Quantifying Performance and Quality Gains in Distributed Web Search Engines

B. Barla Cambazoglu
Yahoo! Research
Barcelona, Spain
barla@yahoo-inc.com

Vassilis Plachouras
Yahoo! Research
Barcelona, Spain
vplachouras@acm.org

Ricardo Baeza-Yates
Yahoo! Research
Barcelona, Spain
rbaeza@acm.org

ABSTRACT

Distributed search engines based on geographical partitioning of a central Web index emerge as a feasible solution to the immense growth of the Web, user bases, and query traffic. However, there is still lack of research in quantifying the performance and quality gains that can be achieved by such architectures. In this paper, we develop various cost models to evaluate the performance benefits of a geographically distributed search engine architecture based on partial index replication and query forwarding. Specifically, we focus on possible performance gains due to the distributed nature of query processing and Web crawling processes. We show that any response time gain achieved by distributed query processing can be utilized to improve search relevance as the use of complex but more accurate algorithms can now be enabled for document ranking. We also show that distributed Web crawling leads to better Web coverage and try to see if this improves the search quality. We verify the validity of our claims over large, real-life datasets via simulations.

Categories and Subject Descriptors

H.3 [**Information Systems**]: Information Storage and Retrieval

General Terms

Experimentation, Performance

Keywords

Data centers, distributed query processing, index partitioning, Web crawling

1. INTRODUCTION

Designing a large-scale Web search engine requires many decisions to be made. These decisions are affected mainly by the users of the search engine, queries submitted, evolution of the Web, and trends in hardware. The goal in a successful design is to have a high-performance search engine that satisfies user needs in terms of the quality of presented results. In this context, performance typically refers to achieving high query processing throughput with low response times. Quality refers to precision and recall.

Achieving high performance and result quality becomes more challenging as the Web grows and evolves, user bases and query traffic increase, distribution of users and hence queries become geographically more diverse, and user expectations get higher. Scaling with all these ever-growing factors necessitates decentralization of the search process [1]. Decentralization can be achieved by distributing resources of a central search engine over multiple, geographically distant data centers. However, distributing the search process brings new design issues such as the optimum number and placement of data centers, assignment of crawling, indexing and query processing tasks to data centers as well as clustering and mapping of user queries to data centers.

In this work, we adopt a geographically distributed search architecture [3, 14], based on partial index replication and selective query forwarding between search sites. In this architecture, unlike centralized search engines, where the index is stored in a single, large data center, the index is partitioned into smaller, non-overlapping parts and distributed over multiple, geographically distant search sites. Partitioning of the index is in compliance with the geographical locality of data centers (e.g., a data center in France crawls and indexes documents in Europe). Additionally, popular or commonly accessed parts of the index are replicated on all search sites. Query processing is also different than centralized search engines, where all user queries are submitted to a single data center. In this architecture, users are mapped to local data centers according to their geographical locality (e.g., European queries are submitted to the data center in France). Local sites may forward queries to other search sites that are likely to contain relevant results for the query.

The search architecture we discuss differs from federated search architectures [5, 6] in the literature in a number of ways. First, the search sites are not autonomous as it is typically the case in federated search, but are parts of a single, large, collaborative system. Second, resources and algorithms are more homogeneous, e.g., all sites execute the same ranking algorithm. Third, there are infrastructural advantages, e.g., we can have a dedicated network between search sites. Fourth, distribution of content in data centers is explicitly managed, i.e., there is an offline mechanism for distributing the content over the search sites.

A related but still different architecture is P2P networks [4,

SIGIR'09, July 19–23, 2009, Boston, Massachusetts, USA.

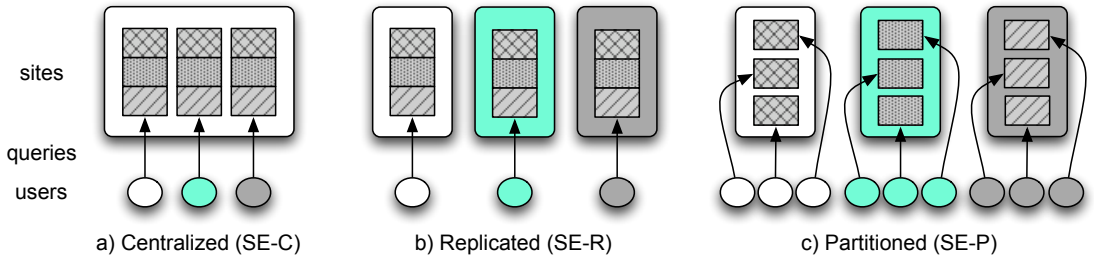

Figure 1: Query processing in different search architectures.

15]. A geographically distributed search engine is different than a P2P system in a number of ways. First, data centers are static and have high availability, unlike the peers that are quite dynamic and volatile in P2P search. Second, the number of data centers is not very high (e.g., several), whereas the number of peers can grow drastically (e.g., millions) in P2P. Third, P2P systems are low cost systems, but constructing data centers can be quite expensive and hence require more intelligent design choices to be made.

