ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/221613462

# On the feasibility of multi-site web search engines

**Conference Paper** · November 2009

DOI: 10.1145/1645953.1646009 · Source: DBLP

CITATIONS
30

READS
101

**5 authors**, including:

Ricardo Baeza-Yates
University Pompeu Fabra
**603** PUBLICATIONS **28,438** CITATIONS
SEE PROFILE

Aristides Gionis
Aalto University
**184** PUBLICATIONS **8,965** CITATIONS
SEE PROFILE

Vassilis Plachouras
Thomson Reuters
**83** PUBLICATIONS **1,793** CITATIONS
SEE PROFILE

**Some of the authors of this publication are also working on these related projects:**

Keyword Advertising View project

Web Searching View project

All content following this page was uploaded by Ricardo Baeza-Yates on 10 September 2017.

The user has requested enhancement of the downloaded file.

# On the Feasibility of Multi-Site Web Search Engines

Ricardo Baeza-Yates Aristides Gionis Flavio Junqueira
Vassilis Plachouras Luca Telloli
Yahoo! Research, Barcelona, Spain
rbaeza@acm.org, gionis@yahoo-inc.com, fpj@yahoo-inc.com
vplachouras@acm.org, telloli@yahoo-inc.com

## ABSTRACT

Web search engines are often implemented as centralized systems. Designing and implementing a Web search engine in a distributed environment is a challenging engineering task that encompasses many interesting research questions. However, distributing a search engine across multiple sites has several advantages, such as utilizing less compute resources and exploiting data locality. In this paper we investigate the cost-effectiveness of building a distributed Web search engine. We propose a model for assessing the total cost of a distributed Web search engine that includes the computational costs and the communication cost among all distributed sites. We then present a query-processing algorithm that maximizes the amount of queries answered locally, without sacrificing the quality of the results compared to a centralized search engine. We simulate the algorithm on real document collections and query workloads to measure the actual parameters needed for our cost model, and we show that a distributed search engine can be competitive compared to a centralized architecture with respect to real cost.

## Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

## General Terms

Algorithms, Design, Performance

## Keywords

Web search, distributed systems, query processing, cost model

## 1. INTRODUCTION

Distributed architectures for search engines are a potential solution to the scalability problem of Web search. As data centers hosting search engine servers have limited capacity, it is critical to have a system design that can cope with the growth of the Web, and that is not constrained by the physical limitations of a data center.

On the other hand, single-site systems are attractive from an economic point of view. It is often the case that companies select the location for a new data center based on the cost of land, labor, and public utilities, in particular power. By considering only such factors, system architects might be overlooking the potential advantages for having many smaller data centers close to the end users [14].

Suppose that we are to design a new system for Web search for users across a number of countries. A typical solution is to use a single, centralized site, since it is a simple and competitive solution. However, the preference for a centralized solution often comes from a lack of understanding of the benefits and the drawbacks of a distributed solution, as it is not clear whether the benefits of a distributed architecture compensate its overheads.

An example of an important benefit of a distributed solution is the proximity of the search engine to Web data and its users. Being closer to data might impact positively the performance of crawlers, since the Web connections are shorter and fetching documents is faster (the time to download a document increases linearly with round-trip time [12]). Consequently, longer connections reduce the throughput of crawlers and front-end servers that host Web servers interacting with users. Additionally, distributing the overall workload mitigates the problem of network bandwidth saturation, and enables redundancy and fault tolerance employing less costly configurations [14, 37]. A distributed architecture also enables the service to exploit the potential local properties of the workload. Locality might improve the ranking by using local features.

We hence provision a site to process locally a fraction of the queries, ideally all local query traffic. In practice, achieving the goal of processing all queries locally is difficult. Some queries, called *non-local queries*, might require multiple sites to process them. The additional communication increases the total latency of query processing, and hence the latency for non-local queries is higher. On the other hand, *local queries* are processed faster. We use local queries to denote queries that can be processed by the site to which they are submitted to. We then call *locality* the fraction of the queries that are processed locally. Thus, if we can process many queries locally, we may be able to reduce the average latency of query processing. In addition to locality, we also look at the volume of queries for which the distributed system retrieves more or fewer clicked documents than a centralized system, assuming that a user click is an indication of relevance.

The sites that comprise the distributed Web search engine may be connected via a simple network topology, such as star, ring or clique. The algorithms we discuss in this paper assume that each site is able to communicate with each other site, possibly using a path of intermediate sites. For our experiments we use a simple star topology, which has a minimal number of connections and requires at most two hops between any pair of sites, but any other network topology is feasible, and in fact part of the design space to explore.

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.
*CIKM'09,* November 2–6, 2009, Hong Kong, China.
Copyright 2009 ACM 978-1-60558-512-3/09/11 ...$10.00.

**Our contributions.** In this paper we assess the feasibility of distributed Web search engines comprising geographically dispersed sites. We propose a detailed cost model that includes operational costs, and enables us to answer questions of the following flavor:

- Given an optimization that reduces the average latency of an operation (crawling, receiving request, query processing), how does it affect power consumption?
- Given different local rates for power, how much resource savings are necessary to compensate for such differences?

We use illustrative, yet realistic examples to demonstrate that a distributed search engine can be more cost effective than a centralized one, assuming that there is sufficient locality in the query workload. To our knowledge, we are the first to propose such a model to assess the operational costs of multi-site Web search systems.

We then design a query-processing algorithm that maximizes the amount of locally-processed queries, without sacrificing the quality of the results. We have implemented this query processing algorithm, and we use this implementation in the emulation of a distributed search system. Different from federated search [9],[1] most queries are processed based on geographical distance in a single site. Also, in our setting all the search sites are homogeneous and under the control of the search engine. We propose modifications to our algorithm to improve the amount of locality, and consequently performance, such as replicating documents retrieved frequently across sites. We use real document collections and real query workloads to measure the actual parameters needed by our cost model, and show that a distributed search engine can be competitive with respect to a centralized one and at the same time be less expensive.

Building a distributed search engine involves designing and optimizing a large number of different components. We do not claim to provide a complete solution to the problem of building a distributed search engine. Our contribution instead is an evaluation of a distributed retrieval solution with respect to the utilization of important system resources such as power and network bandwidth. To our knowledge, no previous work has performed such an evaluation. Our goal hence is not to be comprehensive, but instead to provide insight on the feasibility of distributed retrieval solutions when considering resource utilization.

**Roadmap.** The organization of the paper is as follows: We discuss related work in Section 2, and introduce our basic distributed setting in Section 3. In Section 4 we present our cost model, and in Section 5 we provide an analysis that bridges our cost model with the query-processing algorithm that we present in Section 6. In Sections 7 and 8 we present results from simulating our algorithm on a real data set. We finish with conclusions and future research directions in Section 9.

## 2. RELATED WORK

Designing an operational distributed Web search engine presents several challenges [2]. There has been work on several pieces of the problem that we study in the context of cluster-based centralized search engines. Brin and Page [8] describe an early version of the Google search engine. Barroso *et al.* [4] also discuss issues related to the architecture, cost and electrical power for the search clusters of the Google search engine. Orlando *et al.* [31] describe the design of a parallel and distributed search engine, and compare the performance of task parallelization, data parallelization, and a combination of the two. Risvik *et al.* [32] also propose an architecture for a multi-tier search engine, aiming to increase the capacity of a cluster by placing documents in different tiers.

[1]Also called distributed retrieval in some articles.

There has been a number of publications on large scale crawling. Cho and Garcia-Molina look at the issues of overlap, quality and network bandwidth for parallel crawlers [13]. The design of high performance crawlers, distributed in LANs, is discussed in [21, 34]. Recently, Lee *et al.* reported on the implementation of a crawler running on a single machine and collecting 6 billion pages [23].

Melnik *et al.* [25] study the construction of a large-scale distributed index, optimizing the storage and distribution of files, as well as the pipeline of reading data from disk, processing and then flushing data back to disk. Tomasic and Garcia-Molina [36] investigate the organization of the inverted index for query processing in distributed search engines. Moffat *et al.* [28] compare the throughput of query processing with document partitioning, term partitioning, and pipelined query processing with term partitioning. Load-imbalance issues in term and document partitioned search architectures are discussed in [27] and [1], respectively. Bender *et al.* [6] explore the design space of cluster architectures for large scale search engines, and compare the cost of hardware needed, considering index partitioning, distribution of the index partitions, and the technology used for storage.

