

Basic Issues on the Processing of Web Queries

Claudine Badue¹
claudine@dcc.ufmg.br

Ramurti Barbosa²
ramurti@akwan.com.br

Paulo Golgher³
golgher@akwan.com.br

Berthier Ribeiro-Neto⁴
berthier@dcc.ufmg.br

Nivio Ziviani⁵
nivio@dcc.ufmg.br

^{1,4,5}Federal University of
Minas Gerais
Belo Horizonte, Brazil

^{2,3}Akwan Information
Technologies
Belo Horizonte, Brazil

ABSTRACT

In this paper we study three basic and key issues related to Web query processing: load balance, broker behavior, and performance by individual index servers. Our study, while preliminary, does reveal interesting tradeoffs: (1) load unbalance at low query arrival rates can be controlled with a simple measure of randomizing the distribution of documents among the index servers, (2) the broker is not a bottleneck, and (3) disk utilization is higher than CPU utilization.

Categories and Subject Descriptors: H.3.4 Information Storage and Retrieval: Systems and Software - *distributed systems, performance evaluation*; H.3.5 Online Information Services - *Web-based services*

General Terms: Performance, design.

Keywords: Distributed query processing, search engines, load balance.

1. THE PROBLEM

When a user query reaches a search engine, its processing is broken down into two phases. The first phase consists of taking the query terms, extracting from the disks information on documents that contain each query term, executing a conjunction of the sets of documents that contain each query term, and finally ranking the selected documents. The second phase consists typically of taking the top 10 ranked answers and generating snippets, title, and URL information for each of them.

In this work, we concentrate our attention on the first phase of the query processing task, studying three basic issues that affect the internal operation and fine tuning of the cluster of servers where the indexes are stored.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGIR '05, August 15–19, 2005, Salvador, Brazil.

Copyright 2005 ACM 1-59593-034-5/05/0006 ...\$5.00.

2. ARCHITECTURE

Search engines use a cluster of servers as platform for query processing [1, 4]. Typically, in this network there are two types of machines: a single broker and p index servers. The whole collection of documents is evenly partitioned among the index servers, such that each server stores its own local subcollection with approximated size (in bytes). This type of index organization, here referred to as a *local index organization* [3], is the standard “the facto” in all major search engines.

A user query reaches the broker through a client machine. The broker then broadcasts the query to all index servers. Each index server searches its own local index and produces a partial ranked answer. These partial ranked answers are collected by the broker and combined through an in-memory merging operation. The final list of ranked documents is then sent back to the client machine.

An inverted file is adopted as the indexing structure for each subcollection. To rank documents, the index server reads from disk the inverted lists relative to query terms, intersect the lists to produce the set of documents that contain all query terms, and compute a relevance score for each document. The relevance score is computed in function of the relative frequencies of occurrence of terms within documents and anchor texts, the authoritative degree of documents [2], URL and title information of documents, and the proximity of query terms within documents.

For the experiments reported in this work we use a cluster of 8 index servers. The test collection is composed of Web pages collected by the TodoBR [5] search engine from the Brazilian Web in 2003. Each index server holds a subcollection of 10 million pages. The index for each subcollection occupies roughly 16 gigabytes. The queries are a set of 10,000 unique queries submitted to the TodoBR search engine in January of 2003.

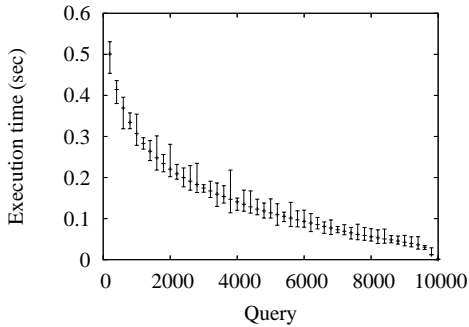
3. BASIC ISSUES

3.1 Load Balance

To reduce load unbalance among index servers we opt for balancing the distributions of the sizes of the inverted lists that compose the local inverted files. (stored locally at the index servers). This can be accomplished by a random dis-

tribution of documents among index servers, which 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, which reflects in inverted lists whose size distributions are also similar.

