Efficient Semantic-based Content Search in P2P Network

Heng Tao Shen, Yan Feng Shu, and Bei Yu

Abstract

Most existing Peer-to-Peer (P2P) systems support only title-based searches and are limited in functionality when compared to today's search engines. In this paper, we present the design of a distributed P2P information sharing system that supports semantic-based content searches of relevant documents. First, we propose a general and extensible framework for searching similar documents in P2P network. The framework is based on the novel concept of Hierarchial Summary Structure. Second, based on the framework, we develop our efficient document searching system, by effectively summarizing and maintaining all documents within the network with different granularity. Finally, an experimental study is conducted on a real P2P prototype, and a large-scale network is further simulated. The results show the effectiveness, efficiency and scalability of the proposed system.

Index Terms

content-based, similarity search, peer-to-peer, hierarchical summary, indexing

I. Introduction

Peer-to-Peer (P2P) computing has recently attracted a great deal of research attention. In a P2P system, a large number of nodes (e.g., PCs connected to the Internet) can potentially be pooled together to share their resources, information and services. Many file-based P2P systems have already been deployed. For example, Freenet [5] and Gnutella [6] enable users to share digital files (e.g., music files, video, images),

H.T. Shen, Y.F Shu, and B. Yu are in Department of Computer Science, National University of Singapore. E-mail: {shenht,shuyanfe,yubei}@comp.nus.edu.sg

and Napster [7] allows sharing of (MP3) music files. However, these systems, including the most recent ones, only provide title-based search facility, which means that the end user cannot retrieve the content unless he knows its unique name. They lack support for semantic-based content search.

Current P2P search mechanisms can be classified into three types. First, a centralized index is maintained at a server, and all queries are directed to the server. An example of this approach is the Napster system [7]. However, with exponential growth in the Internet, it is unlikely that a centralized search engine is capable of performing efficient search. Second, the query will be flooded across the network to other peers - the query node will broadcast the query to its neighboring nodes who will then broadcast to their neighbors, and so on. Gnutella [6] is an advocate of this scheme. Clearly, such an approach will lead to poor network utilization. Yet another approach is the Distributed Hash Table (DHT) based scheme where the peers and data are structurally organized so that the location of a data (i.e., peer that contains the data) can be determined by a hash function. Chord [12] is an example that employs a DHT-based scheme. While this approach is scalable, it can only support exact match queries, and incurs the overhead of frequent reorganization as nodes leave and join the network.

In this paper, we address the problem of semantic-based content search in the context of document retrieval. Given a query, which may be a phrase, a statement or even a paragraph, we look for documents that are semantically close to the query. We propose a general and extensible framework for semantic-based content search in P2P network. The super-peer P2P architecture [18] which is more efficient for contents look-up is employed as the underlying architecture. To facilitate semantic-based content search in such a setting, a novel indexing structure called Hierarchial Summary Indexing Structure, is proposed. With such an organization, all information within the network can be summarized with different granularity, and then efficiently indexed. Based on this framework, we develop our distributed document search system in P2P network.

Our system has several key features. First, information of documents, peers and super peers are effectively summarized in a three-tier hierarchical structure: unit level, peer level and super peer level.

The state-of-the-art IR algorithms such as Vector Space Model (VSM) [15] and Latent Semantic Indexing (LSI) [10] are leveraged. Summaries are first represented as vectors, which are further optimized by LSI techniques and represented as high-dimensional points. Note that the summary information is much smaller than original information in size. It not only enables easy maintenance and update of information, it also reduces both storage and communication costs. The accuracy of summarization method can be tuned based on needs and resources. Higher degree of accuracy requires more information to be stored. In this paper, we study the various effects on the accuracy and update costs. Second, corresponding to the hierarchical summary structure, efficient hierarchical indexing structure is constructed on top of the summaries at the unit level, peer level and super peer level; and named respectively as local index, group index and global index. Summary Indexing facilitates K Nearest Neighbor (KNN) search which is the primary operation required in content-based non-exact match retrievals. Group index and global index are used to locate most relevant peers and peer groups quickly and local index speeds up the searching of most relevant documents. Third, we extend VA-file to support efficient KNN search for text retrieval based on similarity functions other than L_p distance functions. This is achieved by modifying the lower and upper bounds as filtering conditions. Fourth, dynamic updating in our indexing structure is handled efficiently. A peer's joining or leaving the network requires only incremental updates. Re-summarizing and re-indexing is invoked only if the summary is no more effective in representing the information.

We have implemented a prototype P2P document retrieval system that employs our method, and evaluated the system performance over a network containing 30 nodes (PCs). Our experimental results show that our hierarchial summary method achieves better precision than existing methods. To further study the scalability of the system, we also implemented a simuation model. Our simluation results confirm the efficiency of our hierarchical indexing method even in very large P2P network.

The rest of this paper is organized as follows. Section II provides some related work. In Section III, we introduce our Hierarchical Summary Indexing framework. Based on the proposed framework, we develop our peer-based semantic text search system in Section IV. We address updating issues in Section V and present the cost model of our Hierarchical Summary Indexing Structure in Section VI. We present results of an extensive performance study in Section VII, and finally, we conclude with directions for future work in Section VIII.

II. Related Work

We will first review previous work on P2P architecture. [16] provides an analysis of $hybid$ P2P architecture, develops an analytical model and uses it to compare various hybrid P2P architectures. [18] extends [16]'s hybrid architecture to design $super$-peer network, which strikes a balance between the inherent efficiency of centralized search, and the autonomy, load balancing and robustness to attacks provided by distributed search.

Much research effort has focused on improving search efficiency by designing good P2P routing and discovery protocols. However, current systems support only simple queries. For example, Freenet[5], Gnutella[6] and Napster[7] only provide filename-based search facility, which means that the end user cannot retrieve content unless he knows a file's unique name. Queries are broadcast to neighbors which in turn disseminate the queries to their neighbors and so on. Thus, these systems can lead to long response time. Chord[12] and CAN[11] are designed for point queries and focus only on the problem of query routing and object allocation. [17] and [3] support keyword queries with regular expressions. Hence so far the query issued by clients are up to context of keyword's complexity and for keyword matching only. More recently, PlanetP [4] presents a distributed text-based content search algorithm in P2P communities. Each peer has a summary produced by VSM. A local inverted index is then built on this summary. However, to our knowledge, there has not been much work done to facilitate efficient *semantic-based content* search for document retrieval in P2P sharing systems. Issue on fair load distribution has also been addressed by [13]

Summary techniques are crucial in P2P systems. Due to limit on network bandwidth and peer storage, it is not practical to transmit the complete information of a peer to the other peers in the network. Moreover, a peer usually contains thousands of shared files or more. To decide which peer to route the query to needs a similarity comparison between the query and peer's information. From the above discussion, it is clear that effective summarization of peer information is absolutely needed in P2P network. So far, the only known summarization technique for text documents in P2P systems is keywords representation. Existing P2P systems, such as [17], [3] etc, summarize the peers/documents by keyword vectors which contain pairs of keyword and its weight. Given a query, which is also represented as a vector, the similarity between the query and the summary of peers/docuemnts are then computed. However, such techniques are limited to exact keyword matching only and cannot be applied for semantic-based content search. In this paper, we propose a hierarchical summary indexing structure for efficient semantic-based content search in super-peer P2P network, which can support complex semantic-based queries.