We use caching as part of our solution. Caching for search engines has been studied extensively in the literature [18, 33, 3].

Exposto *et al.* try to find the optimal locations for several Web crawlers considering the data volume and the time spent crawling [17]. Li *et al.* study the feasibility of P2P Web search engines in terms of network bandwidth and storage space on the peers [24], and conclude that Web search using P2P technology still requires an order of magnitude more resources than available, despite a range of considered performance optimizations. Zhang and Suel combine top-$k$ query processing with bloom filters in a P2P setting with a term-partitioned index and use bandwidth to model the cost of query processing [38]. Cambazoglu *et al.* use cost models to investigate the benefits of distributed multi-site Web search engines in terms of relevance gains and crawling coverage [11].

None of the papers discussed above considers the real cost of crawling or searching in a multi-site configuration given by the power consumption in each site and the local network costs. Craswell *et al.* [15] compare the cost of different architectures ranging from a centralized search service to a metasearch service, which forwards queries to other independent search services. The developed cost model, however, refers to the search service running on a single site. For modeling the real cost of systems, we use the results on the cost of electrical power in data centers from [19].

We note that our distributed search setting is closer to tiering than to federated search [9] or metasearch [26]. As discussed in Section 1, a site processes any local query first, and only if necessary other sites process such a query. Federated search systems first rank sites [10, 35] and then send queries to top-ranked sites.

Our distributed query-processing algorithm uses score thresholds so that sites can decide when there is no need to request partial results from other sites. Ntoulas and Cho [29] have used a similar idea for pruning inverted indexes in a two-tiered architecture.

We also replicate documents frequently retrieved as top results in previous queries. The frequency with which documents have been retrieved in the past has been used for efficient query processing to sort posting lists [20].

## 3. SYSTEM MODEL

In this section we outline the basic assumptions for our distributed system and we introduce some of the notation that we will use throughout the paper. We consider the problem of designing a distributed Web search engine over $n$ remote sites $\mathcal{S} = \{S_1, \ldots, S_n\}$. We assume that site $S_i$ is associated with geograph-

ical regions (*e.g.*, states, countries, group of countries, continents), but they could also represent other types of entities (e.g., languages, domain names, network clusters, etc.). The organization of the sites does not need to be flat, and sites may have special roles. For instance, we can organize them hierarchically such that sites have distinct roles. Also we assume that the sites communicate with each other via some network topology, such as a star-shaped topology. Deciding the optimal network topology to use is also part of the design parameters of the distributed system architecture.

We assume a collection of documents $\mathcal{D}$ over a set of terms $\mathcal{T}$. We assume that the documents $\mathcal{D}$ are partitioned into two subsets: local ($L$) and global ($G$). Global documents are present in all sites, whereas local documents are further partitioned disjointly among the sites of $\mathcal{S}$. That is, site $S_i$ contains the documents $D_i \subseteq \mathcal{D}$, such that $D_i \cap D_j = G$, for all $i \neq j$ and $\bigcup_i D_i = \mathcal{D}$. We also assume that the search engine serves a set of users $\mathcal{U}$, who are also assigned disjointly to the sites of $\mathcal{S}$. That is, site $S_i$ serves the users $U_i \subseteq \mathcal{U}$, with $U_i \cap U_j = \emptyset$, for all $i \neq j$, and $\bigcup_i U_i = \mathcal{U}$.

We note that the partitioning of documents to local and global sets, as well as the partitioning of users to disjoint sites is only assumed here for simplicity of exposition. In practice the documents and the users may be assigned to sites in any other way and the cost model as well as the query-processing algorithm we present can extend to these cases in a straightforward way.

# 4. COST MODEL

In real systems, a site is a data center hosting a number of racks, where a rack hosts servers and networking equipment. The total cost of ownership (TCO) of a data center is the sum of its capital costs (Capex) and operational costs (Opex). Following the model of Barroso and Hölzle [5], Capex is given by the capital costs of the data center (*e.g.*, construction) and servers, both of them in cost per watt. As the capital cost of constructing a data center scales roughly linearly with the number of watts the data center has been designed for, we assume for simplicity that the total cost for the same number of watts is independent of the number of sites. We also assume that the server cost per watt is constant across site. Such costs include depreciation and interest.

For Opex, we also assume that the data center operational cost per watt is the same across sites. Although differences are likely to exist, modeling such costs is highly complex, and we instead focus on the cost of running the infrastructure as with the model of [5]. As for the operational cost of servers, electricity is the main component, in particular given the trend of higher electricity costs and server power consumption. Consequently, we concentrate on electricity costs.

As we consider settings with multiple data centers, we also include the cost of communicating by looking at bandwidth costs. It is important to observe that under the stated assumptions, looking at the cost of electricity and bandwidth enables us to compare directly the costs of configurations using different numbers of sites.

## 4.1 Model

The cost of a multi-site system is the sum of the individual costs of each site over some period of time. As we mentioned before, the cost of ownership is represented here by the power consumption. We use $W(t,i)$ to denote the power consumption of site $S_i$ at time $t$, and $C_w(\Delta t, i)$ to be the cost of power consumption for site $S_i$ over time $\Delta t$. Hence:

$$Cost(\Delta t) \quad = \quad \sum_i C_w(\Delta t, i).$$

We compute the cost of power consumption of a site:

$$C_w(\Delta t, i) \quad = \quad (\int_{t_1}^{t_1+\Delta t} W(t,i) \cdot dt) \cdot u_w,$$

and $u_w$ is the cost per watt. To account for different functionality, we further split the power cost into different classes, according to the functionalities of the system: $W(t,i) = \sum_f W_f(t,i)$, where $f$ is a functionality of the system, such as crawling and query processing. To estimate the power consumption of each function, we use the following:

$$W_f(t,i) \quad = \quad TOPS(i) \cdot \frac{\ell_f(i)}{c_f(i)} \cdot e_f(t,i),$$

where $TOPS(i)$ is the target number of operations per second (*e.g.*, queries processed, Web pages fetched) that site $S_i$ performs at time $t$; $\ell_f(i)$ is the target latency to perform an operation at site $S_i$; $c_f(i)$ is the capacity in number of simultaneous operations for a server or a cluster, depending on the functionality $f$; $e_f(t,i)$ estimates the power consumption per server or cluster at time $t$. To estimate such a value, we use the models of Fan [19] based on CPU utilization:

$$e_f(t,i) = m_i \cdot (W_{\text{idle}} + (W_{\text{busy}} - W_{\text{idle}}) \cdot cpu(OPS(t,i)), \quad (1)$$

where $m_i$ is the size of a group of servers, $W_{\text{idle}}$ and $W_{\text{busy}}$ is the power utilization of a server when the CPU is idle and busy, respectively, and $cpu(OPS(t,i))$ evaluates to the CPU utilization of a server at time $t$ in site $S_i$. Note that the CPU utilization is a function of the workload at time $t$ given by $OPS(t,i)$.

We use $TOPS(i)$, $\ell_f(i)$, and $c_f(i)$ to estimate the number of servers or clusters necessary for a particular function. We use a server when the processing unit is server. For example, for crawling, we assume that each server crawls individually. For query processing, however, we assume that the processing unit is a cluster because typically systems use document or term partition to increase parallelism when processing a query. Although both document and term partition can potentially cause load imbalance across the servers of a cluster [1, 27], we do not address such issues here, and simply assume that $e_f(t,i)$ evaluates to the total amount of power used at time $t$.

In practice, the values of $TOPS(i)$, $\ell_f(i)$, and $c_f(i)$ can be estimated from demand. For example, practitioners can determine that a given cluster of machines is able to process simultaneously $c_f(i)$ operations keeping the average latency at $\ell_f(i)$, and estimate that the total traffic of a site will be on average $TOPS(i)$. Also note that $e_f(t,i)$ implicitly introduces the current traffic, since the amount of watts depends on the current traffic.

Specializing equation $W_f(t,i)$ to crawling and query processing, we have

$$W_c(t,i) \quad = \quad TPPS(i) \cdot \frac{\ell_c(i)}{c_c(i)} \cdot e_c(t,i) \text{ and}$$

$$W_q(t,i) \quad = \quad TQPS(i) \cdot \frac{\ell_q(i)}{c_q(i)} \cdot e_q(t,i).$$

The rationale for the above equations is the following. For crawling, a server at site $S_i$ can only have a given number of connections open at a time given by $c_c(i)$. Given the number of pages $TPPS(i)$ crawled and the average amount of time to fetch a page $\ell_c(i)$, we determine the total number of servers necessary to crawl. By multiplying by the average amount of power a server uses, we determine the total amount of power necessary for crawling at site $S_i$.

