Supporting Ranked Search in Parallel Search Cluster Networks

Fang Xiong Qiong Luo Dyce Jing Zhao
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
{xfang, luo, zhaojing}@cs.ust.hk

Abstract

Recent work by Cooper et al. proposed the Parallel Search Cluster Network (PSCN) as an efficient P2P network overlay. Organized in clusters, a PSCN allows peers within one cluster to share query workload, and peers across clusters to share indexes. In this paper, we study the problem of supporting ranked keyword search in a PSCN. Because ranking mechanisms, such as TF×IDF, require global information, we investigate how to acquire and distribute the global information in a PSCN. It turns out that this process can be done efficiently by taking advantage of the architectural features of the PSCN. We compare ranked search in a PSCN with that in an unstructured network as well as in a super-peer network, and the results show that our approach is feasible and efficient.

1 Introduction

Recently, Peer-to-Peer (P2P) search networks have been a popular way for data sharing among a large number of peers, in which each peer provides its data to other peers as a server as well as works as a client to request data from other peers. There are two extremes of existing P2P networks, one is flooding-based (e.g., Gnutella [1] and Kazaa [2]), in which queries are broadcasted in the networks, and the other routing-based, in which queries are forwarded based on some addressing mechanisms such as Distributed Hash Table (DHT) [12, 15] or semantics [16]. Some P2P networks lie between the two extremes, for example, the super-peer network [19], in which queries transmitted from normal peers to super-peers are directed, while among super-peers queries are broadcasted. Although routing-based networks have been shown efficient in resource location, flooding-based networks, due to the simplicity of implementation, maintain their wide usage in practice and call for further research efforts on their improvement. In this paper, we choose a new type of flooding-based network, called a Parallel Search Cluster Network [7], or PSCN in short, and study a basic problem in supporting ranked search.

The PSCN is identified through the SIL (Search/Index Link) model [8]. In this model, a connection between peers is categorized into either a search link (transmitting queries and results) or an index link (transmitting indexes and index updates). These links are either forwarding (allowing the recipient to forward the received content) or non-forwarding. Therefore, there are four types of links in a P2P network: NSL (Non-forwarding Search Link), FSL (Forwarding Search Link), NIL (Non-forwarding Index Link), and FIL (Forwarding Index Link).

Based on SIL, a pure search network such as Guntella is one with only FSLs (Figure 1(a)), and a pure index network such as PlanetP [10] is one with only FILs (Figure 1(b)). Both of them are unstructured P2P networks, even though they forward queries and replicate indexes in different ways. A super-peer network is one with FSLs and NILs from normal peers to super-peers as well as NILs between super-peers (Figure 1(c)). Finally, a PSCN is a network consisting of clusters of peers, with FSLs between peers in a cluster and one NIL from each peer to one randomly selected peer in each other cluster. Figure 1(d) shows an example PSCN, with the outgoing links from Clusters 2 and 3 omitted for clarity. It is shown that PSCN is efficient in load balancing, resource utilization, and fault tolerance [8].

As an architectural model for P2P networks, SIL discusses search and indexes in general terms. In the meantime, with the increasing scale and sophistication of P2P networks, keyword search techniques have been developed for data object ID search [12, 15] as well as for content search [10, 14, 16] in various P2P network overlays. Since PSCN is a newly identified P2P architecture, we study how to support content search in it, specifically, how to support the ranking mechanism efficiently by taking advantage of its architectural characteristics.

One widely adopted ranking mechanism for keyword search is TF$\times$IDF [13]. It evaluates the importance of a keyword by considering both the frequency of the keyword occurring in a specific document and that in the document collection. As a result, the ranking process requires aggregate information such as the total number of documents that contain a certain keyword. In a dynamic, decentralized P2P network environment, special care needs to be taken to meet this requirement. For instance, PlanetP [10] approximates TF$\times$IDF at the peer level in order to save storage and communication cost in an unstructured P2P network. As another example, Shen et al. build a hierarchical summary structure that indexes at document, peer, and super-peer levels for a super-peer network.

Based on these previous results on unstructured networks and super-peer networks, we examine how the ranking mechanism can be supported in a PSCN. The process of ranked search in a PSCN is as follows:

1. Every peer builds the local index on its own documents, and transmits its local index across clusters through NILs so that each cluster has the indexes of all peers in the network.

2. At the query time, the querying peer forwards the query through FSLs to other peers of the cluster that it resides in. After the local aggregate

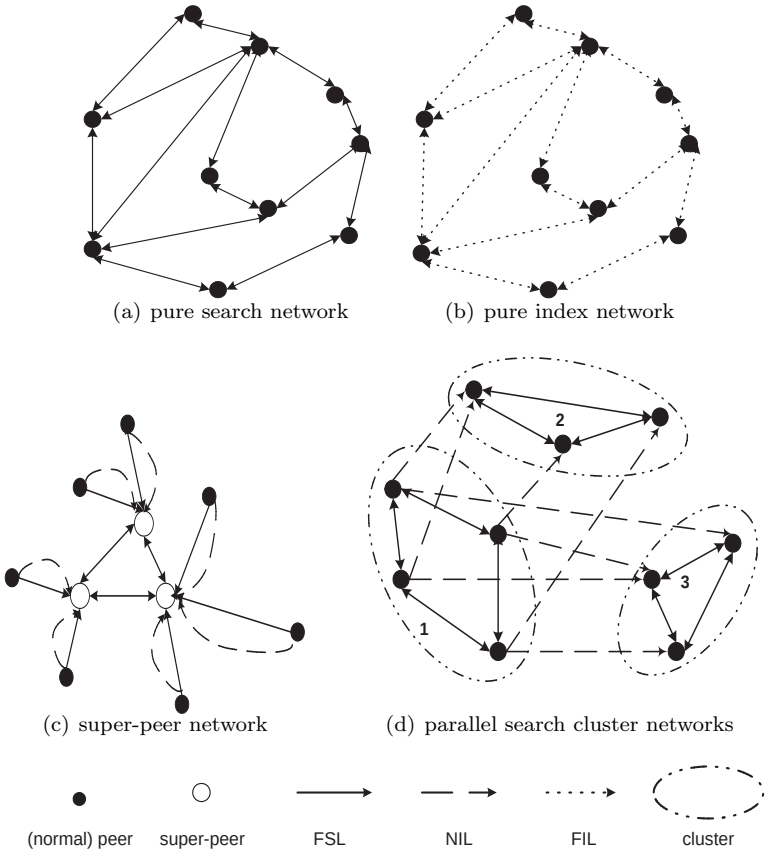

Figure 1: Network overlays under SIL model representation

information related to the query are returned from the involved peers in the cluster, the querying peer merges them into the global aggregate information that are related to the query, and sends the information to the involved peers.