Another related area is high-dimensional indexing. In the literature, many high-dimensional indexing methods have been proposed. A survey can be found in [1]. However, existing methods are typically not efficient for more than 30-dimensions and are not scalable [14] due to the ′dimensionality curse′ phenomenon when the dimensionality reaches higher. VA-file [14] however, has been shown to be superior in nearly uniform datasets by L_P distance functions. In this paper, we extend VA-file to support a different similarity metric for document similarity search.

III. A General Framework for P2P-based Semantic Search

In this section, we present a novel Hierarchical Summary Indexing framework for P2P-based document search system. We shall first discuss the super-peer P2P architecture, and then look at how such a structure can facilitate the design of the proposed framework.

A. Super-peer P2P Network

In our framework, we have adopted a super node P2P architecture [18]. A super-peer P2P network is a P2P network that consists of two types of peers: super peers and their clients (often called peers directly). A super peer is a node that acts both as a server to a set of clients, and as an equal in a network of super peers. A peer group is formed by a super-peer and its clients. A straightforward query processing

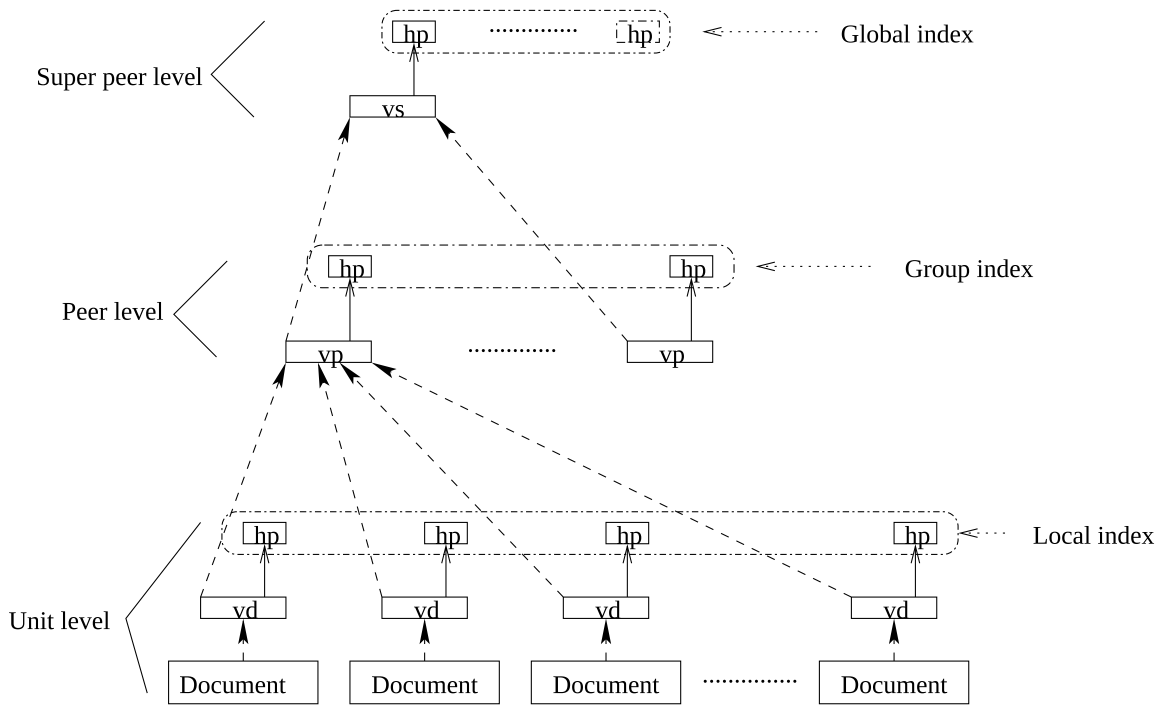

Fig. 1. Hierarchical Summary Indexing Structure

mechanism works as follows. A peer (client) submits its query to the super peer of its group. The super peer will then broadcast the query to other peers within the group. At the same time, the super peer will also broadcast the query to its neighboring super peers. A neighboring super peer will broadcast the query to its clients, and further forward the query to its neighboring super peers. This process is repeated until some criterion is satisfied, for example, a system specified TTL value that is decremented each time the query is broadcast, and the query is dropped when TTL = 0.

Clearly, while in such a simple approach a super peer broadcasts the query, its communication overhead is high. We sought to minimize this overhead using the concept of Hierarchical Summary Indexing Structure.

B. Hierarchical Summary Indexing Structure

Summarization is a necessary step for efficient searching, especially when the amount of information is very large. A summary is a very compact representation. In our framework, we introduce a new interesting concept, Hierarchical Summary Indexing Structure (Summary and Indexing), which is closely related to the super-peer P2P architecture we employed. Our scheme essentially summarizes information at different levels.

We have employed three levels of summarization in our framework. The lowest level, named as *unit level*, an information unit, such as a document or an image, is summarized. In the second level, named as *peer level*, all information owned by a peer is summarized. Finally, in the third level, named as *super level*, all information contained by a peer group is summarized. Clearly, each level covers wider information scope than its former level, while performing coarser summarization. Figure 1 depicts such a structure for document summary.

With the summary information, queries only need to be forwarded to nodes that potentially contain the answers. Each super peer maintains two pieces of summaries: the super level summaries of its group and its neighboring groups, and peer level summaries of its group. By examining super level summaries, a super peer can determine which peer group is relevant. Similarly, by examining peer level summaries, a super peer can determine which of its peers have the answers.

Note that the summarization method is domain specific. In fact, all three levels may use same or different summarization methods. Generally, there is a tradeoff between summarization accuracy and the costs of storage and communication. Higher degree of accuracy requires more information to be retained, thus more storage and more communication overhead incurred.