For query processing, we have a similar derivation. To estimate the total amount of power, we multiply the total number of servers

in a query processing cluster and the average amount of power a server uses according to Equation 1. To determine the total number of clusters, we estimate the target arrival rate of queries (*TQPS*$(i)$) and divide by the number of queries per second a cluster can process ($c_q(i)/\ell_q(i)$). There are different ways to determine the number of servers per cluster. In Section 8, for example, we fix a fraction of the index, and each server holds such a fraction.

## 4.2 Adding the cost of networking

In a multi-site system, we also have to add the cost of communicating the sites. As the rates of network circuits and services vary, we measure such a cost using the total number of bytes that we need to transfer over a period of time, and assume a function that converts such a requirement for bandwidth into currency. Typically, the cost of bits per sec (bps) decreases as the total amount of aggregated bandwidth increases. That is, the price of bandwidth often increases sublinearly with the bandwidth contracted.

We then assume that the cost of bandwidth $C_{bw}(t,i)$ is a function of the total number of bytes that site $S_i$ transfers at time $t$. The total cost then becomes:

$$\begin{aligned} \mathit{Cost}(\Delta t) &= \sum_i C_w(\Delta t, i) + C_{bw}(\Delta t) \text{ and} \\ C_{bw}(\Delta t) &= \sum_i C_{bw}(\Delta t, i). \end{aligned}$$

## 4.3 Example: crawling

Suppose we have two systems:

**System 1:** System 1 has one site $S_{11}$, and its Web collection comprises $P$ pages;

**System 2:** System 2 has five sites $\{S_{j2} : j \in \{1,2,3,4,5\}\}$. The Web collection of site $S_{12}$ comprises $\alpha P$ pages, $1 > \alpha > 0.2$, and the other sites maintain $P \cdot (1-\alpha)/4$ pages each. Site $S_{12}$ has the role of a central site, with more computing power than the others.

We use $W_{c_i}(t,j)$ to denote $W_c(t,j)$ for system $i$, and $\ell_{c_i}(j)$ to denote $\ell_c(j)$ for system $i$. We then have that the power consumption to crawl all $P$ pages with System 1 at a rate $p_r = P/\Delta t$, $\Delta t$ being an interval of choice, is:

$$W_1(t) = W_{c_1}(t,1) = p_r \cdot X \cdot \ell_{c_1}(1)$$

where $X$ represents the computation of all other variables. For simplicity, we assume that the power utilization is the same for all servers across all sites. For System 2, we have:

$$W_2(t) = p_r \cdot X \cdot \alpha \cdot \ell_{c_2}(1) + \sum_{i=2,3,4,5} p_r \cdot X \cdot \frac{1-\alpha}{4} \cdot \ell_{c_2}(i)$$

For the sake of simplicity, we assume that System 2 has been designed in such a way that $\ell_{c_2}(i)$ is the same for all $i \in \{2,3,4,5\}$ and equal to $\ell_o$, $\ell_o < \ell_{c_1}(1)$. We have that the difference is

$$W_1(t) - W_2(t) = p_r \cdot X \cdot (\ell_{c_1}(1) - \alpha \cdot \ell_{c_2}(1) - (1-\alpha) \cdot \ell_o),$$

Assuming that $\ell_{c_1}(1) > \ell_{c_2}(i)$, for $i \in \{1,2,3,4,5\}$ and $\alpha > 0$, we have that $W_1(t) > W_2(t)$.

As we reduce the latency of fetching pages, we also reduce the power consumption of servers used for crawling. For example, if we make $\ell_{c_1}(1) = 1s$, $\ell_{c_2}(i) = 0.6s$, $\ell_o = 0.4s$, and $\alpha = 0.2$, then the difference $W_1(t) - W_2(t)$ is 0.56 of $W_1(t)$, which implies a reduction of over 50% in power consumption.

Note that this simple computation does not include potential costs that might arise from having to communicate crawlers in different sites. It does show, though, that if one builds a crawler distributed across a number of sites, and that requires negligible communication among crawlers in different sites, then such a crawler is cheaper compared to a centralized one.

# 5. MOTIVATING FEASIBILITY

In this section we show how our cost model enables assessing the query-processing cost of a distributed architecture. In particular, we analyse a simple scenario that employs a particular network topology and makes other simplifying assumptions. Nevertheless, we believe that our analysis is illustrative and it motivates the algorithm we present in Section 6.

We assume a fully-distributed system in which we have $n$ sites. Users submit queries to the closest site, and the site either processes them locally, or it sends to all other sites. A user request is therefore classified as either local or global, depending on the sites that process the query. Site $S_i$ is able to resolve a query it receives from a user with probability $x_i$. In this example, we assume that $x_i$ is the same across all sites, and we use $x$ to denote the fraction of the total query volume resolved locally.

Following the model of Section 4.1, we have that the cost is the sum of power costs and bandwidth costs, ignoring initial costs and remaining costs of ownership. As each site processes a fraction $x$ of the query traffic received locally, and the remainder is processed by all other sites, we have:

$$\begin{aligned} W_q(t) &= \sum_i W_q(t,i) \\ &= \frac{\ell(n)}{c} \cdot e_q \cdot \sum_i (q_i + \sum_{j:j\neq i} q_j \cdot T_{ji}) \\ &= \frac{\ell(n)}{c} \cdot e_q \cdot \mathit{QPS} \cdot (x + (1-x) \cdot n) \end{aligned}$$

where:

- $q_i + \sum_{j:j\neq i} q_j \cdot T_{ij} = \mathit{TQPS}(i)$, for all $i$;
- $q_i$ is the number of queries per second that users submit directly to site $S_i$, and $\mathit{QPS} = \sum_i q_i$;
- $T_{ij}$ is the fraction of queries that site $S_i$ sends to site $S_j$ for processing;
- $\ell(n)$ is the latency to process a query. We assume that it decreases with the number of sites such that $\ell(n) = k/n$, where $k$ is a constant representing the time to process a query in a single-site system (DQ principle from [7]);
- $c$ is the capacity of a query cluster. We assume that it is constant across sites and independent of the number of sites;
- $e_q$ is the number of watts that query processors consume. For simplicity, we assume that $e_q(t,i) = e_q$ for all $t$ and $i$.

Note that $W_q(t)$ is a value independent of $t$ in this case, and therefore we simply use $W_q$ instead. We hence have that the cost of power considering only the cost of query processing is:

$$C_w(\Delta t) = \left( \int_{t_1}^{t_1+\Delta t} W_q \cdot dt \right) \cdot u_w = W_q \cdot \Delta t \cdot u_w$$

where $u_w$ is the cost of energy given in units of currency per watt-hour ($/*Wh*)[2]. If $\Delta t = 30 \cdot 24$ (number of hours in a month), then we have:

$$C_w(\Delta t) = W_q \cdot 720 \cdot u_w$$

The amount of traffic increases linearly with the number of global queries, and with the number of sites. We hence represent the cost

[2]We use $ to denote the unit of a given currency, *e.g.*, dollars or euros.

of network bandwidth as follows:

$$C_{bw}(\Delta t) = \sum_i C_{bw}(\Delta t, i) = \left( \sum_{i,j:j\neq i} q_j \cdot T_{ji} \cdot b \right) \cdot \Delta t \cdot u_{bw}$$

where $b$ is the average number of bits for each request; $u_{bw}$ is the cost of bandwidth in $ per Mbps per month ; and $\Delta t$ is time in number of months. For this particular example, we have that $T_{ji} = (1-x)$, $q_j = \frac{QPS}{n}$, and $\Delta t = 1$ month:

$$\begin{aligned} C_{bw}(\Delta t) &= \left( \sum_{i,j:j\neq i} \frac{QPS}{n} \cdot (1-x) \cdot b \right) \cdot u_{bw} \\ &= (QPS \cdot (1-x) \cdot (n-1) \cdot b) \cdot u_{bw} \end{aligned}$$

Adding the terms, we have that the total cost is given by:

$$\begin{aligned} Cost(1\text{ month}) &= C_w(1\text{ month}) + C_{bw}(1\text{ month}) \\ &= QPS \cdot (u_w \cdot 720 \cdot (x + (1-x)\cdot n) \cdot \frac{\ell(n)}{c} \cdot \\ &\quad e_q + u_{bw} \cdot (1-x) \cdot (n-1) \cdot b) \end{aligned}$$

Figure 1 plots $Cost(t)$ assuming that $QPS = 1$ (cost of one query per second). It shows how the cost varies for different fractions of locality $x$, assuming that $u_w/u_{bw}$ is 0.01 *Mbps·month/KWh*. Figure 2 shows the optimal number of sites given a ratio and a value of locality $x$.