3. Next, each involved peer evaluates the query over the indexes it stores, ranks its query results locally, and sends the locally ranked query results back to the querying peer. Finally, the querying peer merges the locally ranked query results into globally ranked ones and returns all or top-K of them to the user.

Similar to the previous work, the indexes transmitted across clusters in a PSCN can be at the peer level, instead of at the document level. At the peer level, the global ranking is done for peers to indicate which peers are most likely to possess matching documents. To obtain these documents, the querying peer needs to forward the query to the top-k_p peers in the global peer rank, each of which in turn ranks its documents locally. Finally, the querying peer merges the query results and selects the global top-K documents.

Different from the previous work, ranking in a PSCN has the following characteristics: (a) the most time-consuming tasks in the query processing, i.e., local aggregate information collection and local ranking calculation, are distributed over all peers in one cluster; this distribution spreads the query processing load across peers evenly, thus reducing the maximum requirement for the capabilities of individual peers and improving the scalability of the network; (b) there is no global index built for every existing keyword in the network; rather, the collection of aggregate information and the calculation of ranking are performed with respect to an incoming query, which reduces the load on peers; (c) each query is answered within the cluster where the query is submitted, without affecting the completeness of the query result, which improves the query response time and saves communication cost.

The remainder of this paper is organized as follows. We discuss related work in Section 2 and present our design and implementation of ranked search in a PSCN in Section 3. We show our experimental results in Section 4 and conclude in Section 5.

2 Related work

Exact-match keyword search has been supported in existing flooding-based networks such as Napster [3] and Gnutella [1], where keyword search on short file descriptions is sufficient for identifying the desired multimedia files. In comparison, DHT-based schemes, such as CAN [12] and Chord [15], place data objects by data object IDs and consequently support search on the IDs.

Most recently, ranked content search has been supported in unstructured networks [10], super-peer networks [14], and semantic overlays [16], where textual documents are shared in the network. These papers adapt information retrieval techniques including the Vector Space Model (VSM), inverted indexes,

and Latent Semantic Indexing (LSI) to support ranked search in P2P networks, and all of them use the TF×IDF term weight calculation or its variation as the basic building block in ranking.

Even though the ranking formulas are similar among different network overlays, their implementations differ significantly due to the differences in the architectural features. On one extreme, networks with central peers or super-peers maintain indexes and handle ranking at these powerful peers only [14]. On the other extreme, unstructured pure index networks, such as PlanetP, replicate indexes everywhere in the network and therefore are able to handle global ranking at each peer [9, 10]. In contrast, in a PSCN index replicas of all peers are distributed within each cluster, and the query processing load is also shared within the cluster. As a result, the storage overhead and index update cost at each PSCN peer is reduced in comparison with the unstructured pure index network, and the heavy workload on super-peers is also avoided in a PSCN.

Finally, unstructured P2P networks, super-peer networks and PSCNs can be uniformly represented in the Search/ Index Link (SIL) model [8]. We study the ranked search in a PSCN and compare it with those in other network overlays using the model.

3 Ranked keyword search in a PSCN

In this section, we first briefly review the Vector Space Model (VSM) and the TF×IDF ranking mechanism, and then present in detail our implementation of ranked keyword search in a PSCN.

3.1 VSM and TF×IDF

VSM (Vector Space Model) represents a document as a vector of weights, which are assigned to the keywords in the document. Similarly, a keyword query is also represented as a vector of weights of query keywords. The rank of a document with respect to a query, or the relevance between the document and the query, is computed as the similarity between the document vector and the query vector, e.g., the dot product of the two vectors.

A widely used method to assign weights to keywords is TF×IDF, where TF (Term Frequency) is the frequency that a keyword occurs in a document, and IDF (Inverse Document Frequency) is the inverse of the frequency that a keyword occurs in the document collection. The intuition of this mechanism is the following: if a keyword occurs frequently in a document, it is likely to be a keyword that can describe the document content well, and thus is an important keyword; if a keyword occurs frequently in many documents, it is less useful to differentiate a document from others.

There are several variations of TF×IDF implementation [13]. In this paper, we adopt the equations suggested by Witten et al. [17]:

$$w_{D,t} = 1 + log(f_{D,t}) \quad w_{Q,t} = log(1 + N/f_t) \tag{1}$$

where $w_{D,t}$ is the weight of term t in document D, $f_{D,t}$ the number of times that term t occurs in document D, $w_{Q,t}$ the weight of term t in query Q, N the total number of documents in the collection, and f_t the number of documents in which term t occurs.

Subsequently, the similarity between document D and query Q is calculated as follows:

$$Sim(Q, D) = \frac{\sum_{t \in Q} w_{D,t} \times w_{Q,t}}{|D|} \tag{2}$$

where $|D| = \sum_{t \in D} f_{D,t}$.

The statistics used in TF×IDF calculation can be obtained from an inverted index. Given a document collection, for each term t an inverted index records which documents contain the term, and the term frequency ($f_{D,t}$) in these documents. In addition, some aggregate information is included in the index, such as the total number of documents, N, in the collection and the number of documents in which a term occurs, f_t. In the remainder of this paper, we assume the existence of the inverted index at each peer.

3.2 Local index in a PSCN

In a PSCN, each peer maintains its local index, i.e., the index about its local documents. This index can be built at the document level or at the peer level.