Since the number of summaries may be large, to further improve the efficiency of the system, we maintain indexes on the summary information. Figure 1 also shows each level of summary has a corresponding index built on top of it. We name the three indexes for unit, peer and super level summaries as *local index*, *group index* and *global index* respectively. Figure 2 shows the hierarchical summary indexes in a peer group. In each peer, there's a local index for the retrieval of locally stored information, built from summaries of all information units within the peer. With summaries from all peers within a group, each super peer maintains an index for quickly locating specific peers in the group which may contain the needed information. Finally, each super peer also has an index on summaries from all neighboring peer

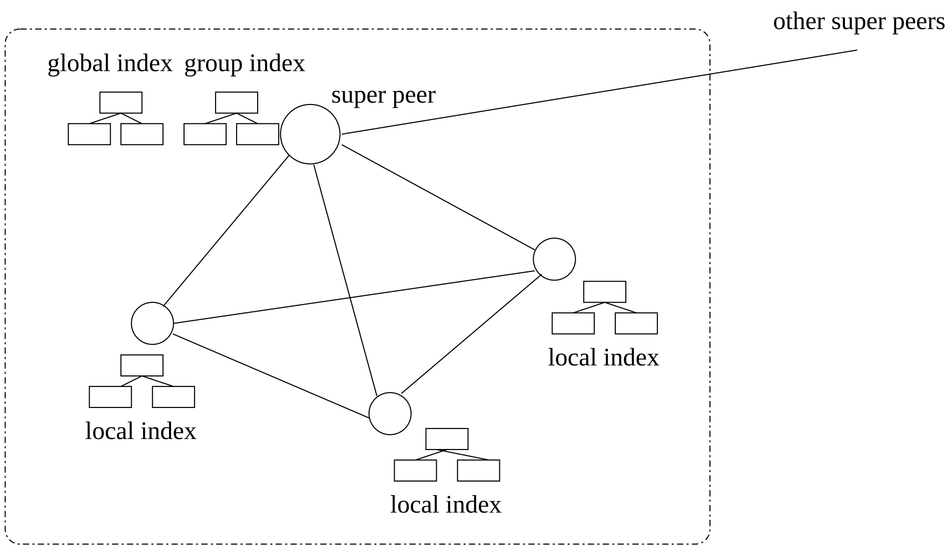

Fig. 2. Summary indices in a peer group.

groups to limit the search to specific peer groups.

Also note that there is no restriction on the index methods to be used and any index method (e.g. hash table, index trees) can be used. In fact, our framework is general enough to allow each peer to autonomously deploy their preferred indexes.

To summarize, in our framework, information searching can become more guided: a peer group is first decided, then a peer, finally, an information unit. Of course, searching performance largely depends on how a system built on this framework is implemented. In the next section, we will describe our semantic-based content search system in detail.

IV. A Semantic-based Content Search System

Our system facilitates distributed document searching in P2P network. Suppose there are a large number of peers in the network, and each peer contains a large number of documents, what we want to achieve is to find the most relevant documents as quickly as possible, given a semantic query, such as a sentence. Built on the above framework, the system first needs to consider how documents can be effectively summarized, and then how summaries can be efficiently maintained by indexing techniques. After that, the real content-based search can begin.

A. Building Summary

To be consistent with the framework, the document summarization is also done in levels - *document level, peer level and super peer level*. For each level, our summarization process consists of two steps by techniques of Vector Space Model (VSM) [15] and Latent Semantic Indexing (LSI) [10] respectively. Briefly, in VSM, documents and queries are represented by vectors of weighted term frequences. Three factors may be used in term weighting, i.e., the term frequency (TF), the inverse document frequency (IDF), and the normalization factor. TF represents how frequently a term appears in a document, IDF represents how frequently the term also appears in other documents, while the normalization factor is used to reduce the side-effect of different document sizes on weights. TF $\times$ IDF is the most frequently used equation for calculating weights, which means that a term is important only if it can differentiate a document from others. Similarity comparisons among documents and/or between documents and queries are made via the similarity between two vectors, such as the dot product of two vectors.

Latent Semantic Indexing (LSI) has been proposed to overcome synonymy, polysemy, and noise problems in information retrieval. LSI discovers the underlying semantic correlation among documents by building a concept space. A technique known as Singular Value Decomposition (SVD) is used to reduce this concept space into a much lower dimensionality, reflecting the major associative pattern in documents, while ignoring the smaller and less important influences. With the concept space, searching is based on concepts, rather than on individual terms. For each document, it usually contains a large number of terms. However, since the frequency of terms in a document typically follows a Zipfian distribution, i.e., a small number of keywords can categorize a document's content, thus we first use VSM to represent documents as vectors, which is the first step of summarization, and each component of the vector corresponds to the importance of a word(term) in all the documents of a peer. Document vectors are then further summarized by SVD to be high-dimensional points, which is the second step of summarization. This summarization step reduces a very high-dimensional space (of tens of thousands) to a much smaller one (of less than two hundreds) to facilitate indexing in each peer. Before SVD is used, we union all documents' terms to build a dictionary for each peer. Note that this dictionary is built dynamically, and different peers may have different dictionaries. First all terms that represent documents are merged to form a dictionary of a peer, then according to this dictionary, each document vector is mapped to the dimensionality of the dictionary. Clearly, each vector will be of very high dimensions. To reduce this large dimensional space, SVD is then applied.

We can perform the same process as the above at the peer level, since each peer's dictionary is still represented by a vector. First a peer group dictionary is formed by performing union on all peers' dictionaries, and then SVD is applied to produce the high-dimensional points. In the same way, we can build the global level summaries on super peers.

Algorithm 1: Building Hierarchical Summaries

```
1.  for each peer
2.      for each document
3.          Generate its vector $vd$ by VSM
4.      Generate peer weighted term dictionary $vp$
5.      for each document vector $vd$
6.          transform it into D(vp) dimensionality
7.          generate high-dimensional point for $vd$ by SVD
8.      Pass $vp$ to its super peer
9.  for each super peer
10.     Generate group weighted term dictionary $vs$
11.     for each $vp$
12.         transform it into D(vs) dimensionality
13.         generate high-dimensional point for $vp$ by SVD
14.     Pass $vs$ to other super peers
15.     Generate global weighted term dictionary $vn$
16.     for each $vs$
17.         Transform it into D(vn) dimensionality
18.     Generate high-dimensional point for $vs$
```

Fig. 3. Building Hierarchical Summaries.