In the figures, a centralized architecture corresponds to the point with value $n = 1$. Inspecting the graphs we observe that, as the cost of bandwidth decreases, the distributed engine has a lower overall cost. If instead we increase the cost of bandwidth relatively to energy, the cost of a distributed architecture becomes higher, and at some point for no value of the locality parameter a distributed engine has lower costs. In fact, the optimal number of nodes is $C_n \cdot \left( \sqrt{\frac{u_w}{u_{bw}} \frac{1}{1-x}} \right)$, where $C_n$ is a normalization constant that cancels out the unit of $\frac{u_w}{u_{bw}}$ and can be computed from the formula above. Hence, the optimal number grows when locality increases and when the fraction $u_w/u_{bw}$ increases as Figure 2 shows.

It is important to note, though, that the average ratio for main western european countries changed from near 0.02 in 2007 to 0.03 in 2008 (see Figure 1), using industrial power prices from Eurostat [16] and high-end broadband prices from a UK study [30]. As the price per Mbps for large customers is smaller compared to the price for home broadband, we use a transformation factor of 100. This difference causes the ratio of $u_w/u_{bw}$ to increase, which coupled with the fact that bandwidth becomes cheaper while electricity does not, impacts positively on the cost of distributed architectures.

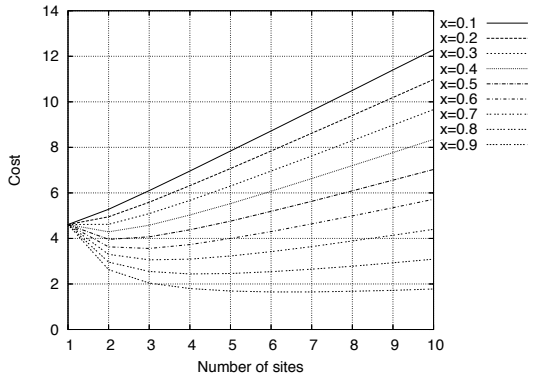


**Figure 1: Cost of processing with a fully-distributed architecture, $u_w/u_{bw} = 0.01$ *Mbps·month/KWh*.**

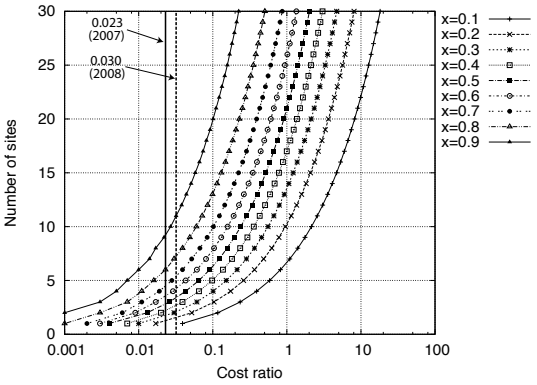


**Figure 2: Optimal number of sites for different cost ratios in *Mbps·month/KWh*.**

# 6. QUERY-PROCESSING ALGORITHM

As we have shown in the previous section, a distributed Web search engine can be more cost-effective than a centralized one if it can process a significant fraction of the user queries locally. To achieve this, we propose a query-processing algorithm for a distributed Web search engine, which aims to maximize the number of local queries without loss in retrieval precision. Furthermore, we explore the trade-off between further increasing locality of query processing and small changes in the quality of results.

## 6.1 The ranking function

Given a query $q = `t_1 \dots t_{|q|}'$, represented by a set of terms, we assume a scoring function $s(d|q)$ that assigns a score to each document $d \in \mathcal{D}$ that contains all terms of the query $q$. The ranking function we consider consists of two components. The first one is a query-independent term $f(d)$, which may combine PageRank score, other link-analysis measures, spam scores, etc. The second component consists of query-dependent terms $g(d|t_j)$ that measure the relevance of each document $d \in \mathcal{D}$ with respect to the query terms $t_j$, and it may combine scores such as TF-IDF, BM25, etc. For a query $q = `t_1 \dots t_{|q|}'$, we consider the ranking function:

$$s(d|q) = w_f f(d) + \frac{w_g}{|q|} \sum_{j=1}^{|q|} g(d|t_j). \quad (2)$$

where $w_f$ and $w_g$ are weights for functions $f$ and $g$, respectively.

We assume that higher values of the functions $f$, $g$, and $s$ indicate more relevant documents. For every document $d$ that does not contain all terms in $q$ we define $s(d|q)$ to be equal to 0 (conjunctive interpretation of queries). Notice that we do not assume anything on the class of functions to be used, $f$ and $g$ can be arbitrarily complex functions, such as neural nets, decision trees, or other, and they can combine an arbitrary number of features extracted from the documents. On the other hand, the general form of ranking function in Equation (2) has some limitations, for instance it does not capture proximity of query terms in the documents. We intend to extend our algorithm for more general classes of ranking functions in our future work.

To answer a query, the search engine needs to find the documents that contain all terms of the query, then evaluate the ranking function on those documents and rank the documents in decreas-

ing order of their score. The top-$k$ documents are returned as an answer to the query.

It is also interesting to consider ranking functions whose second term is the *sum* of query-dependent terms, instead of the average. The algorithm we propose can be extended to such ranking functions, even though the extension is not completely straightforward; and due to space limitations we omit the details in this version.

## 6.2 Overview of the algorithm

We first give an overview of our multi-site query processing protocol and in the next three sections we describe in detail the three main phases of the protocol: indexing, propagation and query-processing. The main idea of the protocol is as follows.

In the first phase of the protocol, the index-construction phase, each site $S_i$ builds an index on the set of documents $D_i$ assigned to it. In addition, a numeric value called *threshold* is computed for each term $t$ of the documents in $D_i$. This threshold is an upper bound on the contribution of the term $t$ to the ranking function for a query that contains the term $t$. Before computing the thresholds, we assume that all sites have consistent global document frequencies so that the computed scores are comparable.

We also assume that a subset $G \subseteq \mathcal{D}$ of good-quality documents is replicated and indexed in all sites (that is, we use partial replication). We refer to the set $G$ as the set of $z$ *globally best documents* in $\mathcal{D}$. We select the set $G$ to be the documents that appear more frequently as results to the queries submitted by users (which can be computed using past logs). We note here that the thresholds computed during the index-construction phase are relative to all documents in $D_i$ *except* those documents that belong in the set $G$.

In the second phase of the protocol, the propagation phase, all sites communicate with each other and exchange information about the thresholds computed at the index-construction phase. As a result, each site acquires complete information about the thresholds of all the terms in all the other sites.

The first two phases of the protocol are performed offline. The third phase is the actual online query-processing phase. Each site $S_i$ handles the queries it receives by first computing the matching set of documents from its local index and ranking them according to the ranking function $s$. Then the site $S_i$ uses the thresholds of the other sites for the query terms to decide whether the results already computed at $S_i$ are identical to the results that would have been obtained by a centralized system.

The main idea is that using properly defined thresholds we can obtain guarantees on the quality of the partial results computed locally at $S_i$. If the guarantees are satisfied the query can be answered directly at $S_i$ without any further communication. If the quality guarantees are not satisfied, then $S_i$ needs to propagate the query to other sites to obtain partial results from those sites, and merge all partial results to a single list. In either case, the results returned to the user for all queries are identical to the results obtained by a centralized system that would have access to all the documents. Next we discuss the three phases of the protocol in more detail.