Denote the total number of peers currently in the network as N_p, and the number of documents at the peer j ($1 \leq j \leq N_p$) as n_j. The document vector vd_{jr} of the rth document located at peer j, d_{jr}, is:

$$\begin{aligned} &vd_{jr} = (keyword_{jr,1}, keyword_{jr,2}, ..., keyword_{jr,l_{jr}}) \\ &keyword_{jrv} = (word_{jrv}, count_{jrv}) \quad (1 \leq v \leq l_{jr}) \end{aligned}$$

where l_{jr} denotes the number of unique terms in document d_{jr}, $word_{jrv}(1 \leq v \leq l_{jr})$ the vth term in document d_{jr}, and $count_{jrv}(1 \leq v \leq l_{jr})$ the number of times that term $word_{jrv}$ occurs in document d_{jr}. This document vector is calculated after stop word removal and stemming.

After document vectors are calculated for all documents, the local index at a peer is built by merging these vectors at one of the two levels - the document level or the peer level. The local index at the document level uses the term frequency with respect to each document whereas that at the peer level records the term frequency with respect to each peer. Apparently, the peer-level local index is at a coarser granularity than the document-level one.

The local index at peer j at the document level takes the following form:

$$\begin{aligned} &doclist_j = (d_{j1}, d_{j2}, ..., d_{j,n_j}) \\ &numlist_j = (D_{j1}, D_{j2}, ..., D_{j,n_j}) \\ &termlist_j = (term_{j1}, term_{j2}, ..., term_{j,m_j}) \\ &term_{jk} = (t_{jk}, tflist_{jk}) \quad (1 \leq k \leq m_j) \\ &tflist_{jk} = (tf_{jk1}, tf_{jk2}, ..., tf_{jk,g_{jk}}) \\ &tf_{jkw} = (loc_{jkw}, f_{jkw}) \quad (1 \leq w \leq g_{jk}) \end{aligned}$$

where $doclist_j$ is the list of n_j documents located at peer j, $D_{jr}(1 \leq r \leq n_j)$ the number of terms contained in document d_{jr}, m_j the number of unique terms at peer j, t_{jk} the kth term at peer j, g_{jk} the number of documents at peer j in which term t_{jk} occurs, $loc_{jkw}(1 \leq w \leq g_{jk})$ the wth document at peer j in which term t_{jk} occurs, and $f_{jkw}(1 \leq w \leq g_{jk})$ the term frequency of t_{jk} in document loc_{jkw}: $f_{jkw} = \{count_{jrv} | loc_{jkw} = d_{jr} \& word_{jrv} = t_{jk}\}$.

In comparison, the local index at the peer level takes a simpler form:

$$termlist_j = (term_{j1}, term_{j2}, ..., term_{j,m_j})$$
$$term_{jk} = (t_{jk}, f_{jk}) \qquad (1 \leq k \leq m_j)$$

where m_j is the number of unique terms occurring at peer j, t_{jk} the kth term at peer j, and $f_{jk} = \sum_{1 \leq w \leq g_{jk}} f_{jkw}$ the number of times that term t_{jk} occurs at peer j. In addition, the local index at the peer level records the total number of terms occurring at peer j: $D_j = \sum_{1 \leq r \leq n_j} D_{jr}$.

3.3 Search in a PSCN

In a PSCN, besides its own local index each peer stores a copy of the local indexes of some peers from other clusters, so that the global aggregate information for the entire network can be computed within one cluster, and therefore a query can be answered within one cluster.

3.3.1 Search using the document-level index

In this subsection, we assume that the indexes transmitted across clusters are at the document level.

Suppose a query $Q = (keyword_1, keyword_2, ..., keyword_q)$ consisting of q unique terms is submitted to a peer, which we call the *querying peer*. The querying peer forwards the query to other peers in the cluster and the collection of the aggregate information is activated.

On receiving the query, a peer p calculates its local aggregate information as follows:

$$doc_p = \sum_{j \in P} n_j$$
$$doc\_freq_{up} = \sum_{j \in P \& t_{jk} = keyword_u} g_{jk} \quad (1 \leq u \leq q)$$

where doc_p is the total number of documents whose local indexes are stored in peer p's index repository P, and doc_freq_{up} the number of documents containing term $keyword_u$ whose local indexes are stored at peer p.

After peer p returns its local aggregate information to the querying peer, the querying peer calculates the global aggregate information with respect to query Q as follows:

$$N = \sum_{p \in C} doc_p = \sum_{1 \leq j \leq N_p} n_j$$
$$DF_u = \sum_{p \in C} doc\_freq_{up} \qquad (1 \leq u \leq q)$$

Recall that N is the total number of documents, N_p the total number of peers in the network, and n_j the number of documents at peer j. C denotes the current cluster, while DF_u represents the total number of documents in the network that contain term $keyword_u$. Note the global aggregate information is calculated as the sum of the local aggregate information returned by every peer p in the current cluster. Furthermore, the aggregate information is collected with respect to the q keywords in the query, instead of all keywords existing in the network.

After the global aggregate information is computed at the querying peer, it is forwarded to other peers in the cluster for document ranking. The similarity between document d_{jr} and query Q is adapted from Equations 1 and 2:

$$Sim_{jr,Q} = \frac{\sum_{1 \le u \le q} [1 + log(freq_{jru})] \times log(1 + N/DF_u)}{D_{jr}}$$

$$freq_{jru} = \begin{cases} count_{jrv} & \exists k, v, keyword_u = t_{jk} = word_{jrv} \\ e^{-1} & \text{otherwise} \end{cases}$$

The ranking is performed locally at each peer within the cluster in parallel.

Finally, the local ranking results of the peers are returned to the querying peer and are merged into globally ranked results with respect to query Q. When the number of relevant documents in the network is much more than user requirement, we will return only the top-K documents.

3.3.2 Search using the peer-level index

The process of searching using indexes at the peer level is similar to that using indexes at the document level, except the computation of the aggregate information and rank.

Given a query $Q = (keyword_1, keyword_2, ..., keyword_q)$, a peer p calculates its local aggregate information as follows:

$$peer_p = \sum_{j \in P} 1$$

$$peer\_freq_{up} = \sum_{j \in P \& t_{jk} = keyword_u} 1 \quad (1 \le u \le q)$$

where $peer_p$ is the total number of peers whose local indexes are stored at peer p's index repository P, and $peer\_freq_{up}$ the number of peers whose local indexes are stored at peer p and whose documents contain term $keyword_u$.