Algorithm 1 indicates the main routine of building summary in the hierarchical structure as shown in Figure 3, where D represents the dimensionality of a vector. As shown, the algorithm is bottom-up. For each peer, each of its documents is first represented by a vector of document (vd) by Vector Space Model(VSM) (line 3). Then all vds are combined to generate the peer's weighted term dictionary - vector of peer (vp) by performing an union operation on vds (line 4). Each vd is transformed into D(vp) dimensionality based on vp (line 6), followed by being reduced into a much lower dimensional point (denote its dimensionality as D_{doc}) by SVD (line 7). So far documents' summaries - D_{doc} dimensional points, have been built. Next, each vp is passed to its super peer (line 8). Each super peer will generate its group's weighted term dictionary - vector of super peer (vs) by performing an union operation on its vps (line 10). Similarly, vps are transformed into D(vs) dimensionality and reduced into a much smaller dimensional point (denote its dimensionality as D_{peer}) by SVD (lines 11-13). So far, each peer's summary - D_{peer} dimensional point, has been generated. After a super peer receives other super peers' vss, it repeats the same step as generating group's summary (lines 15-18) by constructing a global network's weighted term dictionary - vector of network (vn), and then generates its high-dimensional point. A vivid processing routine is also shown in Figure 1. We illustrate the algorithm in Example 1.

EXAMPLE 1 (AN EXAMPLE OF HIERARCHICAL SUMMARY BUILDING) *Table I provides a small P2P network with eight documents, di^n_m represents the i^{th} document which is in m^{th} peer of n^{th} group. The process of summary building is depicted in Figure 4, where the summary dimensionality for each level is reduced to 2. For simplicity, the weight of a term is represented by its frequency only. As we can see, vectors of documents vds (excluding stop words) within a peer form the vector of peer vp. Based on vp, each vd is transformed into D(vp)-dimensional vector (each digit represents the weight of the corresponding word in the dimension) which is then reduced into a 2-dimensional document summary by SVD. Take a look at the first peer which contains documents $d1^1_1$ and $d2^1_1$. Both documents are merged to form its vp of (data 1, join 1, monitor 1, XML 2, web 1) together with term weights, where D(vp) is 5. Based on vp, both documents are mapped into 5-dimensional vectors of (1,0,1,1,1) and (0,1,0,1,0) respectively, which are in turn reduced into a much lower 2-dimensional points by SVD. Similarly, to generate peer summaries, vps within a group are first combined to form the vector of super peer vs. Based on vs, each vp is transformed into D(vs)-dimensional vector which is then reduced into a 2-dimensional peer summary by SVD. Same process is applied to generate a supper peer's summary.*

Id	*Documents*
$d1_1^1$	*Monitoring XML Data on the Web*
$d2_1^1$	*Approximate XML Joins*
$d3_1^2$	*Outlier Detection for High Dimensional Data*
$d4_1^2$	*High Dimensional Indexing using Sampling*
$d5_2^1$	*Document clustering with committees*
$d6_2^1$	*Document clustering with cluster refinement*
$d7_2^2$	*Title language model for information retrieval*
$d8_2^2$	*Document summarization in information retrieval*

TABLE I

A TABLE OF DOCUMENTS.

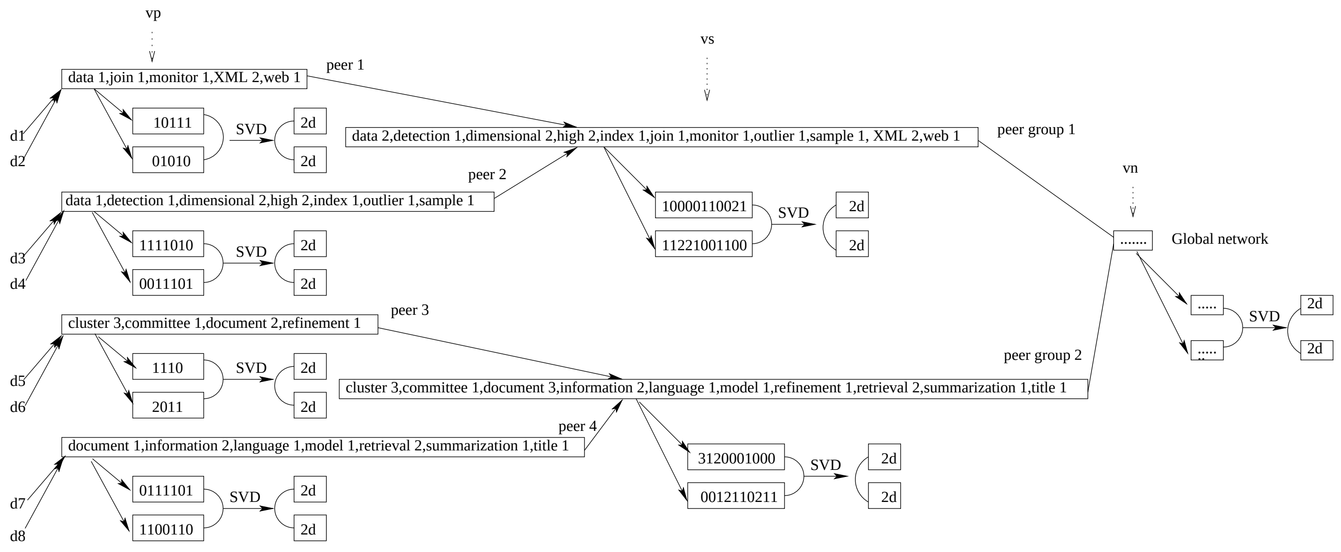
vp
vs
vn
data 1,join 1,monitor 1,XML 2,web 1
peer 1
10111
01010
SVD
2d
2d
d1
d2
data 2,detection 1,dimensional 2,high 2,index 1,join 1,monitor 1,outlier 1,sample 1, XML 2,web 1
peer group 1
peer 2
data 1,detection 1,dimensional 2,high 2,index 1,outlier 1,sample 1
1111010
0011101
d3
d4
10000110021
11221001100
....
Global network
.....
.....
cluster 3,committee 1,document 2,refinement 1
peer 3
1110
2011
d5
d6
peer group 2
cluster 3,committee 1,document 3,information 2,language 1,model 1,refinement 1,retrieval 2,summarization 1,title 1
peer 4
document 1,information 2,language 1,model 1,retrieval 2,summarization 1,title 1
3120001000
0012110211
0111101
1100110
d7
d8

Fig. 4. An example of hierarchical summary building.

B. Indexing Summary

In the last section, we have looked at how to build the summary representation - high-dimensional point for *document level*, *peer level*, and *super peer level* respectively. Each peer may contain thousands of documents. Similarly, each group may have a large number of peers and the whole P2P network may include a large number of groups. As the network size grows, efficient searching in high-dimensional space becomes prevalently important. Hence at each level of summary, we build an efficient indexing structure: a *local index* at *document level* on single peer's documents, a *group index* at *peer level* on peers in a group, and a *global index* at *super peer level* on all groups' super peers. In this paper, we extend the existing high-dimensional indexing technique - VA-file (Vector Approximation file)[14] to perform efficient K Nearest Neighbor (KNN) searching for text documents. We chose VA-file for several reasons: 1) VA-file outperforms sequential scan in high-dimensional space while other indexing techniques fail. 2) In P2P environment, peers are changing dynamically and frequently. VA-file is extremely computationally efficient for insertion. It is a flat structure and when a new point is to be inserted, it can be simply appended to the file without any other cost.

