An Evaluation and Comparison of Current Peer-to-Peer Full-Text Keyword Search Techniques*

Ming Zhong Justin Moore Kai Shen
Department of Computer Science
University of Rochester
Rochester, NY 14627, USA
{zhong,jmoore,kshen}@cs.rochester.edu

Amy L. Murphy
School of Informatics
University of Lugano
Lugano, CH-6904, Switzerland
amy.murphy@unisi.ch

ABSTRACT

Current peer-to-peer (p2p) full-text keyword search techniques fall into the following categories: document-based partitioning, keyword-based partitioning, hybrid indexing, and semantic search. This paper provides a performance evaluation and comparison of these p2p full-text keyword search techniques on a dataset with 3.7 million web pages and 6.8 million search queries. Our evaluation results can serve as a guide for choosing the most suitable p2p full-text keyword search technique based on given system parameters, such as network size, the number of documents, and the number of queries per second.

1. INTRODUCTION

The capability to locate desired documents using full-text keyword search is essential for large-scale p2p networks. Centralized search engines can be employed in p2p networks and provide look-up service. Although these systems may provide a high level of scalability and availability, a p2p keyword search system may be preferable due to its robustness, low maintenance cost, and data freshness.

A large number of p2p keyword search systems have been proposed, including those using document-based partitioning [10, 11], keyword-based partitioning [9, 12, 16, 21], hybrid indexing [23], and semantic search [8, 13, 24]. However, there is still no comprehensive understanding on the tradeoffs between these four types of techniques under different system environments. In this paper, we provide an evaluation and comparison of existing p2p keyword search techniques on 3.7 million web pages and 6.8 million real web queries. To further project the performance of current p2p keyword search techniques on very large datasets, we linearly scale our evaluation results to 10^9 web pages. Our results suggest that there is no absolute best choice among current p2p keyword search techniques. More importantly, our results can serve as a guide for a user to make her choice based on specific system parameters, such as network size, the number of documents, and the query throughput.

Most current performance evaluation results for p2p keyword search systems [8, 9, 13, 16, 21, 23, 24] are based on datasets with less than 530,000 web pages and 100,000 queries, which are an order of magnitude smaller than our datasets. The only exception we are aware of is Li *et al.*'s work [12], which uses 1.7 million web pages and 81,000 queries to evaluate the feasibility of keyword-based partitioning. However, they did not give specific evaluation results on the communication cost and search latency on their datasets. In addition, there is no previous performance comparison for all four types of existing p2p keyword search techniques on the same dataset.

The remainder of this paper is organized as follows. Section 2 provides an overview of our performance evaluation framework and technical background. Sections 3 to 6 evaluate each of the four types of p2p keyword search techniques and explore directions to improve the search quality or to reduce the overhead of current p2p keyword search systems. Section 7 compares current p2p keyword search techniques by scaling our simulation results to 10^9 web pages and Section 8 concludes this paper.

2. EVALUATION SETUP

In our evaluation framework, a search finds the page IDs of several (*e.g.*, 20) most relevant web pages since most users are only interested in the most relevant web pages. A complete search system may also retrieve the page URLs and digests based on the page IDs found. This step could be efficiently supported by using any distributed hash table (*e.g.*, Chord [20] or CAN [15]) and we do not examine that in our performance evaluation.

Our evaluation dataset contains 3.7 million web pages and 6.8 million web queries. The web pages are crawled based on URL listings of the Open Directory Project [6]. The queries are from a partial query log at the Ask Jeeves search engine [1] over the week of 01/06/2002–01/12/2002 and there are an average of 2.54 terms per query. The web pages are pre-processed by using the stopword list of the SMART software package [19] and removing the HTML tags. In addition, we restrict the vocabulary to be the 253,334 words that appear in our query log. After preprocessing, the average number of distinct words per page is approximately 114. In our implementation, each web page is associated with an 8-byte page ID, which is computed by using its URL as a key to the MD5 hash algorithm. Each page ID in the inverted list of term A is associated with its term frequency of A (the number of occurrences of A in the page), which is stored as a short integer (2 bytes). Thus each entry of an inverted list is 10 bytes.