The global aggregate information with respect to query Q is as follows:

$$N_p = \sum_{p \in C} peer_p$$

$$PF_u = \sum_{p \in C} peer\_freq_{up} \quad (1 \le u \le q)$$

where N_p is the total number of peers in the network and PF_u the number of peers in the network that contain term $keyword_u$. The global aggregate

information is the sum of the local aggregate information from every peer p in the current cluster C, the same as the case of using indexes at the document level.

The peer ranking is calculated by the similarity between peer j and query Q, adapted from Equations 1 and 2:

$$Sim_{j,Q} = \frac{\sum_{1 \le u \le q} [1 + log(freq_{ju})] \times log(1 + N_p/PF_u)}{D_j}$$

$$freq_{ju} = \begin{cases} f_{jk} & \exists k,\ keyword_u = t_{jk} \\ e^{-1} & \text{otherwise} \end{cases}$$

The global ranking result obtained at the querying peer is the ranking of peers, indicating the relevance between a peer and a query. To rank the relevant documents, the querying peer contacts the top-k_p peers directly, which will rank their own documents based on their local indexes at the document level. The calculation of similarity between document d_{jr} and query Q is adapted from Equation 1 and 2:

$$Sim_{jr,Q} = \frac{\sum_{1 \le u \le q} [1 + log(freq_{jru})] \times log(1 + n_j/G_u)}{D_{jr}}$$

$$freq_{jru} = \begin{cases} count_{jrv} & \exists k, v, keyword_u = t_{jk} = word_{jrv} \\ e^{-1} & \text{otherwise} \end{cases}$$

$$G_u = \begin{cases} g_{jk} & \exists k, keyword_u = t_{jk} \\ \infty & \text{otherwise} \end{cases}$$

The locally ranked top-K documents returned by each of the top-k_p peers will be sorted at the querying peer by the similarity, and the globally ranked top-K documents will be presented to the user.

3.3.3 Discussion

In a P2P environment, the local index at a peer can be replicated over the network in various ways for different overlays, and ranked search is performed based on the replicated indexes. We have presented the process of ranked search in a PSCN, but the same process can be applied in other overlays with slight modification. In our experiments, we modified the algorithm for a super-peer network and an unstructured P2P network to make comparisons with the PSCN.

In a super-peer network, indexes of normal peers are replicated at super-peers. As a result, the index information stored at all super-peers is sufficient for generating the global aggregate information. Thus, in order to perform ranked search in a super-peer network, all super-peers together are equivalent to one cluster in a PSCN. In an unstructured network, with the full replication of indexes at each peer, both the global aggregate information and the global rank can be calculated at the querying peer locally.

Advanced information retrieval methods, e.g., LSI, can be applied on top of TF×IDF, to improve the quality of query results and the efficiency of query

Table 1: Network overlay parameter setting

Parameter	Description	Default value
N_p	#peers	100
P_a	Average out-degrees per peer	5
P_m	Maximum out-degrees per peer	10
SN	#super-peers	5
CN	#clusters	15

processing. Nevertheless, a simple but basic tenique such as TF×IDF is sufficient for the purpose of comparing ranked search in different overlays.

4 Experiments

In this section, we report simulation results on ranked search in a PSCN in comparison with those in a super-peer network and in an unstructured pure index network (shortened as unstructured in the following). The performance metrics include precision, processing time, bandwidth usage, storage cost, and update handling cost.

4.1 Setup

All of our experiments were conducted on a 4-CPU 3.2GHz Pentium machine with 1GB of memory running Microsoft Windows XP Professional. We simulated three network overlays (PSCN, super-peer, and unstructured) and distributed documents and indexes to the simulated networks.

We adopted the parameter setting of the network overlays (Table 1) from the work by Cooper et al. [7]. This setting has been shown suitable for search-dominant (as opposed to update-dominant) networks, e.g., the PSCN. Peers in an unstructured network, super-peers in a super-peer network, and peers within one cluster in a PSCN, are strongly connected, respectively. These connections are generated using the PLOD algorithm [11]. In a super-peer network, every normal peer is linked to a super-peer. The number of normal peers linked to a super-peer follows a normal distribution with the mean of N_p/SN and the variance of $N_p/(4 * SN)$, generated by the Box-Muller transformation [5]. In a PSCN, each peer is assigned to a cluster. The number of peers assigned to a cluster also follows a normal distribution, with the mean of N_p/CN and the variance of $N_p/(4 * CN)$.

We experimented with five document collections (Table 2), of which the first four (CACM, CISI, CRAN, and MED) are from SMART [6] and the last one (FT) from TREC [4]. As the SMART collections are much smaller than FT, we mainly experimented with SMART for time efficiency unless we saw a need for FT. Over the SMART collections, the average length of a keyword is $L = 6$ characters, the average query size (the number of keywords in a query) is $q = 13$, and the average number of unique terms contained in a document is $w = 42$.

Table 2: Document collections

Collection	#queries	#documents	#terms	Size (MB)
CACM	64	3, 204	122, 533	2.10
CISI	112	1, 460	94, 824	2.32
CRAN	225	1, 400	137, 661	1.57
MED	30	1, 033	84, 897	1.04
FT	50	210, 158	42, 206, 171	564

We distributed documents over peers following either the uniform distribution (default) or Weibull distribution. Under the uniform distribution of SMART collections, there are $DN = 71$ documents and $W = 1923$ unique terms per peer, with one term occurring in $DL = 2$ documents on average. For the Weibull distribution, we adopted the parameter setting of $\alpha = 0.7$ and $\beta = 46$ from PlanetP [10], as this setting simulates a typical document sharing environment where the majority of all documents are provided by 9% of the peers.

After the local indexes were built, we replicated the indexes in each network. In an unstructured network, peers are connected with FILs, and the local indexes of all peers are replicated everywhere. In a super-peer network, the local index of each normal peer is replicated at a super-peer through the NIL. In a PSCN, every peer replicates its local index to each of the other clusters through the NILs. Finally, for indexing at the peer level, only peer-level indexes are replicated, thus each peer has both its document-level index and its peer-level index as well as the peer-level indexes of other peers.