A VA-file represents the original data points by much smaller vectors. It represents each dimension by b bits, where each dimension's range is equally divided into 2^b intervals. By sequentially scanning the VA-file of a dataset, VA-file can filter most of the data points and return a small number of candidates for data access. To perform filtering, lower and upper bounds on the distance from the query to the data points have to be computed. KNN search is performed in two phases. First, the VA-file is sequentially scanned to filter the false $'$positive$'$. As each VA is processed, if its lower bound is greater than the current K^{th} smallest upper bound then it is filtered. Otherwise, it is added into the candidate list and the K smallest upper bounds are updated. In the second step, candidates are randomly accessed in ascending order of lower bounds until the lower bound of the next candidate is greater than the K^{th} smallest distance.

However, VA-file was proposed to search the nearest neighbor with smallest L_p distance. But in text information retrieval, the relevance between two documents is usually indicated by the similarity measured by their dot product, i.e.,

$$sim(Q, P) = \sum_{i=0}^{D-1} Q[i] * P[i]$$

where Q and P are two high-dimensional point representations of documents, and D is their dimensionality. Consequently, the nearest neighbor refers to the point with the largest similarity value. Hence VA-file becomes invalid for such similarity matric. Here we extend VA-file to use similarity metric for text retrieval.

The key of effective pruning in VA-file is the lower and upper bounds computation. The point satisfying the condition of its lower bound greater than the K^{th} smallest upper bound can be safely pruned. By

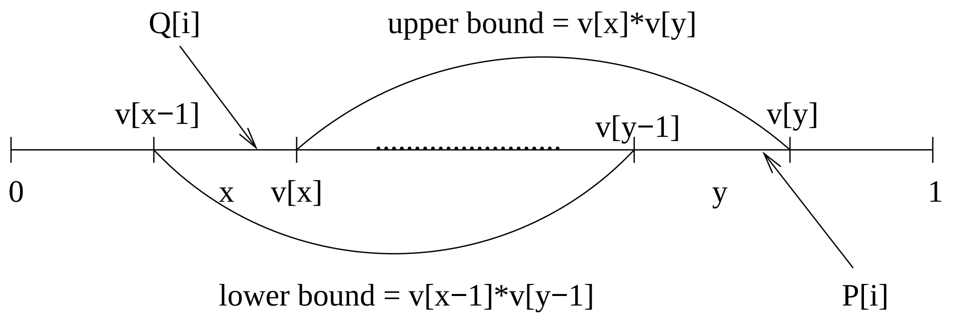

Fig. 5. An example of lower and upper bounds computation for one dimension.

using the similarity metric, the lower and upper bounds computation are extended as follows. Assume at the i^{th} dimension, the values of Q[i] and P[i] are mapped into the x^{th} and y^{th} intervals. Then the lower bound similarity sim^i_{lb} and upper bound similarity sim^i_{ub} on i^{th} dimension are computed as below:

$$\begin{cases} sim^i_{lb}(Q,P) = v[x-1] * v[y-1] \\ sim^i_{ub}(Q,P) = v[x] * v[y] \end{cases}$$

where v[x] is the maximal value in interval x. Figure 5 shows an example for one dimension's lower bound and upper bound computation.

Hence the overall lower and upper bounds on full dimensions is the sum of sim^i_{lb} and sim^i_{ub} for all i. Correspondingly, the pruning criterion has to change. For such similarity metric, the point satisfying the condition that its upper bound is less than K^{th} largest lower bound can be safely pruned. In the second step of the search, candidates are randomly accessed in descending order of upper bounds until the upper bound of the next candidate is less than the K^{th} largest similarity.

C. Query Processing

Having discussed how the summary indices are constructed, we are now ready to present the semantic-based query processing based on the proposed summary indexing hierarchy. In this section, we look at how our hierarchical indexing structure can be used to support efficient evaluation of a query. Figure 6 depicts how a query is being processed in a P2P network. In the figure, dark arrow indicates the direction of a query being transmitted and blank arrow indicates the route of results being returned.

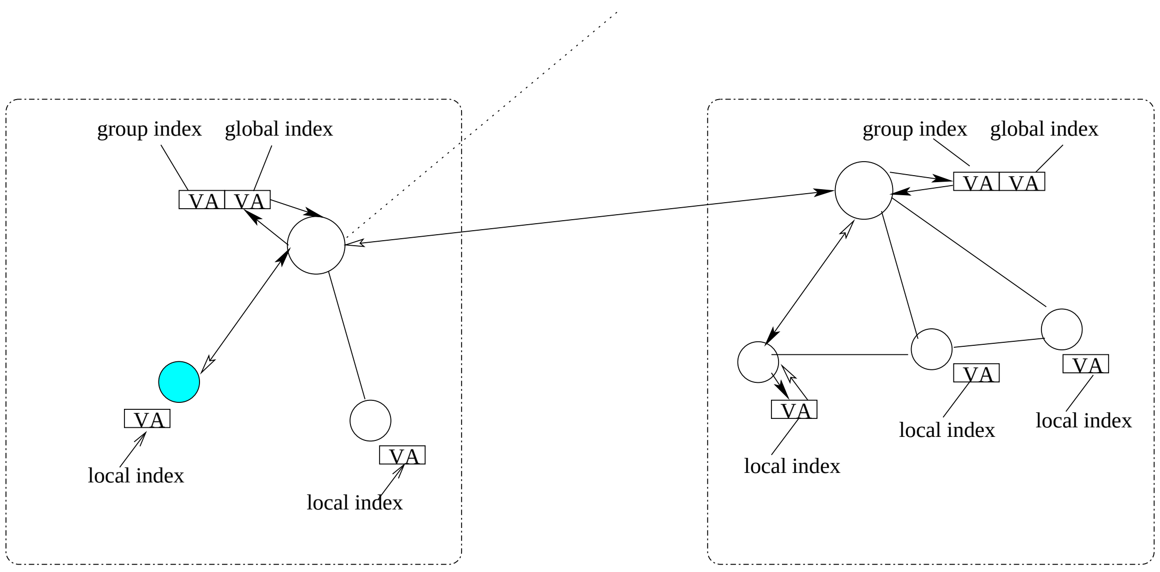

Fig. 6. The routine of query processing initialized by the dark peer.

When a peer issues a query Q, Q is first passed to its super peer, followed by the hierarchical indexing search in order of global index, group index and peer index, which is the reverse order of the summary construction. Notice that there is no broadcast activity during the whole query processing in our structure. This is one of the main achievements by our summary indices.

