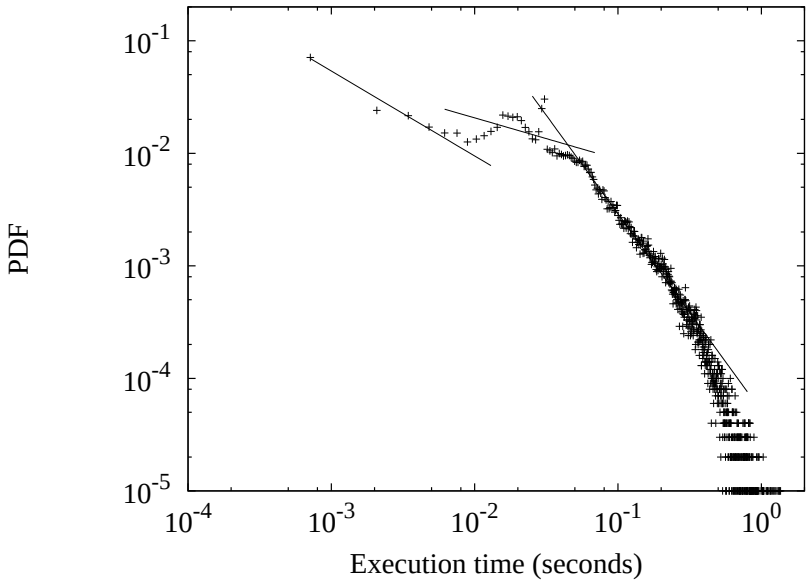


Fig. 12. PDF of the execution times of queries using a single index server.

Figure 13 shows the PDF of the maximum execution times to process queries in our query log in our cluster with 2 and 7 index servers. Similar results have been observed for scenarios with 3, 4, 5, and 6 index servers. We recall that the response time of a particular query depends on the maximum execution time related to the slowest index server to produce the corresponding partial answer set. We observe that each PDF of the maximum execution times of queries follows a Zipf distribution in all considered cases. Moreover, the shape of the distributions of the maximum execution times across the sets with a different number of participating servers in our cluster are very similar.

In the distributions shown in Figure 13, we identify three distinct regions. Figure 14 (a), Figure 14 (b), and Figure 14 (c) show the values for parameter b as a function of the number of index servers in our cluster, for the first, second,

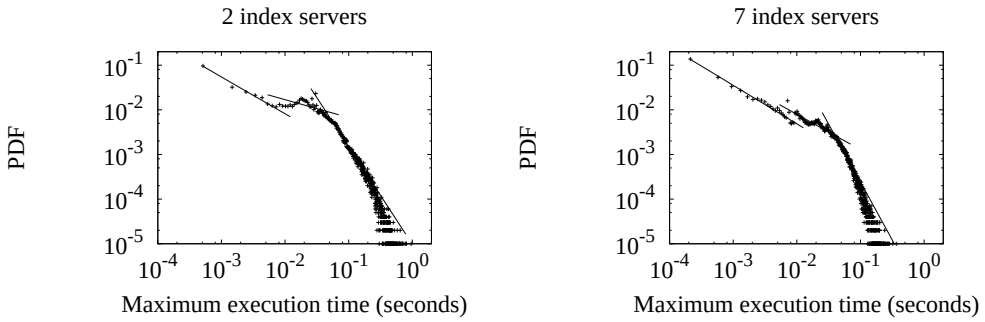
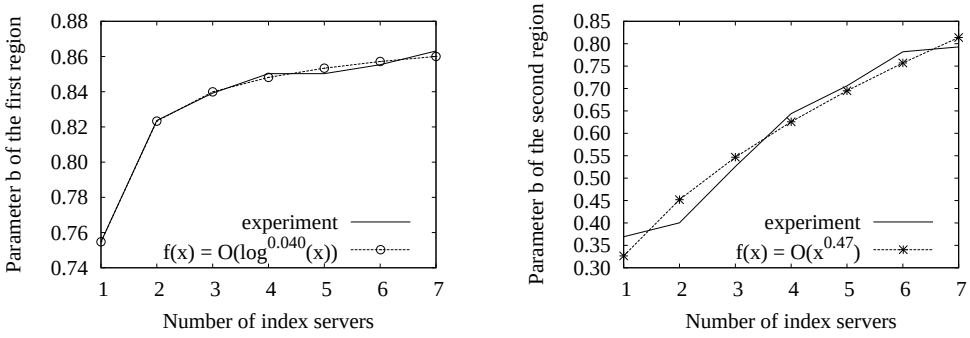