4.2 Precision

As the total number of returned results per query is usually large, e.g., the average number of results for the CACM queries is 1262, which is far beyond user requirement, we decide to limit the number of returned results and measure the quality of top-K query results by precision.

Since the index information of all documents in the network is accessible when performing ranked search in each of the three overlays, the precision in each overlay is the same as that performed in a central environment.

We examine the effect of k, the ratio of the total number of results returned to the user to the total number of results, on the MED collection (Figure 2), using either the document-level index or the peer-level index. Since k_p, the number of peers participating in the local document ranking, has no significant effect on precision, we set it to be 10 for all experiments. Not surprisingly, the precision decreases when more results are returned. As shown in the figure, when $k = 0.05$, the precision using the document-level index remains around 0.5 whereas that using the peer-level index is about 0.4. With this k value, the average number of results is 376, and the average number of results returned to the user is 19. Therefore, we set the number of results returned to the user,

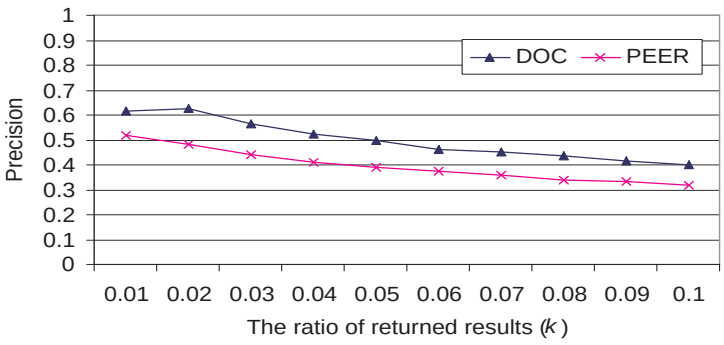

Figure 2: Average precision over MED collection. DOC represents using indexes at the document level, and PEER using indexes at the peer level.

$K = 20$, as the default value.

4.3 Processing time

Recall ranked keyword search in a PSCN is performed in several steps. Using indexes at the document level, it goes through local aggregate information collection (LAI), global aggregate information merging (GAI), local ranking calculation (LR) and global ranking merging (GR). Using indexes at the peer level, it goes through local aggregate information collection (LAI), global aggregate information merging (GAI), local peer ranking calculation (LPR), global peer ranking merging (GPR), local document ranking calculation (LDR) and global document ranking merging (GDR). In Figures 3 and 4, we show the processing time spent on each step for the large document collection FT. This collection is chosen in order to show the difference more clearly.

In Figures 3 and 4, GAI, GR, GPR, and GDR are omitted, because for each of them, the processing time is lower than 10 milliseconds, the smallest time unit reportable on Windows XP, and is consequently reported as 0. In Figure 3, LR is more time-consuming than LAI, because in addition to the common task of searching in indexes, LR needs to calculate the similarity for each document. In Figure 4, the time of LAI is almost the same as that of LPR, in that the number of peers for similarity calculation is only a few. LDR consumes much less time because at the top-k_p peers, only their local documents are considered for ranking.

Common in both figures, the majority of the processing time is spent on local processing (LAI plus LR when using indexes at the document level, and LAI plus LPR when using indexes at the peer level). This suggests it is beneficial to distribute the query processing load evenly in a cluster; otherwise, the bottleneck will be at the super-peers in a super-peer network or at the querying peer in an unstructured network. Also, the processing time remains stable for different K

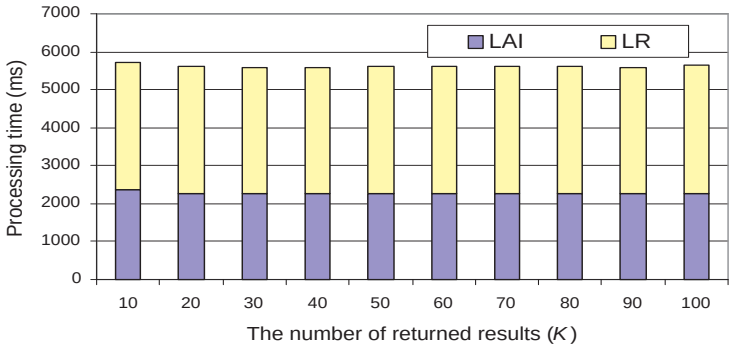

Figure 3: Average processing time spent on each step of ranked search in a PSCN, using indexes at the document level over the FT collection.

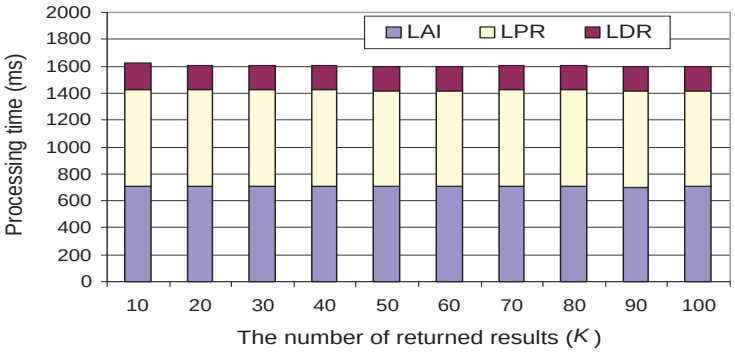

Figure 4: Average processing time spent on each step of ranked search in a PSCN, using indexes at at the peer level over FT collection.

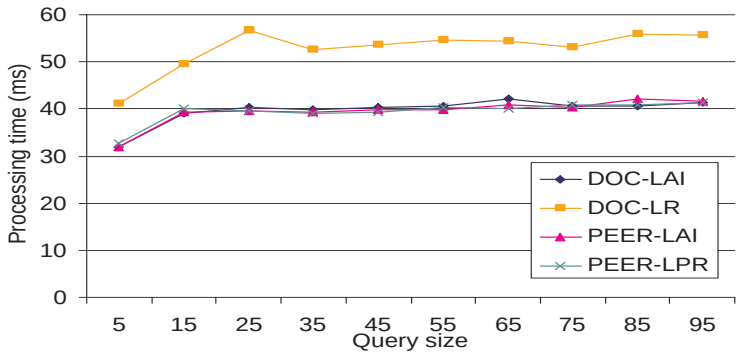

Figure 5: Average processing time with different query sizes, over CACM, CRAN, CISI and MED collections.

values, because even though the number of returned results is not large, each document in the network has to be taken into consideration for ranking.