## 6.3 Indexing phase

Let $G$ be the set of globally best documents among all sites, as defined in the previous section. For a site $S_i$ and a term $t_j$ let $D_{ij}$ be the documents in the set $D_i \cup G$ that contain the term $t_j$. Similarly, let $D'_{ij}$ be the documents in the set $D_i \setminus G$ that contain the term $t_j$. The documents in $D_{ij}$ are organized in an inverted-index data structure. Also for each term $t_j$ at a site $S_i$ we define the upper bound on the scoring function

$$b_{ij} = \max_{d \in D'_{ij}} \{w_f f(d) + w_g g(d|t_j)\}.$$

In other words, we build the index at $S_i$ over the documents of $D_i \cup G$, and compute the upper bound thresholds over the documents of $D_i \setminus G$. For computing the thresholds $b_{ij}$, we assume that the scores $g(d|t_j)$ are computed using the same global document frequencies for all sites $\mathcal{S}$.

## 6.4 Propagation phase

In the propagation phase all sites exchange information about their upper bound thresholds $b_{ij}$ for all terms $t_j$. That is, each site $S_i$ gets to know the upper bound thresholds $b_{i'j}$ for all other sites $S_{i'}$ and all terms $t_j$. Notice that a site must keep upper bound thresholds even for terms that are not in its documents, but storage needs are negligible if compared to the size of the inverted index.

## 6.5 Query-processing phase

Consider a query $q = `t_1 \dots t_{|q|}'$ submitted at site $S_i$, for which we want to compute the top-$k$ documents over the whole collection $\mathcal{D}$. The query-processing algorithm works as follows.

First the query results are computed locally for $S_i$. Let $D_i(q)$ be the documents in $\bigcap_{j=1}^{|q|} D_{ij}$, that is, all the local documents in $S_i$ (documents in $D_i$ plus globally best documents in $G$) that contain all terms of the query $q$. The documents in $D_i(q)$ are ranked according to the ranking function $s(d|q)$, and let $s_{i,k}(q)$ be the score of the $k$-th document.

Now, site $S_i$ computes (locally) an upper bound $b_{i'}(q)$ with respect to each other site $S_{i'}$, $i' \neq i$, as

$$b_{i'}(q) = \frac{1}{|q|} \sum_{j=1}^{|q|} b_{i'j}. \quad (3)$$

This is an upper bound of the score of any document residing at site $S_{i'}$ that could be an answer for the query $q$. If at least one term is missing from the sum in Equation (3) (meaning that in site $S_{i'}$ there is no document containing the corresponding term), then $b_{i'}(q)$ is set to $-\infty$. Let $b_i^*(q) = \max_{i' \neq i} b_{i'}(q)$, that is, an upper bound of any document residing at any site other than $S_i$ that could be an answer for the query $q$.

Now we can test the quality of results computed at $S_i$ with respect to the results of a centralized system by comparing the local score $s_{i,k}(q)$ against the "non-local" upper bound $b_i^*(q)$.

**Local queries.** If $s_{i,k}(q) \geq b_i^*(q)$, then we can guarantee that there is no document in $\mathcal{D}$ with better score than the documents already found in $S_i$. So $S_i$ can just return the documents computed locally. The proof of the following Lemma is simple enough and is omitted from this version of the paper.

LEMMA 1. *For a query $q = `t_1 \dots t_{|q|}'$, a site $S_i$, and the ranking function $s$, assume that $s_{i,k}(q) \geq b_i^*(q)$. Then there is no document $d \in \mathcal{D}$ with score higher than $s_{i,k}(q)$ that it is not returned in the list of results computed locally at site $S_i$.*

**Non-local queries.** If $s_{i,k}(q) < b_i^*(q)$, then there is no guarantee that there are no other better documents for $q$ in other sites. In this case, the query needs to be propagated to other sites, which in turn evaluate the query on their own local indices and return a list of partial results to site $S_i$. The site $S_i$ should then merge all partial results and return the best $k$ documents to the user.

We observe here that large values of $b_i^*(q)$ could be due to only one (or just few) sites, while for many other sites $S_{i'}$ the individual upper bounds $b_{i'}(q)$ might still be small. In such a case, the guarantee of Lemma 1 still holds so that no better documents can be found at sites $S_{i'}$ for which $s_{i,k}(q) \geq b_{i'}(q)$, and thus the query should not be propagated to such sites.

Our algorithm takes advantage of the above observation and compares the minimum score $s_{i,k}(q)$ with each upper bound threshold $b_{i'}(q)$, for $i' \neq i$: if $s_{i,k}(q) < b_{i'}(q)$ the query is propagated to site $S_{i'}$ and partial results are requested from that site, otherwise, the query is not propagated to site $S_{i'}$.

Notice that the amount of communication between sites is small. The query $q$ propagated from $S_i$ to $S_{i'}$ is a string of typically very few bytes. The response of each site $S_{i'}$ to $S_i$ is a list of the best $k$ results for the query $q$ at site $S_{i'}$. Each item in the list contains the url or other id for a document, the value of the ranking function for that document, and possibly a snippet. Also, $k$ can be thought as a small number, say 10.

**Approximate answers.** One way of reducing the amount of inter-site communication is by returning local documents, which are not the same as the results of a centralized system, but have scores that are close enough to the scores of the documents matched by a centralized system. Such approximations can be implemented using the same upper bound thresholds. The idea is to relax the condition $s_{i,k}(q) \geq b_i^*(q)$ with the condition $s_{i,k}(q) \geq b_i^*(q)(1-\epsilon)$, for a carefully selected value of $\epsilon$. If $s_{i,k}(q) \geq b_i^*(q)(1-\epsilon)$ is satisfied the query is answered from the local documents. Otherwise the query is propagated to sites $S_{i'}$ for which $s_{i,k}(q) < b_{i'}(q)(1-\epsilon)$, and local results are merged with the results from those sites.

This scheme guarantees that the documents returned to the query $q$ have scores with relative difference no more than $\epsilon$ than any other document in $\mathcal{D}$ that it is possibly not returned by the algorithm. The value of $\epsilon$ reflects how much we are willing to sacrifice in the quality of the results as a trade-off of answering the query without inter-site communication. The value of $\epsilon$ can be site-dependent and it can be learnt from user evaluations via a machine learning algorithm; we leave this extension as a study for our future work.

**Phrase queries.** A phrase query occurs when a user submits a query with more than one terms, and he/she expects the query to be handled as a whole phrase rather than a set of terms. Examples include "New York", "Warner brothers" "call for papers", etc. Our approach to handling phrase queries is to apply a frequent-itemset mining algorithm and extract from the query log the most frequently submitted multi-term phrases. The resulting phrases are added in the vocabulary as they were singletons and are indexed in all the sites, thresholds are computed for them, etc. During query processing, it is checked if the query contains indexed phrases, and if it does, those phrases are handled as singleton terms. If the user specifies explicitly that the query should be handled as a phrase query (for instance by using double-quotes), and the phrase is not indexed, the local site propagates the query to all other sites. The number of indexed multi-term phrases controls how often queries are propagated to all sites with the tradeoff of a larger index.

# 7. EXPERIMENTAL EVALUATION

We describe here a set of experiments we conducted to evaluate the algorithms introduced in the previous section for the processing of queries in a distributed search engine.

**Dataset.** We consider a setting where the search engine is distributed over five geographic regions and it only receives queries from those same five regions. The $i$-th region has exactly one site $S_i$ which receives queries from users in the same region.

Our proprietary dataset comprises a real workload from the Yahoo! Web search engine, where we consider a random sample of 1% of the queries submitted by users in the five regions during a day, and a random sample of 102 million Web documents from the same five regions. Both the queries and the Web pages have been collected during the same period of time. In Table 1, we show the fraction of documents $|D_i|/|\mathcal{D}|$ assigned to site $S_i$, and the fraction of user query volume $Q_i$ received at site $S_i$. Site $S_2$ has the largest index with 43.11% of the documents, while site $S_1$ receives 43.55% of the total query volume we have considered.

| Site | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ |
|---|---|---|---|---|---|
| $\|D_i\|/\|\mathcal{D}\|$ | 0.2495 | 0.4311 | 0.0649 | 0.1109 | 0.1436 |
| $Q_i$ | 0.4355 | 0.1507 | 0.1011 | 0.1078 | 0.2046 |

**Table 1: Dataset characteristics.**

**Algorithms.** We implemented the distributed algorithm described in Section 6, and we experimented with the following variants:

**B:** Our baseline approach is a straightforward application of the distributed algorithm.

**BC:** The second approach introduces a cache of query results in each site. We assume that the cache has infinite capacity and holds the results of all queries previously submitted to a site. Hence, a site may need to forward queries to other sites only the first time that a given query is received.