We evaluate the performance of p2p keyword search techniques in terms of four metrics: *total storage consumption*, *communication cost*, *search latency*, and *search quality*. Here communication cost measures the average number of bytes that needs to be sent over the network in order to return

*This work was supported in part by NSF grants CCR-0306473, ITR/IIS-0312925, and NSF CAREER Award CCF-0448413.

the top 20 most relevant page IDs for a query. Search latency is the time for a p2p keyword search system to return the top 20 most relevant results. Search quality is defined as the overlapping percentage between the search results of a centralized keyword search and those of a p2p keyword search. Ideally, p2p keyword search should return exactly the same result as the centralized keyword search with moderate search latency and communication cost.

In order to estimate search latency based on the communication cost, we make the following assumptions. We assume that the latency for each link in the p2p overlay is 40 ms and the maximum Internet bandwidth consumption of a p2p keyword search system is 1 Gbps, which is approximately 0.26% of the US Internet backbone bandwidth in 2002 (381.90 Gbps as reported by TeleGeography [25]). We assume that the maximum available network bandwidth per query is 1.5 Mbps — the bandwidth of a T1 link.

We use the Vector Space Model (VSM) to rank the relevance of web pages to a query. In VSM [3], the similarity between two pages is measured by the inner product of their corresponding page vectors, which is typically computed by using variants of the TF.IDF term weighting scheme [18]. We are aware that some term weighting schemes, such as Okapi [17], are reported to have better performance than the standard term weighting scheme. However, it is *not* our goal to explore the performance of centralized keyword search systems with different term weighting schemes.

3. DOCUMENT-BASED PARTITIONING

In document-based partitioning, the web pages are divided among the nodes, and each node maintains local inverted lists of the web pages it has been assigned. A query is broadcast to all nodes, and each node returns the k most highly ranked web pages in its local inverted lists.

In our evaluation, the web pages are randomly distributed among the overlay nodes. Assuming the availability of an overlay multicast management protocol [2], the query broadcast and result aggregation are done through a pre-constructed broadcast/aggregation tree with depth $\log n$ for a network with n nodes. Only the top 20 most highly ranked pages from each node are considered. Each node in the aggregation tree merges its local query result with the results from its children and returns the top 20 pages to its parent. Thus the size of the results returned from each node is constant.

According to the VSM page ranking algorithm, the computation of term weights requires some global statistics (*e.g.*, the popularity of terms), which can only be estimated locally based on the partition at each node. Therefore, the query results of document-based partitioning may be different from the results of the centralized search, which is based on accurate global statistics. We evaluate the quality degradation of document-based partitioning using our dataset. Figure 1(A) presents the results on networks with different sizes.

The total storage consumption of document-based partitioning is $d \cdot W \cdot i$, where d is the total number of web pages, W is the average number of distinct terms per page, and i is size of an inverted list entry. For our dataset, $d = 3720390$, $W = 114$, $i = 10$ bytes. Therefore, the total storage requirement of document-based partitioning is $3720390 \times 114 \times 10 \approx 4.24$ GB.

A message of query results (containing 20 page IDs and their relevance scores) has $20 \times 10 = 200$ bytes. Assume each message has an additional overhead of 100 bytes. Thus the total communication cost for a query is $300 \times (n-1)$ bytes, which grows linearly with the network size.

The search latency of document-based partitioning is dominated by the network broadcast and aggregation time under the assumption that the local search at each node and the merging of search results can be done efficiently (otherwise no efficient p2p keyword search could be possible at all). The network broadcast and aggregation time is $2 \times \log n \times 0.04$ seconds since the tree depth is $\log n$.

4. KEYWORD-BASED PARTITIONING

For keyword-based partitioning, each node maintains the complete inverted lists of some keywords. A query with $k \geq 1$ keywords needs to contact k nodes and requires that the inverted lists of $k-1$ keywords be sent over the network.

The baseline keyword-based partitioning randomly distributes the inverted lists of keywords over network nodes and always sends the smallest inverted list over the network when computing the intersection of inverted lists. Hence a k-word query visits k nodes sequentially in the ascending order of the inverted list sizes, which aims to minimize the network communication overhead of the set intersections.

Unlike document-based partitioning, there is *no* quality degradation when using keyword-based partitioning. The total storage consumption of the baseline keyword-based partitioning is identical to that of document-based partitioning, though some optimization techniques (*e.g.* caching, pre-computation) may lead to extra storage consumption.

In the evaluation of keyword-based partitioning, we use a Chord [20] ring to organize nodes into an overlay, where the inverted list of a term x is stored in the overlay by using x as a hash key. Given a query term A, the inverted list of A can be found within $\log n \times 0.04$ seconds in a Chord ring with n nodes since the network diameter is $\log n$ and the average latency per link is 40 ms in our settings.