- Global index: When a query reaches the super peer, it is first mapped into its high-dimensional point in global index space, followed by KNN searching in the global VA-file. Notice here that the global network for a super peer refers to the network which can be reached by the super peer. The query is then transmitted to the K_{group} most relevant groups.
- Group index: At each group, the query is first mapped into its corresponding high-dimensional point in its group index space, followed by KNN searching in the group VA-file to select the K_{peer} most relevant peers. The query is then broadcast to the K_{peer} most relevant peers.
- Peer index: At each peer, similarly, the query is first transformed into the peer level's high dimensional point, followed by KNN searching in the local index to select the K_{doc} most relevant documents for final document similarity measure.

Finally, each peer returns K_{doc} most similar documents to the client peer issuing the query. At each level of index, the K value may be set based on the user's requirements.

V. Updating Issues

One crucial difference between P2P and traditional information retrieval is that P2P network is dynamic in nature. A peer can join and leave the network at any time. Hence the summarization and indexing techniques have to be able to handle dynamic operation efficiently. We propose the following peer insertion algorithm in our hierarchical indexing structure as shown in Figure 7.

Algorithm 2: Peer Insertion

```
1.  Build peer's local index
2.  Pass peer's $vp$ to its super peer
3.  if $AIR_{group} > \theta_{group}$
4.      Re-build and index group peers summary
5.      Update super peer's $vs$
6.      Broadcast $vs$ to other super peers
7.      for each super peer
8.          if $AIR_{global} > \theta_{global}$
9.              Re-build and index super peers summary
10. else
11.     Generate peer's high-dimensional point
12.     Insert the point into group's index
```

Fig. 7. Process of a peer joining the P2P Network

When a peer joins a P2P network, its documents are first summarized into high-dimensional points on which a local index is built (line 1). Meanwhile, the peer's first level of summary - vp is passed to its super peer (line 2). Every super peer records the accumulated information which has been updated. To measure such information, we define the following parameter called Accumulated Information Ratio (AIR) as follows:

$$AIR(dic, dic_{future}) = \frac{\sum_{i=0}^{D[dic_{future}]} |dic[i] - dic_{future}[i]|}{\sum_{i=0}^{D[dic_{future}]} dic[i]}$$

Where dic and dic_{future} are the current and future weighted term dictionary at group or global level (vs or vn) . $D[dic_{future}]$ is the dimensionality of the future dictionary. Whenever a peer joins the group, the future weighted term dictionary is updated by adding the weight to the corresponding term. If new terms appear, new entries will be created in the future dictionary.

If the AIR at the group level is less than the predefined threshold θ_{group}, the new peer's high-dimensional point is generated and inserted into the index (lines 11-12). Recall that we use VA-file technique for summary indexing. Insertion into the group VA-file (group index) is just simply to append the point into the end of the file. Otherwise, the whole group's index has to be rebuilt (line 4) based on the future dictionary. At this time, future dictionary is treated as current dictionary and a new future dictionary is initialized to be current dictionary. Then the super peer's summary is updated and broadcast to other super peers (lines 5-6). Each super peer then checks if it is necessary to update the whole network's super peers' summaries and re-built the global index (lines 7-9). The threshold on AIR ensures that the indexing are re-built only when the information has been updated significantly. Hence the most frequent operation is the insertion operation to VA-file, which takes constant cost only.

VI. Cost Models

In this section, we will evaluate our hierarchical summary indexing structure in a super-peer network based on the following types of metrics: *Storage Overhead*, *Query Response Time*, $Load$ and *Accuracy of Results*. Furthermore, the cost for indexing construction/updating is also estimated.

A. Storage Overhead

The storage overhead in our structure includes peer overhead and super peer overhead. For peer overhead, each peer contains its documents' summary, local index built on the document summary, local current and future dictionaries, together with the SVD's Singular Vectors. Hence the total peer Storage Overhead (SO) is:

$$SO_{peer} = D_{doc} * N_{doc} + VA_{local} + 2D(vp) + D_{doc} * D(vp)$$

where D_{doc} is the dimensionality of summarized high-dimensional points of documents, N_{doc} is the number of documents in the peer , VA_{local} is the size of local VA-file on points of documents, and D(vp) is the dimensionality of peer's term dictionary (assume current and future dictionary have approximately same dimensionality).

Usually the value of D_{doc} is about 100, and D(vp) is about thousands. $D_{doc} * N_{doc}$ represents the size of documents' summaries. Given that each dimension is represented by b bits, the size of VA is $\frac{32}{b}$ of summaries, assuming each dimension of summary is 4-byte long. Obviously, when N_{doc} is very large, $SO_{peer} \approx D_{doc} * N_{doc} + D_{doc} * D(vp)$.

Similarly, each super peer contains two sets of data: summaries, index of summaries, term dictionaries and Singular Vectors at group level and global level. Hence the total super peer Storage Overhead is:

$$SO_{super} = D_{peer} * N_{peer} + VA_{group} + 2D(vs) + D_{peer} * D(vs)$$
$$+D_{super} * N_{super} + VA_{global} + 2D(vn) + D_{super} * D(vn)$$

where D_{peer} and D_{super} are the dimensionality of summarized high-dimensional points of peers and super peers, N_{peer} and N_{super} are the number of peers in the group and super peers in the network. VA_{group} and VA_{group} are the sizes of group VA-files on points of group peers and global VA-file on points of super peers. D(vs) and D(vn) are the dimensionality of group and global term dictionary respectively.

B. Query Response Time and Load

The essential focus of our work is to perform effective and efficient semantic-based content search on text documents in the P2P environment. Here we derive simple brief cost model for *Query Response Time* and $Load$ in our hierarchical summary indexing structure.

In P2P network, the Query Response Time comprises network delay and peer's processing time. Network delay is affected by the network bandwidth and network $Load$. Here we define $Load$ as the total amount of information the network must transmit. For simplicity, we measure the $Load$ as the number of messages being processed in the network. Given the fixed network bandwidth and a time period, the number of messages is the major factor affecting the network traffic. A peer's processing time is the time to search the K most relevant peers and/or documents. Without summarization, the query has to compare with a peer's/document's complete information. And without indexing, the query has to compare with every single peer/document. Given the fixed CPU power, effective summarization and efficient indexing become two keys.

Our hierarchical indexing structure is designed to avoid network delay and speed up the processing time. In our structure, global index and group index are used to quickly determine which group and peer to be searched next. This avoids extensive broadcast cost. As for the peer's processing time, local index quickens the local document search. Effective summarization condenses large amount of information into small size and makes efficient indexing possible.

In our hierarchical indexing structure, given a query, the times of a query being forwarded, is:

$$Times_{query} = 1 + K_{group} + K_{group} * K_{peer}$$

The client peer first forwards its query to its group's super peer. Its super peer then searches its global index and selects the K_{group} most relevant groups to which it forwards the query. In each selected group, the super peer searches its group index and forwards the query to the K_{peer} most relevant individual peers.