Comparing Figure 3 and 4, we see that the processing time is much less if the search uses indexes at the peer level. This is because the size of the peer-level index is much smaller, resulting in less time used for searching in indexes.

The query size, or the number of keywords in a query, has an effect on the processing time. As shown in Figure 5, we recorded the processing time spent on LAI and LR when using the document-level index, as well as that spent on LAI and LPR when using the peer-level index, with different query sizes. The common trend is that the processing time grows slowly with the increase in the query size, since it takes more time to look up more keywords in indexes.

Under the Weibull distribution, the number of documents located at an individual peer varies dramatically, leading to the diversity of the size of a single local index and the diversity of the processing time consumed by different peers. As you can see in Figure 6, for both LAI and LR, the time goes up steadily with the number of documents processed by a peer. Thus, it makes sense to distribute the query processing load evenly, in order to reduce the query response time.

Finally, we compare the processing time in different network overlays. We show the processing time when using the document-level index in Figure 7, while the one using the peer-level index in Figure 8. Through the comparison between the two figures, we see that the trends in both cases are similar, but the processing time is slightly more when using the document-level index. Clearly, the processing time in the unstructured network is much more than that in the other two, because all work is done by the querying peer. The processing time in the super-peer network is about 30% more than that in the PSCN, because the workload is distributed among the super-peers as opposed to within one cluster in the PSCN.

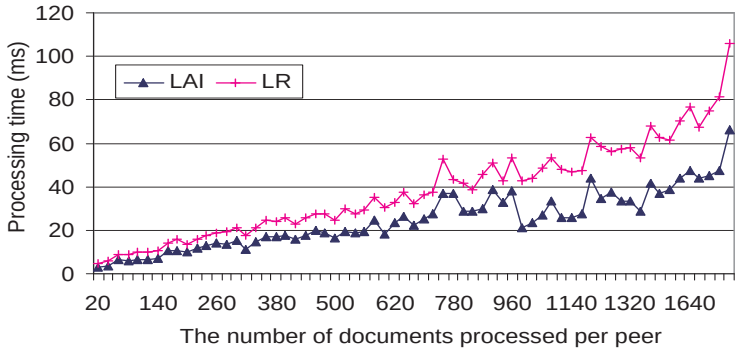

Figure 6: Average processing time for LAI and LR with different numbers of documents, using the document-level index over CACM, CRAN, CISI and MED collections.

4.4 Network bandwidth

In addition to the local processing time, we study the network bandwidth usage for ranked search in different overlays. It includes both the network bandwidth used for query processing and that for index transmission.

There are four types of messages when performing ranked search: queries, the local aggregate information, the global aggregate information and local ranking results. Table 3 lists the bandwidth consumption formulas for these messages. In the following, we first describe the composition of these formulas, and then comment on the estimated bandwidth consumption values.

In a PSCN, each message is transmitted for $P_a * N_c/2$ times, where P_a is the average out-degrees per peer, and $N_c = N_p/CN$ the average number of peers in a cluster. The average size of a query message in bytes is $C * L * q$, where C is the number of bytes to represent one character, L the average keyword length, and q the average query size. For the local/global aggregate information, it takes I bytes to represent the number of documents or peers, and I bytes to represent the term frequency for each of the q terms. There are in total one global aggregate information message and $N_c - 1$ local aggregate information messages from all peers in a cluster except the querying peer. Thus, the total bandwidth used to transmit the aggregate information is $I * (q + 1) * N_c$.

Using indexes at the document level, there are K document entries in each local ranking result. For each document entry, we assume it spends I bytes for the document ID and D bytes for the similarity value. Thus, the total number of bytes used for the local ranking results is $(I + D) * K * N_c$. Similarly, when using indexes at the peer level, it takes $(I + D) * k_p * N_c$ bytes in total for local peer ranking results. Subsequently, $C * L * q * k_p$ bytes are used to forward the query to the k_p peers, and $(I + D) * K * k_p$ bytes for local document ranking results.

The bandwidth usage formulas for a super-peer network are composed in a

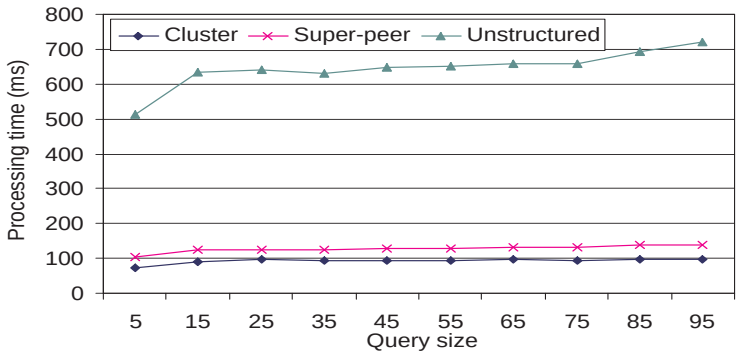

Figure 7: Average processing time in three overlays, using indexes at the document level over CACM, CRAN, CISI and MED collections.

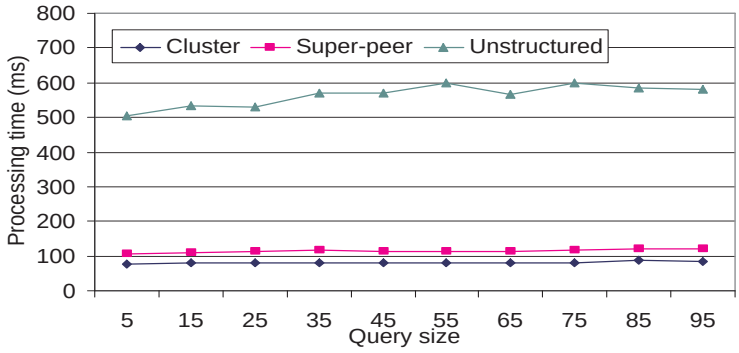

Figure 8: Average processing time in three overlays, using indexes at the peer level over CACM, CRAN, CISI and MED collections.