The focus of this work is on performance and quality gains achievable by geographically distributing the search and crawling processes. In Section 2, we describe a taxonomy of geographically distributed search architectures. Later, we investigate the validity of the following claims:

- The index is partitioned among multiple data centers and local queries are processed over smaller indices. Under equal query throughput, this leads to gains in query processing time (Section 3.2.1).
- Users are more likely to be closer to local search sites than a central site. Hence, there are gains in network latency while transferring queries and results between users and search sites (Section 3.2.2).
- Any performance gain can be converted into improvement in search relevance by employing costly but more accurate ranking algorithms (Section 3.3).
- Proximity of data centers to Web sites increases the download rates and hence the Web coverage. This may lead to a positive impact on relevance (Section 4.2).

2. WEB SEARCH ARCHITECTURES

In this section, we briefly summarize the query evaluation process in search engines, introduce a taxonomy of search architectures, present our assumptions about resource allocation, and discuss the trade-offs in different architectures.

2.1 Query Processing

In large-scale search engines, due to tight response time requirements (e.g., a few hundred milliseconds), queries are processed on large parallel systems, i.e., clusters composed of many processors. As an offline process, an inverted index [10, 16] is built over a Web page collection. This index is distributed over the processors in a cluster. For distributing the index, two possible approaches taken are term-based or document-based partitioning [13]. In practice, due to better load balancing and higher index availability, document-based partitioning is adopted by almost every major search engine. In this approach, the entire Web collection is partitioned such that each processor will have a sub-collection and construct a local index over it. To achieve concurrency and increase query throughput, search clusters are replicated. Also, results of popular queries and frequently accessed parts of the index are cached in memory to decrease the query processing time.

Processing of a query on a search cluster follows several steps. A master processor receives the query and checks the result cache. If the query result is cached, the query is answered without any processing. Otherwise, the master submits the query to all worker processors in the cluster. Each worker fetches from disk the inverted lists corresponding to query terms that are not already cached in memory. Due to strict processing time requirements, large-scale engines employ a boolean AND model, i.e., inverted lists are intersected so that only the documents that contain all query terms are selected. Then, selected documents are scored using various techniques, and top-scored k documents are returned to the master. Finally, the master sorts all results in decreasing order of scores and returns the top k results to the user.

In query processing, there are two types of costs: a fixed cost independent of the number of processors and a cost linearly scaling with increasing processor count. The average processing time T can be represented by the simple equation

$$T = a + b \times I_{\text{avg}}/K, \tag{1}$$

where a is the fixed time cost, b is the time cost of processing a byte of an inverted list, I_{avg} is the average size (in bytes) of an inverted list in the global index, and K is the number of processors in the cluster. If inverted lists are in memory, decompression of postings and list intersection forms the scalable cost. If inverted lists are stored on disk, query processing time is dominated by accessing the data on disk. In this case, the fixed cost is the cost of disk seeks and rotational latency while the scalable cost is the cost of transferring lists into memory. We will use equation (1) throughout the cost analysis in Section 3.2.

2.2 Taxonomy

Based on the way the index is stored, we classify search engine architectures into three as centralized (SE-C), replicated (SE-R), and partitioned (SE-P). Fig. 1 illustrates these architectures. In the figure, inner, small boxes represent search clusters, and patterns represent different ways of index replication/partitioning. Later, in this section, we show that these three architectures are special cases of a more general architecture based on partial replication and query forwarding (SE-H).

The centralized architecture (SE-C) has been heavily used in the past by large Web search engines and is still adopted by small-scale engines. In this architecture, the entire Web index is stored in one, central site. The index is replicated

Table 1: Allocation of computational resources in different search engine architectures

Arch.	Index size	# of sites	# of clusters per site	# of processors per cluster
SE-C	I	1	C	K
SE-R	I	S	C/S	K
SE-P	I/S	S	C	K/S
SE-H	I'	S	C'	K'

locally over multiple search clusters located within the site. All user queries are submitted to this single site, where each query is completely processed by an individual cluster.

The architecture based on index replication (SE-R) is currently adopted by some major Web search engines. The entire index is replicated over multiple (e.g., several), geographically distant data centers. The index is also locally replicated in each data center as in the case of SE-C. There is a static, one-to-one mapping between user queries and search sites, i.e., each site is responsible for processing a subset of user queries. Typically, this mapping is based on geographical proximity of users to data centers. The query evaluation is over the entire index, and each query is processed independently by a separate cluster as in SE-C.

In the SE-P architecture, the global document collection is partitioned into smaller, non-overlapping sub-collections such that each data center has a different sub-collection assigned to it. Hence, the entire index is partitioned and distributed over multiple, geographically distant search sites. Each site replicates its local sub-index on its clusters. Mapping of queries to sites and processing is similar to SE-R.

Despite its performance advantages, low relevance is a limitation for SE-P. In theory, given a perfect ranking function, search relevance in SE-P can never be better than that of SE-C as, in SE-P, only a subset of the documents are evaluated for relevance. There are two ways to alleviate this problem: taking popular data to queries or taking some queries to sites with relevant data. The first is an offline process that requires replicating a portion of the index (e.g., globally popular documents) on data centers. The second is an online process based on forwarding some queries (e.g., queries that are expected to have relevant documents on other sites) to non-local data centers for additional processing. These two solutions lead to a hybrid architecture [3, 14], which we refer to as SE-H. We should note that SE-R is a special case of SE-H with full replication. Also, SE-P can be seen as a simple version of SE-H with no replication and query forwarding. We will go into more detail on SE-H in Section 3.