**BCG:** The third approach introduces replication of a set of documents $G$ across all sites. The replicated documents are the 500,000 documents that have been retrieved most frequently among the top-300 documents retrieved by a centralized search engine for a set of training queries. The training queries correspond to 1% of the query volume submitted by users in the five regions during one day. The training and evaluation queries were sampled from two consecutive days.

**BCG$\epsilon$:** The last approach introduces the slackness parameter $\epsilon$, which discounts the computed thresholds by the algorithm. We use the slackness parameter only for queries with more than one term, since for one-term queries the thresholds are tight.

For the functions $f(d)$ and $g(d|t)$, we employ scores based on hyperlinks and text content, respectively. Before combining $f(d)$ and $g(d|t)$, we normalize the scores so that their mean is 0 and the standard deviation is 1. Initially, the scores are combined with equal weights $w_f = w_g = 1$ for $f(d)$ and $g(d|t)$, respectively. As discussed later, we also look at different values for $w_f$ and $w_g$.

**Metrics.** We evaluate the performance of the different variants of the distributed algorithm with the following metrics:

**$x$:** the fraction of the query volume across all sites that are resolved locally.

**$x_\ell$:** the fraction of the query volume for which the clicked retrieved documents are local to the site that the query was submitted.

**$T_{ij}$:** the fraction of the query volume in $S_i$ that are forwarded to $S_j$ for non-local resolution.

**$E$:** the fraction of the query volume for which the computed top-10 results are not identical to the top-10 results of a centralized search engine.

**$Cl_g$ and $Cl_\ell$:** the fraction of the query volume for which a centralized search engine ranks at least one clicked document among the top-10 documents. In $Cl_g$ clicked documents belong to any region, while in $Cl_\ell$, clicked documents belong to the same region from which the user submitted the query.

**$Cl_i^+$ and $Cl_i^-$:** the fraction of the query volume for which the computed top-10 results have more ($Cl_i^+$) or fewer ($Cl_i^-$) clicked documents than the results of a centralized engine. The index $i$ in $Cl_i^+$ denotes whether the clicked documents belong to the same region as the user submitting the query ($l$) or not ($g$), as specified above. $Cl_i^+$ and $Cl_i^-$ are both computed with respect to $Cl_i$.

We note here that all of the above metrics refer to the online query-processing part of the algorithm. An important part of the al-

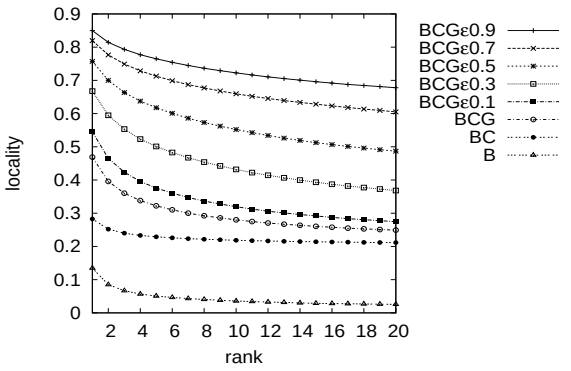


**Figure 3: The fraction $x$ of the volume of queries for which the top-$k$ documents are retrieved locally.**

gorithm refers to the offline phases of index construction and propagation of thresholds. The costs of those offline phases depend on the dynamics of the document collection and on the degree with which the search engine wishes to serve fresh results. We leave this part to further investigation. We choose to use query volume rather than unique queries after caching because the consumed bandwidth in the distributed setting depends on the volume of exchanged data.

We also note that our error measure $E$ is highly conservative. In particular, we count as errors even the cases in which the returned top-10 list differs from the top-10 results of a centralized search engine even by one document. In practice, even if the list is different, the returned list may consist of other relevant documents, or documents that local users are clicking on. While clicks are a biased indication of relevance [22], we use them as a simple indication of the quality of results. Hence, we obtain a different indication of the quality of search results by looking at the fraction of the volume of queries which retrieve among the top-10 documents a higher ($Cl_g^+$ and $Cl_\ell^+$) or a lower number ($Cl_g^-$ and $Cl_\ell^-$) of clicked documents. The fractions are computed relative to the fraction of the query volume $Cl_g$ and $Cl_\ell$, respectively, for which the centralized system retrieves at least one clicked document among the top-10 ranked documents. The clicks were collected from the logs of the Yahoo! Web search engine for the same queries submitted by users in the regions we consider in this work.

**Results and discussion.** Figure 3 shows the locality $x$ achieved by each of the variants of the distributed query processing algorithm, as a function of the number of top-ranked documents that a site returns for a query. The straightforward application of the algorithm (baseline B) has the lowest locality among all the variants we used. Adding a cache of results in each site (BC) results in $x > 0.20$ for all choices of $k \leq 20$. The replication of documents with BCG provides further improvement in $x$ with a notable increase in $x$ for small values of the rank $k$. The slackness parameter in BCG$\epsilon$ improves $x$ as $\epsilon$ increases. For example, when $\epsilon = 0.9$, more than 70% of queries are resolved locally when requesting top 10 documents, with a possible loss of precision. Note that if the latency for non-local queries is $c$ times the latency for local queries due to network, then latency will be on average $x + c(1-x)$ times higher.

Table 2 provides more information on how different values for $\epsilon$ affect locality $x$ and error $E$ at rank $k = 10$. As $\epsilon$ increases, we can see that locality $x$ increases, but at the same time, error $E$ increases too. Moreover, using the set of globally best documents $G$ improves $x$ and reduces significantly $E$ (bold values in columns $x$ and $E$ of Table 2). For all values of $\epsilon$, $Cl_g = 0.0671$ and $Cl_\ell = 0.0604$, because $Cl_i$ depends on the ranking of a centralized engine, and hence, on the weights $w_f$ and $w_g$ of the hyperlinks and text content scoring functions, respectively. The fact that the values of $Cl_g$ and $Cl_\ell$ are similar suggests that users from a given geographic region are likely to click on documents from the same region in the search engine results. Regarding the fraction of the volume of queries with more or fewer clicks among the top-10 ranked documents, we see that both $Cl_i^+$ and $Cl_i^-$ increase as $\epsilon$ increases. The increase in $Cl_i^+$ is due to the sites $S_3, S_4$ and $S_5$, while the decrease in $Cl_i^-$ is mainly due to $S_1$. In particular, BC$\epsilon$ matches more clicked documents among the top-10 documents, either global ($Cl_g^+$) or local ($Cl_\ell^+$), than BCG$\epsilon$ (bold values in columns $Cl_g^+$ and $Cl_\ell^+$ of Table 2). In addition, $Cl_\ell^-$ is lower than $Cl_g^-$ because the query processing algorithm with slackness forwards fewer queries to other sites, and hence, local documents are ranked higher.

In addition, our experiments have shown that the fraction $x_\ell$ is higher than 0.50 for all variants of the distributed query processing algorithm. For instance, $x_\ell$ ranges from 0.54 to 0.95 for B and BC$\epsilon$0.9, respectively. BCG$\epsilon$0.5 also achieves $x_\ell = 0.94$. These results show that the distributed query processing algorithm is successful at retrieving the local documents that users click on.

| | $x$ | $E$ | $Cl_g^+$ | $Cl_g^-$ | $Cl_\ell^+$ | $Cl_\ell^-$ |
|---|---|---|---|---|---|---|
| BC$\epsilon$0.1 | 0.2212 | 0.0034 | **0.0002** | 0.0000 | **0.0002** | 0.0000 |
| BC$\epsilon$0.5 | 0.3940 | 0.2253 | **0.0433** | 0.0319 | **0.0431** | 0.0176 |
| BC$\epsilon$0.9 | 0.6629 | 0.4112 | **0.0710** | 0.0435 | **0.0708** | 0.0207 |
| BCG$\epsilon$0.1 | **0.3197** | **0.0035** | 0.0003 | 0.0028 | 0.0000 | 0.0028 |
| BCG$\epsilon$0.5 | **0.5520** | **0.0596** | 0.0045 | 0.0111 | 0.0040 | 0.0055 |
| BCG$\epsilon$0.9 | **0.7225** | **0.1275** | 0.0067 | 0.0145 | 0.0062 | 0.0057 |

**Table 2: Fraction $x$ of queries resolved locally, error $E$, and fraction of queries with more ($Cl_g^+$ and $Cl_\ell^+$) and fewer clicks ($Cl_g^-$ and $Cl_\ell^-$) in the top-10 ranked documents, for different values of $\epsilon$ in BCG$\epsilon$.**

We gain further insight on the performance of the evaluated approaches by looking at the query traffic between different sites. In Table 3 we consider the case of BCG, where we observe that all sites send most of their traffic to the largest site $S_2$ (see Table 1). Since $S_2$ is the largest site, it is more likely to have documents that rank high for queries submitted to any other site, and respectively, high thresholds. Hence, the algorithm forwards more queries to site $S_2$.