For a k-keyword query, the search latency T is as follows.

$$T = T_{linkLatency} + T_{transmission} \tag{1}$$

$T_{linkLatency} \leq (k+1) \times \log n \times 0.04$ seconds is the total network link latency for a k-keyword query since a k-keyword query needs to go through $k+1$ node-to-node trips and each trip takes at most $\log n \times 0.04$ seconds in a Chord with n nodes. $T_{transmission}$, the time to send the inverted lists over the network, is $\frac{C}{B}$, where C is the communication cost of the query and B = 1.5 Mbps is the available network bandwidth per query in our evaluation settings. For very large datasets, $T_{transmission}$ becomes the dominant factor of the search latency of keyword-based partitioning. Note that we do not consider the local computation time since it is usually small and negligible compared with $T_{linkLatency}$ and $T_{transmission}$.

Figure 1(B) shows the distribution of the communication cost per query for the baseline keyword-based partitioning, where the average communication cost per query is 96.61 KB and the maximum cost is 18.65 MB. Given that the average number of terms per query is 2.54, the average search latency of the baseline keyword-based partitioning can be computed based on Equation (1):

$$\begin{aligned} T &= ((2.54+1) \times \log n \times 0.04) + \left(\frac{96.61 \times 1000 \times 8}{1.5 \times 10^6}\right) \\ &= (0.14 \times \log n) + (0.52) \text{ sec.} \end{aligned} \tag{2}$$

The communication cost of the baseline keyword-based partitioning can be reduced by the following techniques without compromising the quality: Bloom filters, pre-computation, caching, query log mining, incremental set intersection, and

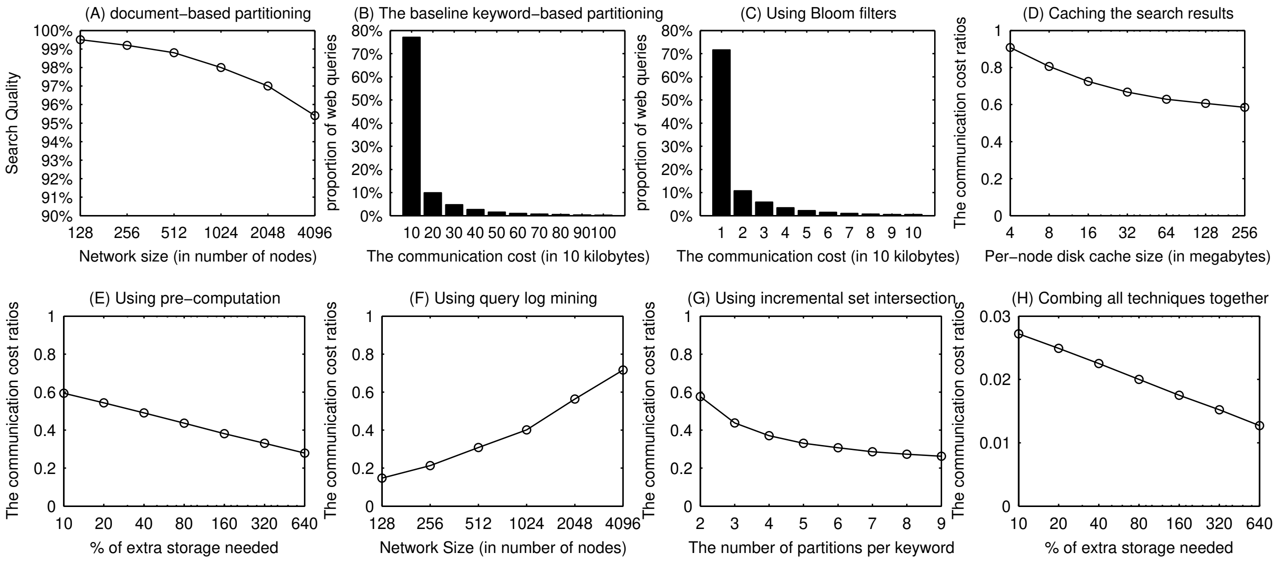

Figure 1: The evaluation results for document-based partitioning and keyword-based partitioning.