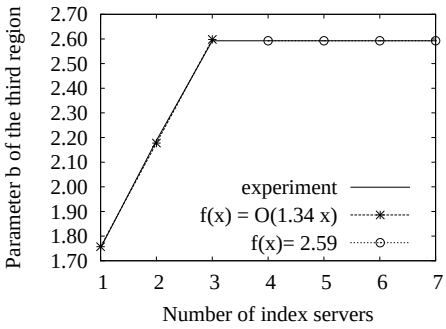
Fig. 13. PDF of maximum execution times for a varied number of servers.

and third distinct regions of the distributions of maximum execution times, respectively. For the parameter of the first region shown in Figure 14 (a), the best fitting we found was a logarithmic growth of the number of servers given by $O(\log^{0.040011}(x))$. For the parameter of the second region shown in Figure 14 (b), we fit a power law given by $O(x^{0.47})$. Finally, for the parameter of the third region shown in Figure 14 (c), for $1 \leq x \leq 3$, we fit a straight line given by $O(1.34x)$ and, for $x \geq 4$, the parameter b becomes a constant equal to 2.59. A possible reason for this is that the maximum execution times decrease with the number of index servers until a lower limit. In our cluster with 3 or more index servers, execution times located in third region approximates the lower limit, which implies in time distributions very similar in shape.

Finally, we evaluate if the per-query load imbalance observed in the previous experiments actually produces a reduction in the average throughput when multiple queries are concurrently served by the cluster of index servers. Figure 15(a) shows the average throughput as a function of the number of index servers in the cluster. The ideal scenario represents the average throughput of the cluster if the execution time in all index servers is perfectly balanced. In contrast, the real scenario expresses the average throughput of the cluster subject to the per-query imbalance in execution times of index servers participating in the parallel query processing, as verified in practice in our previous experiments. Comparing both scenarios, we note that the per-query imbalance actually reduces the average throughput of the cluster of index servers. Moreover, we point out that there is a trend towards a stronger reduction in the average throughput in the real scenario as compared to the ideal scenario. This is because the per-query imbalance increases with the number of index servers, as already shown in Figure 11, thus directly influencing on the reduction of the average throughput. A similar reasoning about the influence of the per-query imbalance applies to the increase of average maximum execution times of queries in the real scenario as compared to the ideal scenario, as shown in Figure 15(b).

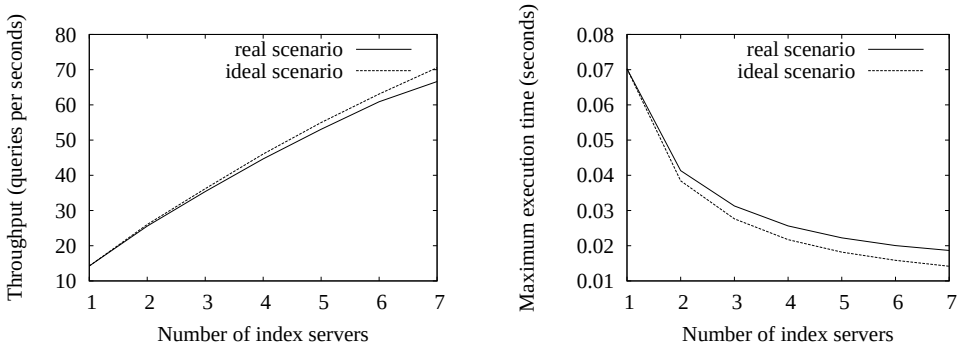


(a) Parameter b of the first region (b) Parameter b of the second region



(c) Parameter b of the third region

Fig. 14. Parameter b as a function of the number of servers.