At each level of index, KNN search is performed. Correspondingly, the total processing time is computed as:

$$Time = Time_{global} + K_{group} * Time_{group} + K_{group} * K_{peer} * Time_{local}$$

Where $Time_{global}$ refers to the processing time of KNN search in global VA-file. Clearly, the efficiency of indexing technique determines the performance. In the VA-file, the total response time for KNN search mainly includes two parts: the time to scan the VA file and the time to compute the lower and upper bounds if the number of candidates is small. It has been shown that it outperforms sequential scan in high dimensional space.

As for the accuracy of results achieved by our summary technique, they will be measured in the experiments section.

C. Cost of Updating

Updating cost is another important factor which may affect the overall performance. It consists of two parts: the processing time to update and the load for the update.

At document level, given the dimensionality of its peer dictionary - D(vp), the dimensionality of summarized document - D_{doc} and the number documents in the peer - N_{doc}, the time to generate document summary by SVD is $O(N_{doc} * D(vp)^2 + N_{doc} * D_{doc}^2)$, and the time to construct local VA-file on top of document summary is $O(N_{doc} * D_{doc})$. Since D(vp) is expected to be much larger than D_{doc}, the total processing time to construct local index is approximately $O(N_{doc} * D(vp)^2)$. Similar formulas can be derived for peer level and super peer level.

Whenever a peer joins a group, its local index is built and its dictionary - vp is passed to its super peer. Hence the load is the peer's vp. In our peer insertion algorithm, if no re-building of group indexing occurs in the super peer, the joined peer is first mapped to D_{peer} dimensional summary point, and then its VA is appended to the VA-file, which takes constant time.

However, if group summary rebuilding is invoked, the total cost will include two more portions: super peer's processing time for group indexing building, and the broadcast load of the super peer's summary - vs to other super peers. The processing cost of group indexing building is approximately $O(N_{peer} * D(vs)^2)$ as explained above. The times of the super peer's summary being broadcast is proportional to the number of super peers which can be reached by this super peer. If a super peers' summary rebuilding is invoked, each reachable super peer performs the same process of summary indexing, which approximately takes $O(N_{super}^2 * D(vn)^2)$ more processing time, by assuming all super peers can be reached.

It can be expected that directly inserting peer into group index is much more frequent than rebuilding of group index, which in turn is more frequent than rebuilding of global index.

Obviously, in the process of indexing building/rebuilding, summary generation by SVD is the most time consuming. To reduce the cost of SVD, sampling techniques can be applied to achieve a better trade-off

Name	Default Value	Description
Network Type	Power-Law	Topology of network, with outdegree 3.2
Max_User_Wait_Time	60s	Time for a user to wait an answer
Query Rate	8e-3	The expected number of queries per user per second
TTL	5	Time-To-Live of an message
Network_Size		Number of peers in the network
Peer_Group_Size		Number of peers in each peer group
K_{group}		Number of super peers to return
K_{peer}		Number of peers for a super peer to return
K_{doc}		Number of documents for a peer to return

TABLE II
PARAMETERS AND SETTINGS.

between time and accuracy. In the experiment section, we will see how the sampling technique can help to reduce the processing time while keeping a high accuracy. Furthermore, we will also see how stable the SVD technique can be in producing the accurate summary information as peers keep joining the network and when the index need to be rebuilt.

VII. EXPERIMENT

We have evaluated the proposed hierarchical summary and indexing scheme in a real P2P setting as well as via simulation. In this section, we report the results of the performance study.

A. Experiment Setup

Table II gives some experiment parameters and their default settings for both the real system and the simulator respectively. The system has *Network_Size* peers, and *Peer_Group_Size* peers per group. Assuming that each group has the same number of peers, the total number of peer groups, N_{super}, can be determined by *Network_Size*/*Peer_Group_Size*. The topology of super-peer is based on power-law, generated according to the PLOD algorithm presented in [9] with the average outdegree of 3.2.

We compare our proposals with the methods applied in [4], both on summary technique and indexing technique. Precision of results, Query Response Time, and Load are three metrics we are interested, which are used to measure system performance.

	MED	CISI	CACM	TIMES
Number of documents	1033	1460	3204	425
Number of queries	30	76	64	83
Number of terms occurring in more than one document	5831	5743	4867	10337

TABLE III
CHARACTERISTIC OF REAL DATABSETS.

B. Retrieval Precision

In this experiment, we examine the effectiveness of our summary technique. We first implement a relatively small real network to show that our proposals are very practical and applicable to P2P systems.

Our real network has 30 nodes. We use 4 benchmark collections of documents which were used by Smart [2], together with their queries and human ranking. Table III presents the characteristics of the datasets. We divide each dataset into a number of smaller document collections, so that we have 30 document collections in total, each of which contains around 200 documents. Next, we allocate these 30 collections to the 30 nodes in the network individually. We then cluster the 30 nodes into 6 groups. One peer in the group is appointed as the super-peer randomly. Finally we build our hierarchical summary structure in the super-peer architecture and evaluate it by comparing with the VSM summary technique used in [4]. The software we used to compute the truncated SVD is provided in SVDPACK package [8].

1) Effect of Dimensionality: We begin by looking at the effect of dimensionality of summaries. Notice the dimensionality of high-dimensional points (i.e., the dimensionality of summaries) generated by SVD affects the retrieval precision. Considering the hierarchical architecture of our proposed semantic-based content search system, here we test the retrieval precision separately in all the three layers, that is, the effectiveness of retrieving documents in the local machine, selecting peers in one peer group and locating peer groups in the whole network.

Different datasets may have different optimal dimensionality to achieve the best precision by using SVD [10]. We select the peer group that contains MED dataset to present the result and show how the dimensionality of summary affects the precision result. The precision is measured by the ratio of the number of relevant documents over the number of returned documents. Each benchmark collection provides the queries and their corresponding relevant documents as shown in Table III. First, we look at the local document level by selecting a single peer in the group. Figure 8 shows the changes of the average precision when the summary for the documents is reduced to different dimensions with SVD technique. The precision of the VSM method is also tested and marked in the figure for comparison. We can see that the precision of SVD by SVD reaches the highest point when the dimension is reduced to around 100. We also observe that SVD outperforms VSM method greatly if the dimension is adjusted to the proper value. Notice that the precision of SVD can reach 85% which is very high in text retrieval. This is because the number of documents in a single peer is small (i.e., around 200). Naturally, in a smaller data collection, relevant results are more likely to be searched and retrieved. This experiment confirms the superiority of SVD. Furthermore, notice that the VSM representation of a document is typically in tens of thousands of dimensions. However, the summary produced by SVD is typically in dimensions of less than two hundreds, which is in a different order of magnitude. It is obvious that both peer processing time and storage overhead will be reduced dramatically due to much smaller representation.