2.3 Allocation of Computational Resources

To be able to make a fair trade-off analysis, we need to have some assumptions about the allocation of computational resources in different architectures. We assume that the centralized architecture has an index of size I, replicated over C search clusters, each containing K processors. We also assume that C and K values are optimally chosen for best performance, i.e., we assume the optimality of resource allocation in the centralized architecture.

In practice, selecting the optimal K value depends on many factors such as the average inverted list size, hardware parameters, and properties of queries. In any case, the optimal value is the minimum K value that satisfies a given maximum tolerable average query processing time T^* for a percentage (herein, we assume 100%) of queries, i.e., it should always take $T \leq T^*$ units of time to process queries on a cluster of K processors, on average. Choosing the optimal K value requires a study on its own as query processing time does not linearly scale due to the fixed time costs mentioned before. If K is known, choosing the optimal C value, which depends on T and query arrival rate, is relatively easier as it can be chosen such that a given constraint on the minimum permitable query throughput is satisfied.

Table 2: Performance of the three search engine architectures in various criteria

Arch.	Through.	Response time	Search relevance (using a perfect ranker)
SE-C	C/T	$\ell_{\mathrm{C}} + T$	$R_{\mathrm{SE-C}}$
SE-R	C/T	$\ell_{\mathrm{D}} + T$	$R_{\mathrm{SE-R}} = R_{\mathrm{SE-C}}$
SE-P	SC/T	$\ell_{\mathrm{D}} + T$	$R_{\mathrm{SE-P}} < R_{\mathrm{SE-C}}$
SE-H	SC'/T'	$\ell_{\mathrm{D}} + T'$	$R_{\mathrm{SE-P}} \leq R_{\mathrm{SE-H}} \leq R_{\mathrm{SE-C}}$

In the rest of the analysis, to be fair, we fix the number of processors to $C \times K$ so that the total computational power available to each architecture is the same. In SE-R and SE-P architectures, to simplify the analysis, we do not consider the load imbalance in search sites; instead, we assume that search sites have similar index sizes and query traffic.

Given these assumptions, Table 1 summarizes the optimal distribution of computational resources in different architectures. In SE-R, since every search cluster stores and processes the entire index, clusters are composed of K processors. Hence, there are S sites each with C/S clusters. In SE-P, on the other hand, search clusters have K/S processors as each site has an index of size I/S, and there are C search clusters. Note that this allocation preserves our assumption about optimality of the K value in SE-C. We delay the discussion about SE-H to Section 3.

2.4 Performance Trade-offs

The success of a search engine is proportional to the number of users attracted. Attracting more users requires good performance, which is usually determined by high query throughput, low query response time, and high search relevance. Table 2 displays a comparison of the search architectures in terms of these factors, which we discuss below.

2.4.1 Response Time

Response time is the time passed between the submission of a query and the display of the results. Typically, search engines try to keep it below a few hundred milliseconds. In general, the response time is composed of two components: network latency and query processing time.

In our context, network latency accounts for the time spent over the network to receive the query from the user plus the time spent over the network to send the search results to the user. Latency is determined by both the geographical distance and the network distance between the user and the search site. Query processing time, on the other hand, is determined by the architecture of the search engine, available computational resources, and the query.

For our response time analysis, we assume that, in SE-C,

the average query processing time is T, which we define as the average time needed to process a single user query over an index of size I using a single search cluster containing K processors. Since the I/K ratio is fixed, query processing time is T also for SE-R and SE-P architectures. However, since there are multiple, geographically scattered search sites and assignment of user queries to sites is based on proximity, the average network latency (ℓ_D) of distributed architectures SE-R and SE-P is lower than the average network latency (ℓ_C) of SE-C. Therefore, SE-R and SE-P have lower query response times, on average.

2.4.2 Query Throughput

Query throughput is the number of queries processed per unit of time by the search engine. The peak, sustainable throughput determines the query drop rates under high traffic. Throughput is typically increased by having replicas of the same search index on multiple, identical search clusters.

Among the architectures, SE-C and SE-R can process at most C queries concurrently (assuming no multi-threading) since there are C search clusters in these architectures. On the other hand, in SE-P, it is possible to achieve a relative throughput increase by a factor of S because, although smaller, there are more ($S \times C$) clusters, which can process queries at the same rate provided by SE-C and SE-R.

2.4.3 Search Relevance

In measuring relevance of search results, various metrics can be used (e.g., precision at k, mean average precision, and discounted cumulative gain). Search relevance is a very important criterion for the success of a search engine as it is directly related to user satisfaction and revenues.

Among the architectures, SE-C and SE-R are expected to achieve the same search relevance as queries are processed under the same conditions, i.e., evaluated over exactly the same index. Assuming a perfect ranking algorithm is available, SE-P is expected to produce less relevant results than SE-C no matter how the index is partitioned and users are assigned to sites. This is due to the fact that queries are evaluated locally over only a small portion of the entire index and relevant documents available on non-local sites cannot be included in query results.