(a) (b)

Fig. 15. Throughput and average maximum execution time.

5 Conclusions

In this paper, we investigated and analyzed the (im)balance among homogeneous index servers in a cluster for parallel query processing. As an outcome of our analysis, we verified that the idealized scenario, that supposes balanced execution times as a consequence of an uniform data distribution among homo-

geneous index servers, is unlikely to be found in practice. This is an important contribution because it sheds light on the usual assumption of balanced execution times taken by many theoretical models in the literature to simplify their modeling task (Tomasic and Garcia-Molina, 1993; Ribeiro-Neto and Barbosa, 1998; Cacheda et al., 2004; Chowdhury and Pass, 2003). Our findings derive from a comprehensive experimental analysis using an information retrieval testbed and real data obtained from a real-world search engine. Besides verifying the presence of a certain level of imbalance among homogeneous servers, we have also identified and characterized the main sources for this unexpected imbalance.

The key factor for imbalance is the use of disk caching at the different servers. We verified that imbalance for each query increases with the number of servers that retrieve the needed documents from the cache. On the one hand, the worst case for imbalance is achieved when a single index server must actually access the disk for documents while all remaining servers are using the cache for the same query. The worst case presents a much higher average imbalance of 3.45—computed as a ratio between the maximum and the average execution time among servers—because in this case a single server has an execution time much larger than the corresponding execution times at all the remaining servers, thus resulting in a high imbalance. On the other hand, the best case to avoid imbalance results in an average imbalance of 1.08. It is achieved when all servers operate in the cache region, thus leading to a relatively small and similar disk access times throughout the cluster of servers.

Other identified source of imbalance is the size of the main memory of the homogeneous index servers, which affects the availability of resources for disk caching at servers. We verified that the average imbalance decreases as the size of main memory increases, but this happens with a diminishing return with an increase in memory capacity. When the main memory size is relatively large as compared to the size of the local index stored at the index servers, all servers retrieve data from their caches for a high percentage of queries, which is the best case scenario that produces the lowest imbalance. In contrast, considering a relatively small main memory capacity available for disk caching, for a given query some servers might access their caches while the others might need to actually access the disk, thus leading to a higher imbalance. In our cluster with 7 index servers, the average imbalance is 1.32 and 1.18 for 200 and 400 megabytes of main memory at each server, respectively.

Another identified source of imbalance is the number of index servers in the cluster. We verified that, for a fixed size of main memory, the average imbalance in execution times of index servers increases with the number of servers in the cluster. The reason is that the larger the number of index servers participating in the parallel query processing, the higher the probability of some servers to operate in the cache region and, as a consequence, the higher the

imbalance in execution times of those servers. For 200 megabytes of main memory, the average imbalance is 1.08994 and 1.31512 in our cluster with 2 and 7 index servers, respectively.

Overall, the primary source of imbalance per query is the heterogeneous use of disk caching among the homogeneous index servers. Main memory size at servers and the number of index servers are actually indirect sources of imbalance because they cause heterogeneous use of disk caching. A direction for future work is to develop and validate an analytical model to evaluate the performance of parallel query processing in a cluster of homogeneous index servers, while taking into account the imbalance in execution times among the index servers. We intend to use such an analytical model to evaluate and compare search architectures under different operational constraints, such as a maximum response time and a minimum throughput to be achieved when processing user queries.