other techniques. We discuss these techniques in the remainder of this section. During our discussion, the *communication cost ratio*, c, is defined as the average ratio of reduced communication cost to the communication cost of the baseline keyword-based partitioning. Hence the new network transmission time $T_{transmission}$ is c times that of the baseline keyword-based partitioning and $T_{linkLatency}$ remains almost unchanged. Based on Equation(2), we have the new search latency:

$$T(c) = (0.14 \times \log n) + (0.52 \times c) \text{ sec.} \tag{3}$$

4.1 Bloom Filters

Li *et al.* [12] and Reynold and Vahdat [16] suggest to use *Bloom filters* [4] as a compact representation of sets. By using Bloom filters, the communication overhead for set intersections is significantly reduced at the cost of a small probability of false positives. Given two sets A, B with $|A| < |B|$ and each element in the sets having i bytes. The number of bits, m, in a Bloom filter that minimizes the communication cost for computing $A \bigcap B$ can be determined by the following equation from Reynold and Vahdat [16].

$$m = |A| \cdot \log_{0.6185}\left(\frac{2.081}{i} \cdot \frac{|A|}{|B|}\right) \tag{4}$$

We implemented Bloom filters on our dataset based on Equation (4), where $i = 64$ for our system. Figure 1(C) shows the distribution of per-query communication cost of our implementation of Bloom filters, where the communication ratio is 0.137. Hence the search latency is reduced to $(0.14 \times \log n) + (0.137 \times 0.52)$ seconds according to Equation (3).

4.2 Caching

Previous research [12, 16] has suggested that the communication cost for set intersections can be reduced by *caching* the sets or their Bloom filters received at each node. Our experiments show that it is more helpful for each node to directly cache its search results. LFU policy is used in our cache implementation. Figure 1(D) presents the communication cost ratio of our cache implementation with different cache sizes. The new search latency can be easily calculated based on Equation (3).

4.3 Pre-Computation

Gnawali as well as others [9, 12] suggest to use *pre-computation*, which computes and stores the intersection results of the inverted lists of popular query keywords in advance. Here we pre-compute the intersections of the most frequently used keyword pairs, keyword triplets, and keyword quartets in the query log. Figure 1(E) illustrates how pre-computation saves communication cost at the expense of extra storage consumption.

4.4 Query Log Mining

We propose to use *query log mining*, which explores a better way to distribute inverted lists over the nodes than the uniformly random scheme used by the baseline keyword-based partitioning. Our query log mining clusters keywords into similar-sized groups based on their correlations. By distributing each group of keywords to a node, the intersection of the inverted lists of keywords within the same group does not incur any network communication.

We represent our query log as a weighted graph, where each node represents a query term and the weight of an edge (u, v) is the number of queries that contain both u, v. By using the chaco [5] package recursively, the graph is partitioned into groups with nearly balanced storage consumption such that the words in the same group tend to be highly correlated. A sampled group on a 4096-node network includes the following words: san ca diego francisco puerto tx austin rico antonio earthquake jose juan vallarta lucas rican luis cabo fransisco bernardino.

Figure 1(F) illustrates how our query log mining results help to reduce the communication cost ratios for networks with different number of nodes (keyword groups).

4.5 Other Techniques

Based on the assumption that users are only interested in the most relevant results of a search, *incremental set intersection* reduces the communication cost by only retrieving the top k most relevant web pages. Variants of Fagin's algorithm [7] has been used in some p2p keyword search systems [21] to achieve *incremental set intersection*.

Figure 1(G) presents the communication cost ratios when different values of the number of partitions per keyword, are used for our dataset.

As suggested by Gnawali and Li *et al.* [9, 12], other compression methods, such as compressed Bloom filters [14] and gap compression [26], may also be used to reduce the communication cost. However, these methods only lead to slight improvement in our experiment.

4.6 Combine Them Together

Figure 1(H) presents the communication cost ratios after using Bloom filters, pre-computation, caching (256 MB per node), query log mining, and incremental set intersection (with 3 partitions per keyword) together on 4096-node networks. The ratios in Figure 1(H) are larger than the product of the ratios in Figure 1(C) to 1(G) since the performances of these techniques are not completely orthogonal. Given the communication cost ratios, the search latency can be easily calculated based on Equation (3).

5. HYBRID INDEXING

Hybrid indexing [23] saves the communication cost of keyword-based partitioning by associating each page ID in an inverted list with some *metadata* of the corresponding web page.