Table 3: Network bandwidth consumption per query

Overlay-Index level	Bandwidth usage formula (bytes)	Value (KB)
Cluster-DOC	$[C*L*q+I*(q+1)*N_c$ $+(I+D)*K*N_c]*P_a*N_c/2$	38
Cluster-PEER	$[C*L*q+I*(q+1)*N_c$ $+(I+D)*k_p*N_c]*P_a*N_c/2$ $+C*L*q*k_p+(I+D)*K*k_p$	26
Super-DOC	$[C*L*q+I*(q+1)*SN$ $+(I+D)*K*SN]*P_a*SN/2$ $+C*L*q+(I+D)*K$	20
Super-PEER	$[C*L*q+I*(q+1)*SN$ $+(I+D)*k_p*SN]*P_a*SN/2$ $+C*L*q+(I+D)*K$ $+C*L*q*k_p+(I+D)*K*k_p$	15
Unstructured	0	0

similar way, except that it takes $C*L*q$ bytes at the beginning to transfer a query from a normal peer to a super-peer, and $(I+D)*K$ bytes at last to return results to the normal peer. In an unstructured network, since all index information is available locally, there is no need to forward a query and the bandwidth usage for query processing is 0.

In Table 3, we calculate the bandwidth consumption using the formulas and the actual value of each parameter in our simulation environment. For instance, $C=1$, $I=4$, $D=8$. We see that the network bandwidth consumption using indexes at the peer level is $22-31\%$ less than that using indexes at the document level, due to the small value of k_p compared with K. The bandwidth consumption in a super-peer network is $41-47\%$ less than that in a PSCN, because the size of the strongly connected network formed among the super-peers is smaller than the average size of a cluster, resulting in less bandwidth spent on flooding.

Finally, we give a back-of-envelope calculation of the bandwidth consumption for index transmission. In a PSCN, each index, either at the document level or at the peer level, is transferred for $CN-1$ times; in a super-peer network, the index of each normal peer is required to be transmitted to one super-peer; and in an unstructured P2P network, the index of each peer is transmitted for N_p-1 times to be replicated everywhere. Thus, the unstructured network has the largest bandwidth consumption on index transmission, and the bandwidth consumption in the PSCN is $CN-1$ times of that in the super-peer network due to the index replication.

4.5 Storage cost

Peers are required to provide some space for index storage. To discuss the space usage, we first examine the size of the local index at a peer, either DI for the

Table 4: Storage cost per peer

Overlay-Index level	Storage space (bytes)	Value (KB)
Cluster-DOC	$DI * CN$	855
Cluster-PEER	$PI * CN + DI$	347
Super-DOC	$DI * N_p/SN$	1, 140
Super-PEER	$PI * N_p/SN + DI$	444
Unstructured-DOC	$DI * N_p$	5, 698
Unstructured-PEER	$PI * N_p + DI$	1, 990

Table 5: Storage cost in the entire network

Overlay-Index level	Storage space (bytes)	Value (KB)
Cluster-DOC	$DI * CN * N_p$	85, 500
Cluster-PEER	$(PI * CN + DI) * N_p$	34, 700
Super-DOC	$2 * DI * N_p - DI * SN$	11, 115
Super-PEER	$PI * N_p + DI * N_p$	7, 635
Unstructured-DOC	$DI * N_p * N_p$	569, 800
Unstructured-PEER	$PI * N_p * N_p + DI * N_p$	199, 000

document-level index or PI for the peer-level index.

For the document-level index, it takes $2 * I$ bytes to represent the document ID and the number of terms contained in a document, for each of the DN documents located at one peer. For each of the W keywords occurring at one peer, it spends $C * L$ bytes on the keyword itself, and I bytes on the number of the documents that contain the keyword. A keyword occurs in DL documents on average. For each of these documents, it uses I bytes for the document ID and I bytes for the term frequency in the document. Thus, $DI = 2 * I * DN + (C * L + I + 2 * I * DL) * W$, measured as 57KB in our experiments. For the peer-level index, it takes I bytes to represent the number of terms occurring at a peer, and $(C * L + I)$ bytes for each of the W keywords occurring at the peer. So, $PI = I + (C * L + I) * W$, measured as 19KB in our experiments, which is about one third of the size of the document-level index.

In a PSCN, in addition to its own local index, a peer needs to store $(N_p - N_c)/N_c = CN - 1$ indexes of peers from other clusters; in a super-peer network, a super-peer takes charge of $N/SN - 1$ indexes from normal peers; and in an unstructured P2P network, all of the N indexes are stored at each peer. Based on the measured values of DI and PI, we list the storage cost per peer (or super-peer) for each overlay in Table 4, and the storage cost in the entire network in Table 5. We see that in a PSCN, the maximum requirement of storage space for a single peer is $22 - 25\%$ lower than that in a super-peer network, even though the total storage cost is $4.5 - 7.7$ times higher due to the index replication across clusters.

4.6 Update handling

Finally, we study the cost incurred by updates in a PSCN. We first discuss document updates, and then turn to peer updates. Document updates include the addition of new documents as well as the deletion and the modification of existing documents. All of the updates are represented by document update vectors, with the term frequency set to be positive or negative accordingly. In the following, we restrict document updates to document additions for simplicity.

The cost incurred by document updates is reflected in the processing time used to merge document update vectors into an existing index, and the bandwidth used to transmit document update vectors. The latter is related to the average size of a document update vector. For each of the w keywords occurring in a document, it spends $C * L$ bytes on the keyword itself and I bytes on the term frequency, so the average size of a document update vector is $(C*L+I)*w$ bytes, or 420 bytes in our experiments.

Instead of handling document update vectors one at a time, we can combine several document update vectors to build an update index for batch processing. In Figure 9, we show the document update cost for peer-level indexes. The curves stand for the processing time, while the bars indicate the size of the update index. As the number of document updates increases, the processing time spent on merging document update vectors into an existing index increases linearly. In comparison, when using the update index, the time spent on merging remains at a stable low level, while the time used to build the update index grows linearly. However, the update index is built only once at the peer where a batch of document updates occur, while the merging process takes place at each peer receiving the update information. As to the size of the update index, it increases with the number of updates, which is the common trend over four different document collections.