First, we programmed the client to submit queries one at a time, for evaluating local processing times independently of interference among the various processing threads at index servers. Figure 1 illustrates the distributions of average, maximum, and minimum local processing times per query. Average time for a query q is computed as the average time among all 8 local processing times. The time interval surrounding average times have limits given by the slowest and the fastest index server for that query.



1: Average, maximum, and minimum local processing times (seconds) per query. Queries are sorted by decreasing order of average time. Only selected queries are shown explicitly, at intervals of 200 queries.

We notice that load unbalance is controlled when queries are submitted sequentially. Second, we programmed the client to submit queries at an average rate of r queries per second. We vary r until the cluster saturates. We notice that at high arrival rates, variance of local processing times quickly increases limiting scalability of cluster throughput. That is, randomizing the distribution of documents among servers does not help at high query arrival rates. This clearly suggests that adding a module of dynamic load balance to a cluster of servers (of a search engine) might be an effective measure to improve throughput at high arrival rates.

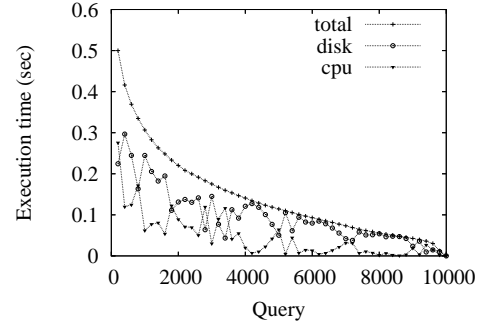
3.2 The Broker is not a Bottleneck

In this cluster architecture, the broker constitutes a potential bottleneck. To stress the broker and observe how it behaves, we implemented a concurrent process monitor that simulates a cluster with p index servers. The broker takes queries at an arrival rate r (which we varied) and passes them to a single machine that runs our concurrent monitor. For each query, the broker sends p requests to our monitor, one for each index server. For each request it receives, the monitor simulates a local query processing task.

We observe that even at high loads and with a large number of servers, time at the broker is not affected. In fact, with 256 servers and query arrival rates around 100 queries per second, average time at the broker per query is less than 10 milliseconds. That is, the broker is not a bottleneck in our architecture. This is because all the tasks the broker executes are simple tasks that do not take much CPU time and carried out fully in main memory.

3.3 Disk and CPU Times at Servers

For examining how local processing time is split among disk and CPU at index servers, we again submitted the queries to the servers one at a time. Figure 2 illustrates average local processing time, average disk access time, and average CPU execution time (in seconds). We observe that disk times are remarkably dominant.



2: Average local processing time (total), average disk access time, and average CPU execution time (seconds) per query. Queries are sorted by decreasing order of local processing times (total).

4. CONCLUSIONS

We have studied three basic and key issues related to the internal operation and fine tuning of a cluster of index servers for search engines: load balance, broker behavior, and CPU and disk times at individual index servers. Load unbalance among index servers can be controlled by randomizing the distribution of the documents of the collection among the servers. This works well at low arrival rates, but is not effective at high arrival rates. The broker is not a bottleneck in a shared-nothing architecture of independent index servers. Disk times dominate local processing times.

5. ACKNOWLEDGMENTS

This work was supported by the GERINDO project—grant MCT/CNPq/CT-INFO 552.087/02-5, by CNPq scholarship 140262/2001-6 (Claudine Badue), by CNPq grant 520.916/94-8 (Nivio Ziviani) and by CNPq grant 30.0188/95-1 (Berthier Ribeiro-Neto).

6. REFERENCES

- [1] L. A. Barroso, J. Dean, and U. Holzle. Web search for a planet: The google cluster architecture. In *IEEE Micro*, pages 22–28, 2003.
- [2] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. In *Journal of the ACM*, volume 46, pages 604–632, 1999.
- [3] B. A. Ribeiro-Neto and R. A. Barbosa. Query performance for tightly coupled distributed digital libraries. In *Proceedings of the third ACM Conference on Digital Libraries*, pages 182–190, 1998.
- [4] K. M. Risvik, Y. Aasheim, and M. Lidal. Multi-tier architecture for web search engines. In *Proceedings of the First Latin American Web Congress*, pages 132–143, Santiago, Chile, 2003.
- [5] TodoBR. Main page: <http://www.todobr.com.br>.