A naive approach is to associate each page ID in an inverted list with a complete term list of the corresponding web page. This way the intersection of the inverted lists for multiple keyword search can be done locally with no communication needed. Let L be the average size of the term lists of web pages. Let l be the average size of the entries in the original inverted lists. The above naive approach requires $\frac{L}{l} + 1$ times the storage consumption of the original inverted lists, which may be as high as several hundreds and thus is prohibitive for very large datasets.

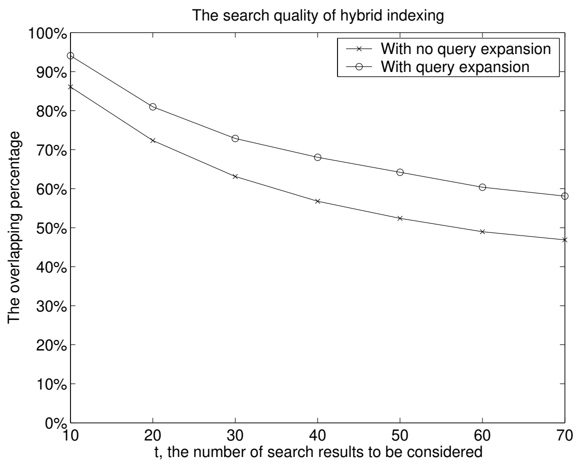

Figure 2: The search quality of hybrid indexing on our dataset. The quality is measured by the overlapping percentage between the top t search results of the hybrid indexing and those of centralized search.

The hybrid indexing approach proposed by Tang *et al.* [23] uses VSM [3] to identify the top k terms (with k highest term weights) in a document and only publishes the term list of the document to the inverted lists of these top terms. Briefly speaking, the inverted list of a term, x, only contains those page IDs that have x as their top terms. This approach may degrade the quality of the search results since if the terms of a query are not among the top terms of a document then the query cannot find this document. In their approach, classical IR algorithms and query expansion are used to improve search quality. Query expansion works by expanding the query with new relevant terms extracted from the best matched pages to the original query. For the details of query expansion, please refer to [23].

The total storage consumption of hybrid indexing is $1 + \frac{k}{W} \cdot \frac{L}{l}$ times that of the standard keyword-based partitioning. Here L is the average size of the term lists of web pages. l is the average size of the entries in an inverted list. k is the number of top terms under which a web page is published. W is the the average number of distinct terms per page. Let $k = 20$ and each entry of a term list consists of a 4-byte term ID and a 2-byte term frequency. Hence the total storage consumption of hybrid indexing on our dataset is $1 + \frac{k}{W} \cdot \frac{L}{l} = 1 + \frac{20}{114} \times \frac{6 \times 114}{10} = 13$ times that of the baseline keyword-based partitioning.

The communication cost of hybrid indexing on our dataset is 7.5 KB per query, which is independent of the dataset size.

For hybrid indexing, distributed hash tables (DHT) are necessary for storing and finding the inverted list of a term x by using x as a hash key. Let D denote the network diameter of the underlying DHT. If query expansion is not used, then a hybrid indexing search contacts the inverted list of a query term (or all inverted lists of query terms in parallel) and retrieve the search results. Hence the search latency of hybrid indexing is $2 \times D \times 0.04$ seconds. If query expansion is used, then a hybrid indexing search consists of two searches: one for the original query and the other for the expanded query. Hence the search latency is $4 \times D \times 0.04$ seconds if query expansion is used.

Figure 2 studies the search quality of hybrid indexing, where each web page is published under its top 20 terms. As suggested by Tang *et al.* [23], the query expansion in Figure 2 is based on the 10 most relevant terms in the top 10 best matched pages to the original query.

6. SEMANTIC SEARCH

Semantic search [8, 13, 24] use classical IR algorithms (*e.g.*, Latent Semantic Indexing) to map web pages and queries to points in a semantic space with reduced dimensionality (typically between 50 and 350). For each query, semantic search returns the top a few *closest* points to the query point in the multi-dimensional semantic space, where the *closeness* between points A, B is typically measured by the dot product of vectors $\vec{A}, \vec{B}$. As a result, semantic search can be characterized as *nearest neighbor search* in multi-dimensional semantic space.