Peer updates, either the joining or the leaving of peers, are simpler to handle than document updates. Since the index information of each peer is stored separately, there is no need to merge the updates into an existing index. Network bandwidth consumption is the size of the local index of the joining/leaving peer multiplied by the number of times for index transmission, which is similar to the discussion of the bandwidth usage for index transmission in Section 4.4.

4.7 Summary

We have experimented with ranked search in a PSCN in comparison with that in a super-peer or in an unstructured network. Through the experiments, we see that compared with the processing time spent on local aggregate information collection and local ranking calculation, the time used to merge local aggregate information or to merge locally ranked results, is negligible. This suggests it is beneficial to distribute the query processing load over peers; otherwise, the bottleneck will be at the super-peers in a super-peer network or at the querying peer in an unstructured network. As a result, the processing time and the storage cost per peer in a PSCN is the lowest among the three overlays.

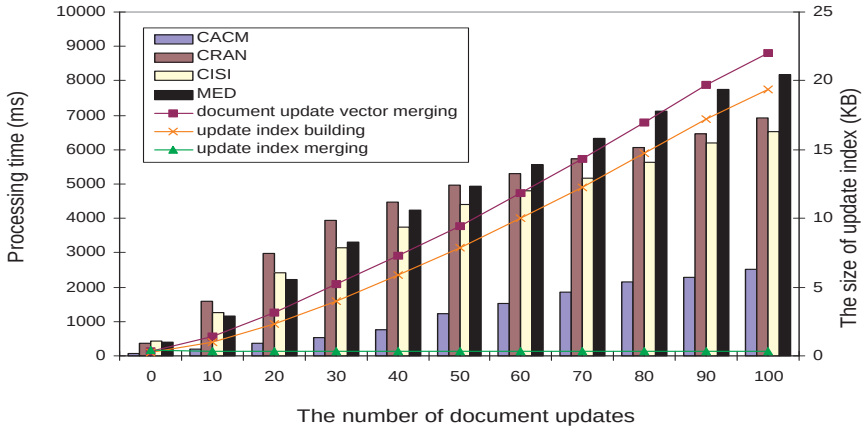

Figure 9: Average time spent on document update processing and the size of the update index, with different numbers of document updates, over CACM, CISI, CRAN and MED collections.

However, the downside of a PSCN is the flooding communication within a cluster and the index replication cost across clusters. The super-peer network wins on the network bandwidth usage and the total storage cost due to the directed query forwarding from normal peers to super-peers and the absence of the index replication among normal peers.

Additionally, we find that compared with document-level indexes, peer-level indexes save 70% of the processing time, 30% of the network bandwidth usage and 30% of the storage space, with a slight decrease in precision.

5 Conclusion

We have presented our approach to supporting the basic TF×IDF ranking for ranked search in a PSCN and have conducted extensive simulation experiments to study its performance in comparison with a super-peer network and an unstructured network. We find that ranked search can be done efficiently in a PSCN by taking advantage of the architectural features of the clusters. The most time-consuming tasks in the ranked search are distributed over the peers within a cluster, and the storage cost per peer is low.

In the future work, we plan to add more advanced indexing techniques, such as the LSI, to the basic TF×IDF method, to further reduce the index size and to improve the precision of ranked keyword search in a PSCN.

References

[1] Gnutella. http://www.gnutella.com/.

[2] Kazaa. http://www.kazaa.com.

[3] Napster. http://www.napster.com/.

[4] TREC. http://trec.nist.gov/.

[5] G. E. P. Box and M. E. Muller. A note on the generation of random normal deviates. *The Annals of Mathematical Statistics*, 29(2):610–611, June 1958.

[6] C. Buckley, A. Singhal, M. Mitra(, and G. Salton). New retrieval approaches using smart: Trec 4. In *The Fourth Text REtrieval Conference (TREC-4)*, pages 25–48, Gaithersburg, Maryland, USA, November 1995.

[7] B. F. Cooper and H. Garcia-Molina. SIL: Modeling and measuring scalable peer-to-peer search networks (extended version). Technical report, Stanford University, 2003.

[8] B. F. Cooper and H. Garcia-Molina. Studying search networks with SIL. In *Proceedings of IPTPS 2003*, pages 216–224, Berkeley, CA, USA, February 2003.

[9] F. M. Cuenca-Acuna and T. D. Nguyen. Text-based content search and retrieval in ad-hoc p2p communities. In *Proceedings of NETWORKING 2002 Workshops*, pages 220–234, Pisa, Italy, May 2002.

[10] F. M. Cuenca-Acuna, C. Peery, R. P. Martin, and T. D. Nguyen. PlanetP: Using gossiping to build content addressable peer-to-peer information sharing communities. In *Proceedings of HPDC-12 2003*, pages 236–249, Seattle, WA, USA, June 2003.

[11] C. R. Palmer and J. G. Steffan. Generating network topologies that obey power laws. In *Proceedings of GLOBECOM 2000*, volume 1, pages 434 – 438, San Francisco, CA, USA, 2000.

[12] S. Ratnasamy, P. Francis, M. Handley, R. M. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of SIGCOMM 2001*, pages 161–172, San Diego, CA, USA, August 2001.

[13] G. Salton, A.Wang, and C. Yang. A vector space model for information retrieval. *Journal of the American Society for Information Science*, 18:613–620, 1975.

[14] H. T. Shen, Y. Shu, and B. Yu. Efficient semantic-based content search in p2p network. *IEEE Transactions on Knowledge and Data Engineering*, 16(7):813–826, July 2004.

[15] I. Stoica, R. Morris, D. R. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of SIGCOMM 2001*, pages 149–160, San Diego, CA, USA, August 2001.

[16] C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. In *Proceedings of SIGCOMM 2003*, pages 175–186, Karlsruhe, Germany, August 2003.

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

[18] B. Yang and H. Garcia-Molina. Improving search in peer-to-peer networks. In *Proceedings of ICDCS 2002*, pages 5–14, Vienna, Austria, July 2002.

[19] B. Yang and H. Garcia-Molina. Designing a super-peer network. In *Proceedings of ICDE 2003*, pages 49–60, Bangalore, India, March 2003.