Table 4 shows the effect of varying the weights $w_f$ and $w_g$ of functions $f(d)$ and $g(d|t)$, respectively. We can see that locality $x$ is stable in the cases when $w_g > w_f$, but $E$ increases significantly as the weight $w_f$ of the hyperlink scores decreases, because slackness has a greater impact on the text similarity scoring function $g(d|t)$. We obtain the highest $Cl_g = 0.0710$ when $w_f = 0.5$ and $w_g = 1.0$. In that case, for more than 97% of the query volume with at least a clicked document among the top-10 ones, there is no change in the results between a centralized and distributed system using our algorithm. If we further decrease the weight $w_f$ to 0.10, then we observe a small decrease in $Cl_g$, but also a small increase in $Cl_g^+$, which is mainly due to the higher number of clicked documents for queries submitted to the smaller sites $S_3$, $S_4$ and $S_5$.

Overall, we showed that our algorithm results in improved locality. Moreover, the algorithm successfully retrieves the locally clicked documents in each site. When exploring different values for the weights $w_f$ and $w_g$, we have seen that there is a trade-off between higher locality obtained when $w_f \geq w_g$ and retrieving more clicked documents when $w_g > w_f$.

| | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ |
|---|---|---|---|---|---|
| $S_1$ | - | 0.6894 | 0.5568 | 0.5897 | 0.5724 |
| $S_2$ | 0.6491 | - | 0.5767 | 0.6226 | 0.5948 |
| $S_3$ | 0.6860 | 0.7713 | - | 0.6835 | 0.6602 |
| $S_4$ | 0.6096 | 0.7539 | 0.6237 | - | 0.5722 |
| $S_5$ | 0.5918 | 0.7113 | 0.6042 | 0.5751 | - |

**Table 3: The query traffic $T_{ij}$ sent from site $S_i$ to $S_j$, using BCG. For example, $S_4$ sends $60.96\%$ of its query volume to $S_1$.**

| $(w_g, w_f)$ | $x$ | $E$ | $Cl_g$ | $Cl_g^+$ | $Cl_g^-$ |
|---|---|---|---|---|---|
| (0.5,1.0) | 0.5882 | 0.0334 | 0.0589 | 0.0020 | 0.0103 |
| (1.0,1.0) | 0.5520 | 0.0596 | 0.0671 | 0.0045 | 0.0111 |
| (1.0,0.5) | 0.5256 | 0.1241 | 0.0710 | 0.0185 | 0.0125 |
| (1.0,0.25) | 0.5247 | 0.1727 | 0.0647 | 0.0397 | 0.0116 |
| (1.0,0.10) | 0.5269 | 0.2256 | 0.0500 | 0.0750 | 0.0145 |

**Table 4: Fraction $x$ of queries resolved locally, error $E$, fraction of query volume $Cl$ with at least one clicked document in the top-10 rank documents, and fraction of query volume with more ($Cl^+$) and fewer clicks ($Cl^-$) in the top-10 ranked documents, for different weights $w_f$ and $w_g$ in BCG$\epsilon$ with $\epsilon = 0.5$.**

# 8. INSTANTIATING THE COST MODEL

The variants of the algorithm in the previous section are characterized by the locality $x_i$ of site $S_i$ and pairwise communication $T_{ij}$. In this section we express the cost of query processing of a centralized versus a distributed system for thirty days of operations, using the model of Section 4. We show our results in Table 5.

We estimate the costs of power $C_w$ and communication $C_{bw}$ by using system parameters (CPU utilization, capacity, etc.) that correspond to reasonable design choices, and prices according to what we discuss in Section 4. Energy prices are expressed in \$/KWh and cost of bandwidth in \$/*Mbps · month*.

To simplify calculations, we assume a uniform distribution of queries in time ($QPS(i)$) and a constant latency $\ell_q$; the latter motivated by the fact that each node in a site $S_i$ holds an equal portion of the index. Thus the formulas in section 4 can be re-written as

$$C_w(\Delta t, i) = W_q(i) \cdot \Delta t \cdot u_w(i)$$

where $W_q(i) = QPS(i) \cdot \frac{\ell_q}{c} \cdot e_q(i)$ , $\Delta t = 30 \cdot 24$ is expressed in hours, capacity $c$ and power consumption per node are equal for both centralized and distributed settings.

For communication costs, we now assume a star topology of $n$ sites with center at site $S_c$, and a full-duplex channel between $S_i$ and $S_c$; such an architecture has $n-1$ links as opposed to the $\frac{n(n-1)}{2}$ of a fully connected architecture, thus allowing significant savings in network costs on large deployments.

The virtual traffic between two non-adjacent nodes is projected over the physical links, as in Figure 4, and the final cost is calculated over this flow. For a full-duplex link we then model the rate of bits through each physical channel with center $S_c$ as in the following formula:

$$B_{cj} = \max\big(\sum_k (T_{kj} \cdot \bar{s}_k + T_{jk} \cdot r),\ \sum_k (T_{jk} \cdot \bar{s}_j + T_{kj} \cdot r)\big)$$

where $\bar{s}_j$ is the average number of bits per second for queries sent to $S_j$, and $r$ is the bits per second for responses including snippets.

The optimal center is the one producing the minimum value in

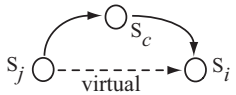


**Figure 4: The byte flow from $S_j$ to $S_i$ goes through the physical links that have one end in the center $S_c$ of the star.**

| Setting | $C_w$ | $C_{bw}$ | $C_{rel}$ |
|---|---|---|---|
| B | 1.483 | 0.019 | 1.502 |
| BC | 1.278 | 0.016 | 1.294 |
| BCG | 1.156 | 0.013 | 1.169 |
| BCG$\epsilon_{0.1}$ | 1.103 | 0.012 | 1.115 |
| BCG$\epsilon_{0.3}$ | 0.970 | 0.010 | **0.980** |
| BCG$\epsilon_{0.5}$ | 0.835 | 0.008 | **0.843** |
| BCG$\epsilon_{0.7}$ | 0.719 | 0.006 | **0.725** |
| BCG$\epsilon_{0.9}$ | 0.652 | 0.005 | **0.657** |

**Table 5: Costs for different settings of the distributed system with respect to the cost of a centralized system.**

the multiplication between $B$ and a symmetric matrix $C$, where $C_{ij}$ represents the cost of a Mbps of bandwidth for a link between sites $S_i$ and $S_j$, and $C_{ii} = 0$:

$$C_{bw}(\Delta t) = \min\big\{\phi_i : \phi_i = \big(\sum_j B_{ij} \cdot C_{ij}\big) \wedge i \in \{1 \ldots n\}\big\} \cdot \Delta t$$

Next we apply our cost model to the variants of the distributed algorithm of Section 6 with weights $w_f = w_g = 1$ for the hyperlinks and text content scoring functions, respectively. The estimated costs in Table 5 are expressed relative to the estimated cost of a centralized system, located in the site with the most favorable energy prices. $C_{rel}$ expresses the ratio of the total cost of a distributed system over the total cost of a centralized system, thus equal to one in centralized settings. Note that we do not consider caching of results for local queries. Such a cache should have a similar impact in both kinds of architecture, since we expect a distributed system to have as many misses as a centralized one (we assume that the same query coming different regions is effectively a different query). It is subject of future work to evaluate the impact of caching further.

It is worth observing that exact values depend on the local cost of power and bandwidth, and a comprehensive market analysis is also out of the scope of this paper. Nonetheless, a trend where the cost of network infrastructure decreases in time is always favorable to our proposed model.