Here we evaluate the performance of pSearch, the semantic search system proposed by Tang *et al.* [24] on our dataset. In our evaluation, we use LSI matrices with dimensionality 200, which are computed by applying the SVDPACK software package [22] to a term-document matrix that consists of 38457 web pages uniformly sampled from our dataset. The LSI matrices fold web pages or queries (253334-dimensional vectors) into 200-dimensional vectors, which form a semantic space. In our evaluation, the storage consumption of a web page in the semantic space is $8 + (200 \times 4) = 808$ bytes since each 200-dimensional vector (200×4 bytes) is associated with its corresponding page ID (8 bytes).

One of the key obstacles to semantic search is the mismatch between the dimensionality of the semantic space (50

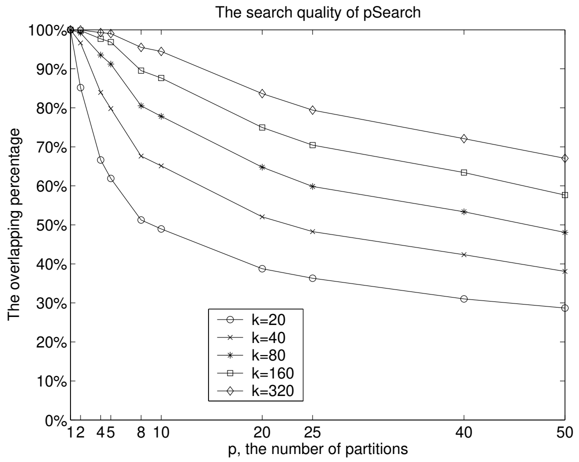

Figure 3: The quality of pSearch on our dataset. The quality is measured by the overlapping percentage between the top 20 search results of pSearch and the top 20 results of centralized semantic search.

to 350) and the effective dimensionality of the p2p overlay hash space, which is at most $2.3 \ln n$ for a CAN network with n nodes [24]. The dimension reduction technique used in pSearch is called *rolling-index*, which partitions a semantic space into p subspaces (each has m dimensions). Hence $p \times m$ is equal to the dimensionality of the semantic space, which is 200 in our evaluation. Another constraint is that m has to be less than or equal to the effective dimensionality of the underlying CAN network. Otherwise a m-dimensional subspace cannot be efficiently implemented on the CAN network hash space. Given that the maximum effective dimensionality of a CAN network with n nodes is $\lfloor 2.3 \ln n \rfloor$ [24], we have $m \leq \lfloor 2.3 \ln n \rfloor$ and $p \geq \lceil \frac{200}{2.3 \ln n} \rceil$.

In rolling-indexing, each point is stored at p places (one for each semantic subspace) in the CAN hash space. For each query, p parallel searches are performed (one for each subspace) and each of them returns the top k points (and their page IDs) found in its corresponding semantic subspace. The query-initiating node merges the results and only returns the top 20 page IDs as the final result.

As we can see from Figure 3, the search quality increases when k (the number of the retrieved points from each semantic subspace) grows. The search quality increases when p (the number of semantic subspaces) gets smaller since a small p means that each subspace has high dimensionality and their closest points to the query are more likely to be among the global closest points. However, p has a lower bound of $\lceil \frac{200}{2.3 \ln n} \rceil$ as we explained before. Specifically, p must be at least 10 for a 6000-node CAN, which leads to 87.64% search quality if 160 points are retrieved from each semantic subspace.

Let i denote the number of bytes needed for storing a point in each semantic subspace. Let d denote the total number of documents in the system. The total storage consumption of pSearch is $d \cdot p \cdot i$. If $p = 10$ (for 6000-node CAN networks), then the total storage requirement of pSearch on our dataset is $3720390 \times 10 \times 808 \approx 30.06$ GB, which is 7.09 times the storage requirement of document-based partitioning or the baseline keyword-based partitioning.

The communication cost of pSearch is $p \cdot k \cdot i$ bytes, which is independent of the dataset size. If we choose $p = 10$ and $k = 160$ (lead to 87.64% search quality), then the communication cost per query is $10 \times 160 \times 808 \approx 1.29$ MB.

According to Equation (1), the search latency of pSearch on a CAN network with n nodes is

$$T = T_{linkLatency} + T_{transmission} = (2 \times n^{\frac{1}{d}} \times 0.04) + \frac{p \cdot k \cdot i \cdot 8}{1.5 \times 10^6} \ sec.$$

where d is $\lfloor 2.3 \ln n \rfloor$, $p = \lceil \frac{200}{2.3 \ln n} \rceil$, and $i = 808$. Note that k, the number of the retrieved points from each semantic subspace, is decided based on the desired search quality since the quality increases as k grows.