3. ANALYSIS OF SE-H

3.1 Architecture

In SE-H, index replication requires deciding on what parts of the index should be replicated on which data centers. Similarly, query forwarding requires deciding on which queries should be forwarded to which data centers. A recent work [3] has proposed a naïve technique for index replication and a novel algorithm for query forwarding in SE-H. The replication algorithm used is naïve because it is simply based on replicating frequently accessed parts of the index on all data centers, without taking the relevance relation between the queries and documents into account. The query forwarding algorithm is novel because it has a filtering step which prevents forwarding of queries to sites with irrelevant content. In SE-H, given a replication factor β, a subset of size βI of the global index is replicated on all sites while the rest of the index is disjointly partitioned such that a sub-index of size $(1-\beta)I/S$ is assigned to each center. Hence, the total index size stored on a site is $I(\beta + (1-\beta)/S)$. In addition to partial replication of the index, a fraction α of non-local queries is forwarded to γ remote sites for further processing.

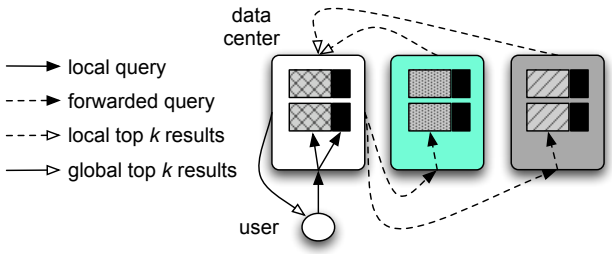

Figure 2: Query processing in SE-H.

In SE-H, queries are processed as follows. According to assignment of users to data centers, the user query is submitted to a local search site. This site evaluates the query over both the local and replicated index generating the local top k results for the query. Depending on the query and a forwarding criterion, the local site may decide to forward the query to one or more remote sites. One forwarding criterion is whether a remote site holds documents with higher scores than some of the local top results for the same query. If the query is not forwarded, the local top k results are simply returned to the user. If the query is forwarded, the remote sites that receive the query evaluate the query over their local indices and each site generates a remote top k result set. These results are returned to the local site, where they are merged into a global top k list, which is then returned to the user. Fig. 2 illustrates query processing in SE-H.

3.2 Cost Models

Replication and query forwarding both help improving search relevance in SE-H. However, increasing the size of the replicated index results in an increase in index sizes. Hence, search clusters in SE-H now have to process a larger index, which means an increase in average query processing time. Query forwarding, on the other hand, incurs both query processing overhead on remote sites and a network latency overhead on the response time of the forwarded queries.

In what follows, we develop analytical models to compare SE-H and SE-C architectures. Such a comparison is possible either by fixing available hardware resources and investigating the query processing performance or by fixing the performance and observing the savings in hardware. Here, we take the first approach while the authors of [3] take the latter. In what follows, we compare the two architectures in terms of query processing performance and network latencies.

3.2.1 Performance Analysis

In SE-H, a search site processes its queries over an index of size $I((1-\beta)/S + \beta)$. Also, for every query submitted to the search system, the search site has to process additional $\gamma\alpha$ queries over an index of size $I(1-\beta)/S$. These additional queries are those forwarded by other search sites. There is no need to evaluate these queries over the replicated part of the index as this is already done at the local site.

According to the discussion above, on average, a search cluster needs to process an index of size $I((1-\beta)/S + \beta) + \gamma\alpha(1-\beta)/S)$ per query submitted to the distributed system. For the SE-H architecture, let K', C', and T' respectively be the number of processors per cluster, number of clusters per data center, and average processing time for a query. By (1), we obtain the average query processing time in SE-H as

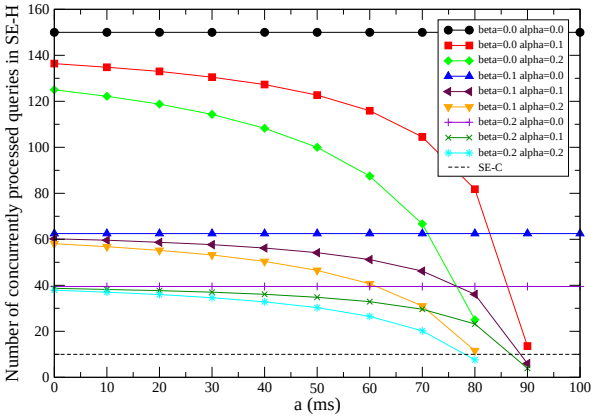

Figure 3: Number of queries concurrently processed in SE-H with varying replication rate (β) and query forwarding rate (α) under the query processing time constraint $T'=T$.