6 Acknowlegments

This work was supported by the GERINDO project–grant MCT/CNPq/CT-INFO 552.087/02-5, by CNPq scholarship 140262/2001-6 (Claudine Santos Badue), by Millenium Nucleus Grant P04-067-F of the Chilean Planning Ministry (Ricardo Baeza-Yates), by CNPq grant 30.0188/95-1 (Berthier Ribeiro-Neto), and by CNPq grant 520.916/94-8 (Nivio Ziviani). We thank the anonymous referees for the valuable comments and suggestions that helped us improve the paper.

References

Badue, C. S., Baeza-Yates, R., Ribeiro-Neto, B., Ziviani, N., 2001. Distributed query processing using partitioned inverted files. In: Proceedings of the Eighth String Processing and Information Retrieval Symposium (SPIRE'01). IEEE Computer Society, Laguna de San Rafael, Chile, pp. 10–20.

Badue, C. S., Barbosa, R., Golgher, P., Ribeiro-Neto, B., , Ziviani, N., 2005. Basic issues on the processing of web queries. In: Proceedings of the 28th Annual International ACM SIGIR Conference on Reseach and Development in Information Retrieval (SIGIR'05). Salvador, Bahia, Brazil, pp. 577–578.

Baeza-Yates, R., Ribeiro-Neto, B. (Eds.), 1999. Modern Information Retrieval. ACM Press New York, Addison Wesley.

Barroso, L. A., Dean, J., Holzle, U., 2003. Web search for a planet: The google cluster architecture. IEEE Micro 23 (2), 22–28.

Brin, S., Page, L., 1998. The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems 30 (1-7), 107–117.

Cacheda, F., Plachouras, V., Ounis, I., 2004. A case study of distributed information retrieval architectures to index one terabyte of text. Information Processing and Management 41 (5), 1141–1161.

Chowdhury, A., Pass, G., 2003. Operational requirements for scalable search systems. In: Proceedings of the Twelfth International Conference on Information and Knowledge Management (CIKM'03). New Orleans, LA, USA, pp. 435–442.

Jupiter Media, 2005. JupiterResearch forecasts online advertising market to reach 18.9 billion by 2010; search advertising revenue to surpass display. www.jupitermedia.com/corporate/releases/05.08.15-newjupresearch2.html, Press Release.

Kleinberg, J., 1998. Authoritative sources in a hyperlinked environment. In: In Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms. San Francisco, California, USA, pp. 668–677.

MacFarlane, A., McCann, J. A., Robertson, S. E., 2000. Parallel search using partitioned inverted files. In: Proceedings of the Seventh International Symposium on String Processing and Information Retrieval (SPIRE'00). IEEE Computer Society, La Coruña, Spain, pp. 209–220.

Orlando, S., Perego, R., Silvestri, F., 2001. Design of a parallel and distributed web search engine. In: Proceedings of the 2001 Parallel Computing Conference (ParCo'01). Imperial College Press, Naples, Italy, pp. 197–204.

Ribeiro-Neto, B. A., Barbosa, R. A., 1998. Query performance for tightly coupled distributed digital libraries. In: Proceedings of the Third ACM Conference on Digital Libraries (JCDL'98). pp. 182–190.

Risvik, K. M., Aasheim, Y., Lidal, M., 2003. Multi-tier architecture for web search engines. In: Proceedings of the First Latin American Web Congress (LAWEB'03). Santiago, Chile, pp. 132–143.

Salton, G., McGill, M. J., 1983. Introduction to Modern Information Retrieval. McGraw-Hill.

TodoBR, 2003. http://www.todobr.com.br.

Tomasic, A., Garcia-Molina, H., 1993. Performance of inverted indices in shared-nothing distributed text document information retrieval systems. In: Proceedings of the Second International Conference on Parallel and Distributed Information Systems (PDIS'93). San Diego, California, USA, pp. 8–17.

Witten, I. H., Moffat, A., Bell, T. C., 1999. Managing Gigabytes: Compressing and Indexing Documents and Images, 2nd Edition. Morgan Kaufmann Publishers, Inc.