7. PERFORMANCE COMPARISON

In order to project the performance of current p2p keyword search techniques on very large datasets, we scale our evaluation results to 10^9 web pages as shown in Table 1.

Table 2 summarizes the advantages and constraints of the four types of p2p full-text keyword search techniques that we considered in this paper.

Document-based partitioning is desirable for a large set of documents since its communication cost is independent of the dataset size and its storage consumption is small compared with other p2p keyword search techniques. However, document-based partitioning requires that the network size and the total number of queries per second must be small. For example, if we assume that each node can handle up to 100 queries per second and the assumptions in Section 2 hold, then document-based partitioning can support up to 4167 nodes and 100 queries per second in total. Generally speaking, the communication cost of document-based partitioning grows linearly with the network size. The number of queries received by each node per second is exactly the number of queries going into the whole system each second since document-based partitioning broadcasts each query to every node. Hence the query throughput of document-based partitioning is bounded by the query throughput of each node.

Keyword-based partitioning (optimized as described in Section 4) is the only known p2p keyword search technique with no quality degradation. Keyword-based partitioning is suitable for large-sized networks since the communication cost of keyword-based partitioning is independent of the network size. However, the communication cost of keyword-based partitioning grows linearly with the number of documents in the system. Hence a user should choose keyword-based partitioning when she prefers no quality degradation or when the total number of documents in the system is not too large. Specifically, if we require that the total network bandwidth consumption is bounded by 1 Gbps and the average search latency is less than 10 seconds, then keyword-based partitioning can support up to 10^8 web pages and 1000 queries per second in total.

Hybrid indexing has small communication cost per query (7.5 KB), which is independent of the total number of documents and network size. This small per-query communication cost is also very helpful when a large number of queries go into the system each second. However, these advantages are achieved at the cost of 10%–50% quality degradation and significant extra storage consumption (13 times under our settings). When quality degradation and extra storage consumption are acceptable, hybrid indexing is a good choice.

Semantic search favors large-sized networks because large-scale networks have high effective dimensionalities and thus lead to small dimension mismatch between the semantic space and the overlay hash space. For instance, for a 6000-node network, semantic search has 87.64% quality with 1.29 MB

Techniques	Total storage	C, comm. cost per query	Latency	Quality
Document-based partitioning	1139.67 GB	$0.3 \times (n-1)$ KB	$2 \times \log n \times 0.04$ seconds	75% to 95% (varies with n)
Keyword-based partitioning (160% extra precomputation storage and 256 MB per-node cache are used)	2963.10 GB	905.82 KB to 1221.78 KB (varies with n)	$(0.14 \times \log n) + 4.82$ seconds to $(0.14 \times \log n) + 6.52$ seconds	100.00%
Hybrid Indexing (with query expansion and the top 20 results in consideration)	14815.70 GB	7.5 KB (independent of n)	$4 \times \log n \times 0.04$ seconds	86.05%
Semantic Search ($n = 6000, k = 160$)	8080.26 GB	1290 KB	7.59 seconds	87.64%

Table 1: The scaled performance of p2p full-text keyword search techniques on a n-node network with 10^9 pages.

Techniques	Advantages	Constraints
Document-based partitioning	1. Suitable for a large number of documents 2. Relatively small storage consumption	1. Requires small network size 2. Requires small number of queries per second in total 3. Has moderate quality degradation
Keyword-based partitioning	1. No quality degradation 2. Suitable for large-sized networks	1. The communication cost grows linearly with the total number of documents 2. Relatively high communication cost 3. Requires moderate extra storage consumption compared with document-based partitioning
Hybrid Indexing	1. Suitable for large-sized networks 2. Suitable for a large number of documents 3. Suitable for a large number of queries per second	1. Has quality degradation 2. Requires significant extra storage consumption than document-based partitioning
Semantic Search	1. Favors large-sized networks 2. Can do concept-based search	1. Requires moderate extra storage than document-based partitioning 2. Has moderate quality degradation 3. Relatively high communication cost 4. Its underlying IR techniques, *e.g.*, LSI, may have scalability problems

Table 2: The advantages and constraints of current p2p full-text keyword search techniques.

communication cost per query and 8080 GB total storage consumption. When the network size grows to 36 million nodes, semantic search can achieve 91.22% quality with 322.50 KB communication cost per query and 4040 GB total storage consumption. In addition, semantic search can find those web pages with similar concepts to the query terms, though they may not have exactly the same term. For example, semantic search can retrieve documents containing the term "automobiles" for queries containing the term "cars". In summary, semantic search is suitable for large-scale networks or concept-based queries.