As we apply the variants of the query processing algorithm that increase locality $x$ using caching, replication and the slackness parameter $\epsilon$, the relative cost of the distributed architecture becomes more competitive with respect to that of the centralized architecture. In fact, starting from the variant BCG$\epsilon_{0.3}$, the cost of the distributed architecture is increasingly lower (Table 5). We also see that increase in locality improves not only the communication cost, but power consumption as well. The reason is that increasing locality reduces the average number of sites that process a query.

In all cases, the central site $S_c$ of the star topology was the site with the highest volume of queries submitted by users, namely $S_1$. This result is expected because if $S_c \neq S_1$ the traffic from and to $S_1$ would have to be routed through $S_c$, thus increasing the total communication cost.[3]

[3]For a given load of forwarded queries the power consumption on each site remains constant and only the communication flow impacts the selection of the optimal center.

# 9. CONCLUSIONS AND FUTURE WORK

Distributed search engines yield important benefits such as using less compute resources and exploiting locality to deliver results faster. Our results in this paper indicate that distributed engines are practical and can potentially reduce the costs of query processing. Using a realistic scenario, we have shown that a fully-distributed architecture can reduce costs as the cost of bandwidth decreases faster than the cost of power. To make our argument more concrete, we have proposed and discussed an algorithm for query processing and variants that increase the locality of the system by replicating and allowing to retrieve different results compared to a centralized engine. When evaluating with a real query log, our distributed architecture outperforms a centralized one in costs for reasonable approximation values. More concretely, our results show that the cost reduction can be as high as 15% compared to the centralized case for practical locality levels of 50%. Such a cost reduction comes in addition to other clear benefits such as shorter response time to queries. Also, for most of the queries, we have seen that our algorithm returns the same number of local clicked documents, and in some cases, it may return even more clicked local documents.

Instead of a self-contained set of results, our effort should be viewed as a first step towards the design of realistic distributed Web search engines. As such, it leaves several open questions, which we intend to address in our future work. First we need to investigate the components that were left unspecified: a full implementation of the crawling and indexing phases, the frequency of the threshold propagation phase, etc. In our experiments, we used a star network topology, but as we already mentioned, the optimal choice is one of the parameters of the design process that can be optimized. There are also possible extensions to the query-processing algorithm, enabling it to handle more complex ranking functions, such as one that captures proximity of the query terms in the documents. Finally, we believe that there is potential in using clicks to choose the subset $G$ of global documents more effectively. A better choice of set $G$ might lead to further improvements in the locality of queries.

# 10. REFERENCES


[1] C. S. Badue et al. Analyzing imbalance among homogeneous index servers in a web search system. *Inf. Process. Management.*, 43(3):592–608, 2007.

[2] R. Baeza-Yates, C. Castillo, F. Junqueira, V. Plachouras, and F. Silvestri. Challenges in distributed information retrieval (invited paper). In *ICDE*, pages 6–20, 2007.

[3] R. Baeza-Yates, A. Gionis, F. Junqueira, V. Murdock, V. Plachouras, and F. Silvestri. The impact of caching on search engines. In *SIGIR*, pages 183–190, 2007.

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

[5] L. A. Barroso and U. Hölzle. *The Datacenter as a Computer - an introduction to the design of warehouse-scale machines*. Morgan and Claypool Publishers, May 2009.

[6] M. Bender, S. Michel, P. Triantafillou, and G. Weikum. Design Alternatives for Large-Scale Web Search: Alexander was Great, Aeneas a Pioneer, and Anakin has the Force. In *1st LSDS-IR Workshop*, pages 16–22, 2007.

[7] E. A. Brewer. Lessons from giant-scale services. *IEEE Internet Computing*, 5(4):46–55, 2001.

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

[9] J. Callan. Distributed information retrieval. In W. B. Croft, editor, *Advances in Information Retrieval. Recent Research from the Center for Intelligent Information Retrieval*, chapter 5, pages 127–150. Kluwer Academic Publishers, 2000.

[10] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *SIGIR*, pages 21–28, 1995.

[11] B. B. Cambazoglu, V. Plachouras, and R. Baeza-Yates. Quantifying performance and quality gains in distributed web search engines. In *SIGIR*, pages 411–418, 2009.

[12] N. Cardwell, S. Savage, and T. Anderson. Modeling TCP latency. In *INFOCOM*, pages 1742–1751, 2000.

[13] J. Cho and H. Garcia-Molina. Parallel crawlers. In *WWW*, pages 124–135, 2002.

[14] K. Church, A. Greenberg, and J. Hamilton. On delivering embarrassingly distributed cloud services. In *HotNets*, 2008.

[15] N. Craswell, F. Crimmins, D. Hawking, and A. Moffat. Performance and cost tradeoffs in web search. In *ADC*, pages 161–169, 2004.

[16] EuroStat. European prices for industrial electricity, 2007-8.

[17] J. Exposto, J. Macedo, A. Pina, A. Alves, and J. Rufino. Geographical partition for distributed web crawling. In *Workshop on Geographic Information Retrieval*, pages 55–60, 2005.

[18] T. Fagni, R. Perego, F. Silvestri, and S. Orlando. Boosting the performance of web search engines: Caching and prefetching query results by exploiting historical usage data. *ACM Trans. Inf. Syst.*, 24(1):51–78, 2006.

[19] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *ISCA*, pages 13–23, 2007.

[20] S. Garcia, H. E. Williams, and A. Cannane. Access-ordered indexes. In *ACS*, pages 7–14, 2004.

[21] A. Heydon and M. Najork. Mercator: A scalable, extensible web crawler. *World Wide Web*, 2(4):219–229, 1999.

[22] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR*, pages 154–161, 2005.

[23] H.-T. Lee, D. Leonard, X. Wang, and D. Loguinov. IRLbot: scaling to 6 billion pages and beyond. In *WWW*, pages 427–436, 2008.

[24] J. Li, B. T. Loo, J. Hellerstein, F. Kaashoek, D. R. Karger, and R. Morris. On the feasibility of peer-to-peer web indexing and search. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, Feb. 2003.

[25] S. Melink, S. Raghavan, B. Yang, and H. Garcia-Molina. Building a distributed full-text index for the web. *ACM Trans. Inf. Syst.*, 19(3):217–241, 2001.

[26] W. Meng, C. Yu, and K.-L. Liu. Building efficient and effective metasearch engines. *ACM Comput. Surv.*, 34(1):48–89, 2002.

[27] A. Moffat, W. Webber, and J. Zobel. Load balancing for term-distributed parallel retrieval. In *SIGIR*, pages 348–355, 2006.

[28] A. Moffat, W. Webber, J. Zobel, and R. Baeza-Yates. A pipelined architecture for distributed text query evaluation. *Inf. Retr.*, 10(3):205–231, 2007.

[29] A. Ntoulas and J. Cho. Pruning policies for two-tiered inverted index with correctness guarantee. In *SIGIR*, pages 191–198, 2007.

[30] OfCom. The international communications market 2008, 2008. Chapter 2, International comparative pricing.

[31] S. Orlando, R. Perego, and F. Silvestri. Design of a parallel and distributed web search engine. In *ParCo*, pages 197–204, 2001.

[32] K. M. Risvik, Y. Aasheim, and M. Lidal. Multi-tier architecture for web search engines. In *LAWEB*, pages 132–143, 2003.

[33] P. C. Saraiva, E. S. de Moura, N. Ziviani, W. Meira, R. Fonseca, and B. Riberio-Neto. Rank-preserving two-level caching for scalable search engines. In *SIGIR*, pages 51–58, 2001.

[34] V. Shkapenyuk and T. Suel. Design and implementation of a high-performance distributed web crawler. In *ICDE*, 2002.

[35] L. Si and J. Callan. Relevant document distribution estimation method for resource selection. In *SIGIR*, pages 298–305, 2003.

[36] A. Tomasic and H. Garcia-Molina. Query processing and inverted indices in shared-nothing text document information retrieval systems. *The VLDB Journal*, 2(3):243–276, 1993.

[37] Uptime Institute. Tier classifications define site infrastructure performance. `www.upsite.com/file_downloads/PDF/Tier_Classification.pdf`, 2006.

[38] J. Zhang and T. Suel. Efficient query evaluation on large textual collections in a peer-to-peer environment. In *5th IEEE International Conference on Peer-to-Peer Computing*, pages 225–233, 2005.


View publication stats