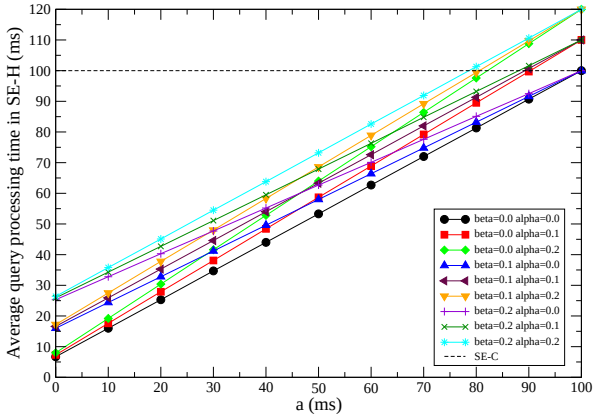

Figure 4: Effect of different configurations of replication rate (β) and query forwarding rate (α) on query processing time (T') in SE-H (assuming $T=100$ms).

$$T' = a(1+\gamma\alpha) + \frac{bI((1-\beta)(1+\gamma\alpha)+S\beta)}{SK'}. \quad (2)$$

Using this formula in our analyses requires setting parameters such as a, b, and I, which are hardware- and collection-specific. Instead of fixing these parameters to specific values, we assume a generic centralized search engine setup, where the average query response time is $T=100$ms, and each cluster (we assume $C=10$) is composed of $K=100$ processors. For SE-H, we use a reasonable value of $S=15$ search sites and assume $\gamma=1$. To measure the relative gains in SE-H, we gradually increase the cost component a and observe the performance with varying β and α values.

Since SE-H is supposed to work on a smaller index but with the same computational power available in SE-C, we can expect a performance improvement over SE-C. For performance analysis, one alternative is to fix the average query processing time ($T'=T$) and observe the increase in the number of concurrently processed queries. This is possible by constructing many, small-size clusters in SE-H.

We know that, in SE-C, C queries can be concurrently processed and answered in T units of time on clusters composed of K processors. If we fix the average query processing time in SE-H, then SC' concurrent queries are answered in T unit of time on clusters composed of K' processors. Setting $T'=T$ and using equation $CK=SC'K'$ (the total hardware is fixed), we obtain the number of queries that SE-H can concurrently process as

$$SC' = \frac{CS(bI - aK\gamma\alpha)}{bI((1-\beta)(1+\gamma\alpha)+S\beta)}. \quad (3)$$

Fig. 3 shows the behavior for various replication and query forwarding rates for the number of queries that can be processed concurrently in SE-H. Note that, for SE-C, this number is fixed and equal to $C=10$ for our setup. For SE-H, if a is small, then performance is mainly determined by the replication rate. With sample values of $\alpha=0.1$ and $\beta=0.1$, it is possible to concurrently process six times more queries than SE-C. Performance degrades only for very high a values and when query forwarding rate increases.

Although this analysis is useful, it is more interesting to develop a cost model that fixes the query throughput to that of the centralized architecture and compute the average query processing time. Furthermore, it may not be that useful to increase the throughput unless query traffic also increases. By reducing the response time, however, we may obtain more time to utilize complex ranking functions and increase the relevance in SE-H.

This analysis requires fixing the concurrently processed query count. Now, in SE-H, C concurrent queries are answered in T' units of time using C' clusters each containing K' processors. This can be achieved by fixing the number of clusters in SE-H to $C'=C/S$ so that the number of concurrently processed queries is equal in both architectures. This enforces $K'=K$. Since we know K', we can now plot the query processing time in SE-H using equation (2).

Fig. 4 shows the behavior for sample β and α values as a varies. In practice, if posting lists are in memory, a is expected to be small (e.g., $a<0.10T$). If they are on disk, a is relatively larger due to the overhead of disk seeks, which is not parallelizable. As seen in the figure, for low a values, considerable time gain is possible in query processing.

We should note that query processing times can be further decreased by using fewer clusters consisting of larger number of processors ($K'>K$) without violating the throughput constraint (via queuing of queries), but we do not go into this analysis due to space limitations.

3.2.2 *Network Latency Analysis*

The cost analysis for network latency is relatively simpler. In the centralized architecture SE-C, network latency is simply twice the latency between the user and the search site, i.e., $L=2\ell_C$. In SE-H, on the other hand, the time spent on the network depends on the type of the query. For forwarded queries, there is the overhead of network latency between the local site and remote search sites. If the average latency between two search sites is denoted by ℓ_R, the average network latency cost in SE-H is $L'=2\ell_D+2\alpha\ell_R$. Hence, the latency difference between SE-H and SE-C is

$$L'-L = 2(\ell_D + \alpha\ell_R - \ell_C). \quad (4)$$

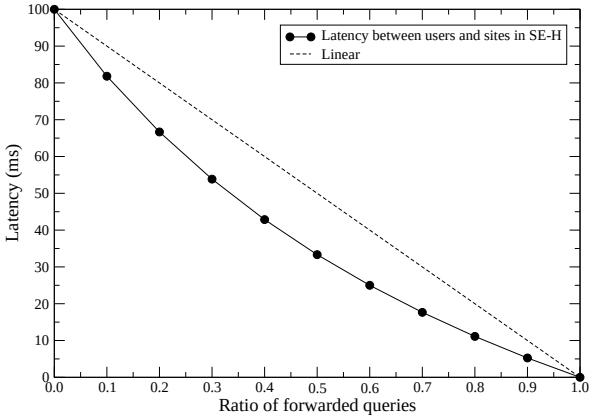

Figure 5: Minimum ℓ_D values required (feasible network latency) in SE-H to have a gain in network latency against SE-C (assuming $\ell_C = 100$ms) with increasing α, under the worst-case scenario, where $\ell_R = \ell_D + \ell_C$.