8. CONCLUSION

This paper provides a performance evaluation and comparison of current p2p full-text keyword search techniques on a dataset of 3.7 million web pages and 6.8 million queries. Our dataset is an order of magnitude larger than the datasets employed in most previous studies (up to 528,543 web pages and 100,000 queries). To further project the performance of current p2p keyword search techniques on very large datasets, we linearly scale our evaluation results to 10^9 web pages. Our evaluation results can serve as a guide for a user to choose p2p keyword search techniques based on specific system parameters, such as network size, the number of documents, and the query throughput.

9. REFERENCES

[1] Ask Jeeves Search. http://www.ask.com.
[2] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable Application Layer Multicast. In *Proc. of ACM SIGCOMM*, pages 205–217, 2002.
[3] M. Berry, Z. Drmac, and E. R. Jessup. Matrices, Vector Spaces, and Information Retrieval. *SIAM Review*, 41(2):335–362, 1999.
[4] B. H. Bloom. Space/time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM*, 13(7):422–426, 1970.
[5] http://www.cs.sandia.gov/ bahendr/chaco.html.
[6] The Open Directory Project. http://www.dmoz.com.
[7] R. Fagin, A. Lotem, and M. Naor. Optimal Aggregation Algorithms for Middleware. In *Proc. of ACM Symp. on Principles of Database Systems*, 2001.
[8] P. Ganesan, B. Yang, and H. Garcia-Molina. One Torus to Rule Them All: Multidimensional Queries in P2P Systems. In *Proc. of WebDB'04*, 2004.
[9] O. D. Gnawali. A Keyword-Set Search System for Peer-to-Peer Networks. Master's thesis, Dept. of Computer Science, Massachusetts Institute of Technology, June 2002.
[10] Gnutella. http://www.gnutella.com.
[11] KaZaA. http://kazaa.com.
[12] J. Li, T. Loo, J. Hellerstein, F. Kaashoek, D. Karger, and R. Morris. On the Feasibility of Peer-to-Peer Web Indexing and Search. In *Proc. of IPTPS'03*, 2003.
[13] M. Li, W. Lee, and A. Sivasubramaniam. Semantic Small World: An Overlay Network for Peer-to-Peer Search. In *Proc. of IEEE ICNP*, 2004.
[14] M. Mitzenmacher. Compressed Bloom Filters. In *Proc. of 20th ACM PODC*, 2001.
[15] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-Aware Overlay Construction and Server Selection. In *Proc. of IEEE INFOCOM*, New York, NY, June 2002.
[16] P. Reynold and A. Vahdat. Efficient Peer-to-Peer Keyword Searching. In *Proc. of Middleware'03*, 2003.
[17] S. E. Robertson, S. Walker, S. Jones, M. HancockBeaulieu, and M. Gatford. Okapi at TREC-3. In *TREC-3*, 1994.
[18] G. Salton and C. Buckley. Term-weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, 24(5):513–523, 1988.
[19] SMART. ftp://ftp.cs.cornell.edu/pub/smart.
[20] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proc. of ACM SIGCOMM*, pages 149–160, San Diego, CA, Aug. 2001.
[21] T. Suel, C. Mathur, J. Wu, J. Zhang, A. Delis, M. Kharrazi, X. Long, and K. Shanmugasunderam. ODISSEA: A Peer-to-Peer Architecture for Scalable Web Search And Information Retrieval. In *Proc. of WebDB'03*, 2003.
[22] SVDPACK. http://www.netlib.org/svdpack/index.html.
[23] C. Tang, S. Dwarkadas, and Z. Xu. Hybrid Global-Local Indexing for Efficient Peer-to-Peer Information Retrieval. In *Proc. of the First USENIX/ACM NSDI*, San Franscisco, CA, Mar. 2004.
[24] C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-Peer Information Retrieval Using Self-Organizing Semantic Overlay Networks. In *Proc. of ACM SIGCOMM*, 2003.
[25] Global Internet Geography 2003. TeleGeography, Inc.
[26] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. 1999.