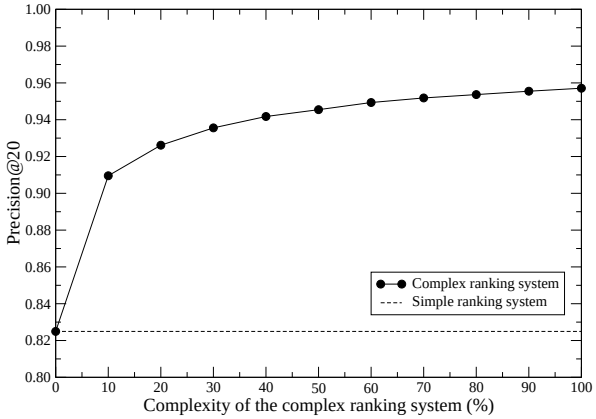

Figure 6: Variation in precision@20 as the cost of the ranking function increases.

To make a further, worst-case scenario analysis, assume that the data center in SE-C is used as one of the local data centers in SE-H (a quite reasonable assumption). We can simply assume Euclidean distances in place of latency values as round-trip time correlates better with geographical distance [11]. The worst case scenario is observed when $\ell_R = \ell_D + \ell_C$, i.e., when the user is on the line between the local site and the central site. To have a gain in average latency, we must have $L' - L < 0$, which implies

$$\ell_D < \frac{1-\alpha}{1+\alpha}\ell_C. \quad (5)$$

According to this equation, for example, if half of the queries can be processed locally ($\alpha = 0.5$), the average latency between the user and a local site in SE-H must be at least three times less than the average latency between the user and the central site in SE-C. Fig. 5 displays the feasible ℓ_D values necessary to have a latency gain in SE-H relative to SE-C (assuming $\ell_C = 100$ms) as α varies.

3.3 Search Relevance

As illustrated in the previous two sections, it is possible to have gains in both query processing time and network latency in the SE-H architecture, relative to SE-C. These gains can now be used to improve the search relevance by employing more costly but accurate ranking functions in query processing. That is, given the same query response time constraint, we can show that SE-H has higher relevance than SE-C since it can afford a high-quality ranking function.

To simulate a cheap ranker for SE-C, we used a linear combination of a BM25 variant with a link analysis metric. For the complex ranker in SE-H, we used a machine learned ranking system with many scorers. We adjusted the complexity of this ranker by simply increasing the number of scorers. The execution cost linearly scales with the increase in complexity as the scorers has similar cost.

To form a base-line set of top documents, we collected the top 20 results for 5000 queries from a commercial search engine. To evaluate the performance of the ranking functions, we blended 200 documents selected by the cheap scoring function into the base-line set. We ranked all 220 documents using a complex ranking function composed of 1000 scorers, at every 100 scorer measuring precision@20 values.

Fig. 6 shows the behavior for precision@20 as the complexity of the ranking system increases. Zero complexity corresponds to the case where we use the cheap ranking function. Full complexity corresponds approximately to an execution cost of 100ms. This result indicates that considerable relevance improvements are achievable in SE-H even by introducing little complexity into the ranking function utilizing the gains in response time (i.e., the gains in both query processing time and network latency).

4. WEB COVERAGE VERSUS RELEVANCE

Large Web search engines continuously crawl the Web and build search indices. In SE-C, the entire Web is crawled by a central system and stored in a single data center. An advantage of distributed search architectures over SE-C is the distributed nature of the crawling process, i.e., data centers crawl the Web pages that are of interest to their users. Hence, the distances between search sites and crawled Web sites are lower compared to the centralized case. As pages will be downloaded faster, page refresh rates can be improved and Web coverage can be increased. Here, we are interested in the latter.

A recent study [7] has investigated performance gains in distributed Web crawling via real-life performance experiments over geographically distributed sites based on sampling of above-mentioned latency and bandwidth values. It reports considerable throughput increase for systems distributing the crawling process. Obviously, given a fixed amount of crawling time, the speedup in download rates leads to better coverage of the Web.

Herein, we investigate the claim that better Web coverage in distributed search engine architectures leads to better search quality. In practice, both Web crawling and processing of queries are continuous. For evaluating the claim, we can assume two timelines, each starting at the same time. The first timeline contains the occurrence of queries. The second contains the discovery of Web pages. At a specific time-point t (checkpoint), by measuring the "overlap" be-

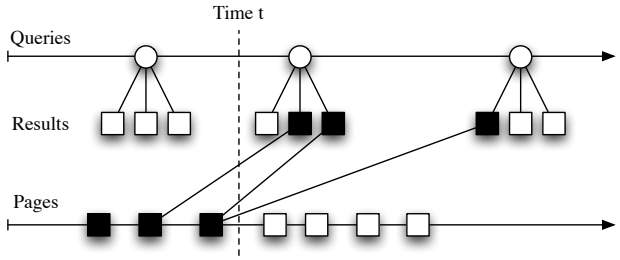

Figure 7: Overlap between future query results and documents crawled in the past.

tween the search results that will be displayed within the time interval $[t \ldots \infty)$ and the pages crawled during the time interval $[0 \ldots t]$, the effect of Web coverage on search relevance can be measured. In the example in Fig. 7, 50% of the future query results can be served by the repository, i.e., an overlap of 0.5 at time point t. For our analysis, however, we assume that queries are repeated and evaluate the relevance at every checkpoint over the entire set of queries.

4.1 Dataset and Setup

We use a random sample of 102 million Web pages, crawled in September 2007. Pages are partitioned into five distinct geographical regions, based on the location of the Web server holding the document and the content of the document.

As a ground-truth for relevant results, we use clicks obtained from a commercial search engine. We assume that a document that was clicked for a given query is relevant. For simplicity, we ignore the rank at which the document was clicked. We collect pairs of queries and clicked documents for those queries by users from a particular geographic region. For example, queries from users whose IP address is in France, or queries submitted to the French site. We form our document collection such that it contains all relevant documents for queries we process.

We measure search relevance as the average reciprocal rank of relevant documents. Web pages are ranked using a linear combination of a BM25 variant and a link analysis metric. The parameters of BM25 and the weights for the linear combination have been set to maximize the average reciprocal rank achieved by the centralized system.

In the experiments, we investigate the effect of increased crawling throughput, crawling order, and region boosting on search quality. We report the relevance values at 10 equally spaced time points (checkpoints). The 10th checkpoint corresponds to the time that the fastest system downloaded all Web pages. That is, at this checkpoint, only a subset of the collection will be downloaded by slower systems.

4.2 Experiments

For throughput experiments, we use four different download throughput (D) values for the centralized crawling scenario and one for the distributed case (obtained from [7]). Fig. 8 shows the accuracies achieved with varying throughput. According to the results, by the time a distributed engine downloads all pages, the fastest centralized engine only has about 3/4 of the accuracy of the distributed engine.

The previous experiment considers a random crawling order. However, more intelligent crawling orders can be employed so that more important pages are downloaded earlier [2, 8, 9, 12]. In our experiments, we try different crawling orders: random, decreasing page size, increasing page size, increasing URL depth, and decreasing importance (as measured by a link analysis metric).

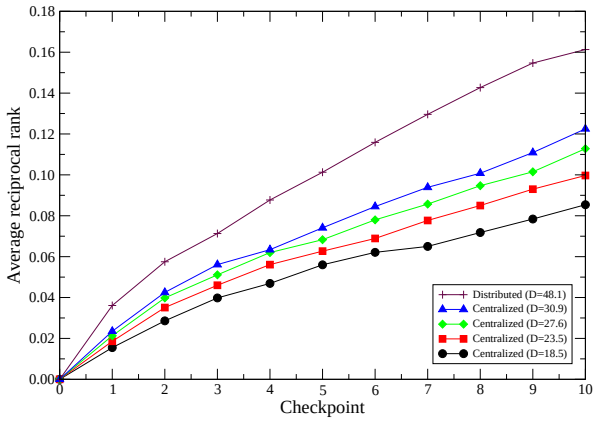

Figure 8: Effect of throughput on result quality.

Fig. 9 shows the impact of different crawling strategies on relevance. In general, ordering by URL depth or page importance achieve good relevance rates very early. Specifically, 95% of the accuracy is already achieved by the page importance metric when only 10% of pages are crawled. We also display the results for varying throughput rates in Table 3. This table shows that centralized engines can also achieve high precision values by employing a link analysis metric, approaching distributed engine's performance. This is due to the power-law distribution in important pages, i.e., most important pages can be downloaded very early without being affected much from the download throughput.

Partitioning documents geographically results in some form of regional document boosting if no query forwarding is employed. But, the same effect can also be achieved by the centralized engine. We conduct an experiment to observe the effect of region boosting. Fig. 10 shows the performance of different architectures (with $D = 48.1$). It is seen that SE-P performs better than SE-C since many irrelevant documents are automatically filtered due to partitioning. However, SE-C with explicit boosting of documents performs slightly better.

5. CONCLUSIONS

We have shown via analytical models that there are considerable performance gains possible by distributed search architectures relative to a centralized architecture. Also, it is shown that the performance gains may lead to gains in search quality as they let more complex ranking functions be employed. Finally, we have investigated the effect of distributed Web crawling on search relevance.

Although, in this work, the distributed architecture (SE-H) is shown to be superior to the centralized architecture (SE-C) in terms of performance, this mainly depends on the amount of redundant work performed, i.e., the processing on the replicated index and remote query processing. Hence, there are two main research directions that can be pursued for further improving the performance in SE-H. First, novel replication models are needed. These models should take into account the current mapping of users to data centers

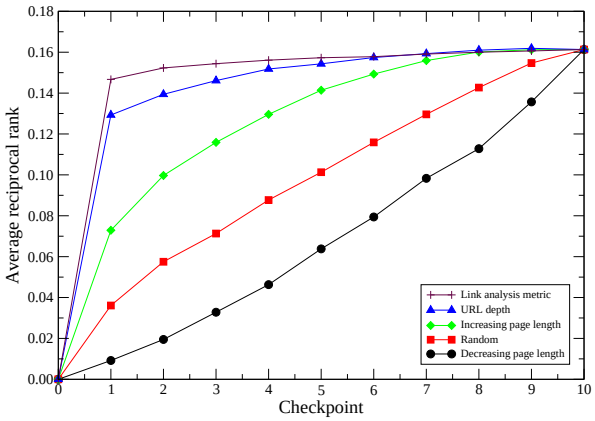

Figure 9: Effect of crawling order on result quality.

Table 3: Average reciprocal rank with different ordering strategies and download rates

	D			
Ordering strategy	18.5	23.5	27.6	30.9
Decreasing page length	0.041	0.058	0.072	0.086
Random	0.085	0.100	0.113	0.123
Increasing page length	0.130	0.143	0.150	0.154
URL depth	0.153	0.156	**0.158**	**0.159**
Link analysis metric	**0.156**	**0.158**	0.157	**0.159**

and also the relevance relationship between documents and user queries while replicating the data. The data should be replicated such that each data center should store the parts of the index that are most relevant to its users. Second, novel query forwarding techniques are needed. This problem is somewhat similar to the collection selection problem in federated IR. However, due to the collaborative nature of search sites, it is possible to devise superior algorithms.

6. REFERENCES

[1] R. Baeza-Yates, C. Castillo, F. Junqueira, V. Plachouras, and F. Silvestri. Challenges in distributed information retrieval. In *Proc. 23rd Int'l Conf. on Data Engineering*, pages 6–20, 2007.

[2] R. Baeza-Yates, C. Castillo, M. Marin, and A. Rodriguez. Crawling a country: better strategies than breadth-first for web page ordering. In *Special Interest Tracks and Posters of the 14th Int'l World Wide Web Conf.*, 2005.

[3] R. Baeza-Yates, A. Gionis, F. Junqueira, V. Plachouras, and L. Telloli. On the feasibility of multi-site web search engines. Technical Report, 2009.

[4] M. Bawa, G. S. Manku, and P. Raghavan. Sets: search enhanced by topic segmentation. In *Proc. 26th Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 306–313, 2003.

[5] J. Callan. Distributed information retrieval. In *Advances in Information Retrieval. Recent Research from the Center for Intelligent Information Retrieval, volume 7 of The Kluwer Int'l Series on Information Retrieval, chapter 5*, pages 127–150, 2000.

[6] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *Proc. 18th Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 339–348, 1995.

[7] B. B. Cambazoglu, F. Junqueira, V. Plachouras, and L. Telloli. On the feasibility of geographically distributed web crawling. In *Proc. 3rd Int'l Conf. on Scalable Information Systems*, 2008.

[8] J. Cho and H. Garcia-Molina. Parallel crawlers. In *Proc. 11th Int'l World Wide Web Conf.*, pages 124–135, 2002.

[9] J. Cho, H. Garcia-Molina, and L. Page. Efficient crawling through url ordering. *Computer Networks and ISDN Systems*, 30(1-7):161–172, 1998.

[10] D. Harman and G. Candela. Retrieving records from a gigabyte of text on a multicomputer using statistical ranking. *Journal of the American Society for Information Science*, 41(8):581–589, 1990.

[11] B. Huffaker, M. Fomenkov, D. J. Plummer, D. Moore, and K. Claffy. Distance metrics in the internet. In *Proc. Int'l Telecommunications Symposium*, 2002.

[12] M. Najork and J. L. Wiener. Breadth-first crawling yields high-quality pages. In *Proc. 10th Int'l World Wide Web Conf.*, pages 114–118, 2001.

[13] D. Puppin, F. Silvestri, and D. Laforenza. Query-driven document partitioning and collection selection. In *Proc. 1st Int'l Conf. on Scalable Information Systems*, 2006.

[14] C. Sarigiannis, V. Plachouras, and R. Baeza-Yates. A study of the impact of index updates on distributed query processing for web search. In *Proc. 31st European Conf. on Information Retrieval*, pages 595–602, 2009.

[15] C. Tang, Z. Xu, and M. Mahalingam. Peersearch: Efficient information retrieval in peer-to-peer networks. In *HotNets-I*, 2002.

[16] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann, 1999.

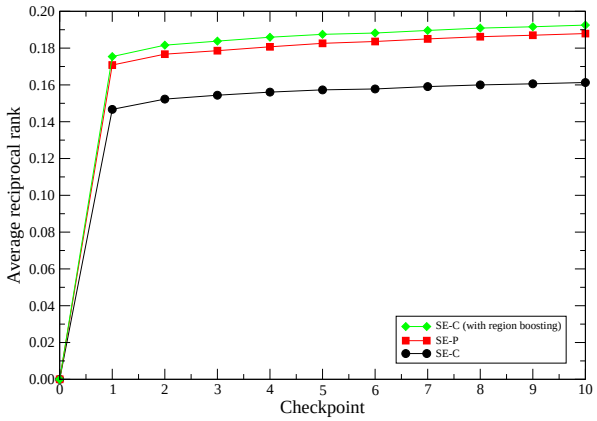

Figure 10: Result quality with region boosting.