Next, we study the retrieval precision at the group level with the peer level summaries. We test if the correct peers that contain the relevant documents for the queries can be returned. Hence at this level, the precision is measured by the ratio of the number of relevant peers over the number of returned peers. Figure 9 illustrates the variation of the average precision as the number of dimension increases. Lastly, we repeat the experiment on the highest level of hierarchy to test if the correct groups that contain the relevant documents can be returned. At this level, its precision is measured by the ratio of the number of relevant peer groups over the number of returned peer groups. The result is shown in Figure 10. From Figure 9 and Figure 10, we can see that different dimensionality of summary may achieve different precision. The smallest value with highest precision is always chosen as the final dimensionality of summary at each level. Note that the dimension in these two figures are very small. This is because the reduced dimensionality of SVD is limited to the smaller one between the number of rows (or dimensionality of

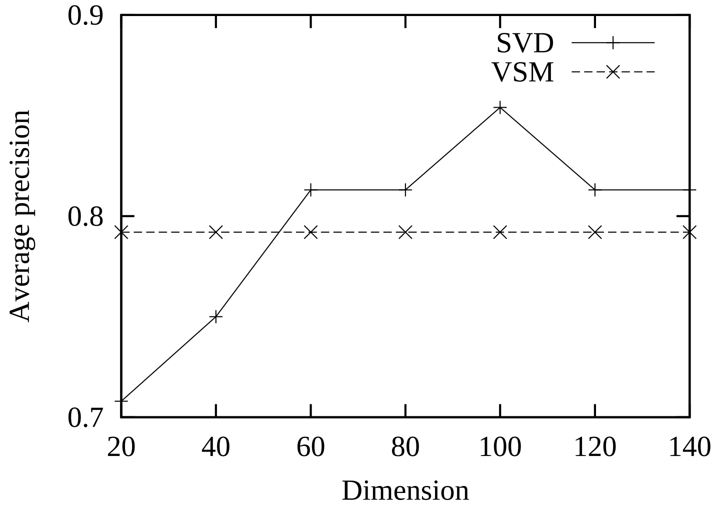

Fig. 8. Document Level Summary Precision.

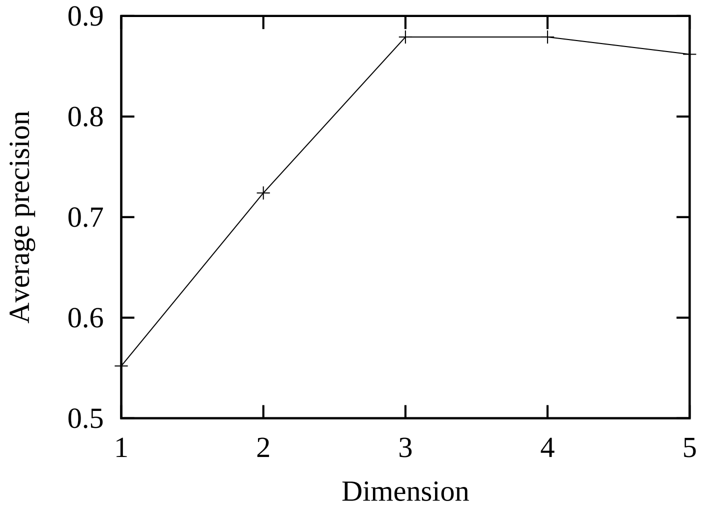

Fig. 9. Peer Level Summary Precision.

term dictionary) and number of columns (or number of peers) of the matrix. In our case, since the peer group size and group number are very small, the reduced dimension is relatively small. Nevertheless, we can still observe the effect of dimension reduction from the following figures. Naturally, summaries with smaller dimensionality at higher level will lead to faster peer locating.

2) Precision of the Whole System: In the above subsection, we have seen how the dimensionality of summary affects the precision at each individual level. After the dimensionality of summary at different levels has been determined, the hierarchical summary structure is constructed. In this experiment, we integrate the three levels and test the overall precision of the whole system. The precision is measured

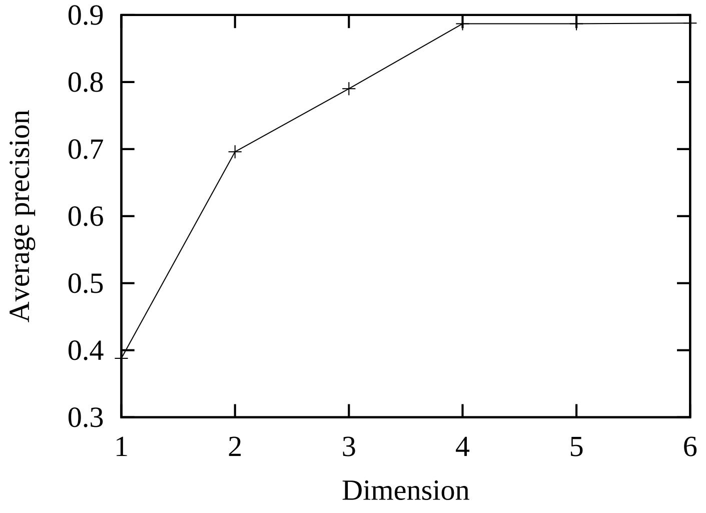

Fig. 10. Super Peer Level Summary Precision.

by the ratio of the number of relevant documents over the number of returned documents after the whole network has been searched. Obviously, the precision of the whole system is expected to be lower than the precision at documentation level since the precision is further reduced at higher levels.

Figure 11 compares the overall precision achieved by SVD and VSM in the system. We tested a pair of combinations for (K_{group}, K_{peer}, K_{doc}): (2,2,4) and (1,2,4). For combination (2,2,4), SVD outperforms VSM by 3% (16%-13%), or relatively 23% (3%/13%). And for combination (1,2,4), SVD outperforms VSM by 2.5% (28%-25.5%), or relatively 10% (2.5%/25.5%). Notice that the precision for combination (2,2,4) is much lower than combination (1,2,4). This is because the number of returned documents for (2,2,4) is 16 and only 8 for (1,2,4). This experiment shows that our hierarchical summary method achieves significant improvement comparing with the existing method. It proves that our method is much more effective and it is applicable for the document search in P2P systems. Users have the freedom to set the values of (K_{group}, K_{peer}, K_{doc}). To achieve higher precision of the whole system, users may set the K values to be smaller at levels with higher precisions. For example, in our real system, since the precisions of document level, peer level and super peer level increase, it is recommended to set K_{group} smaller than K_{peer}, which is in turn smaller than K_{doc}.

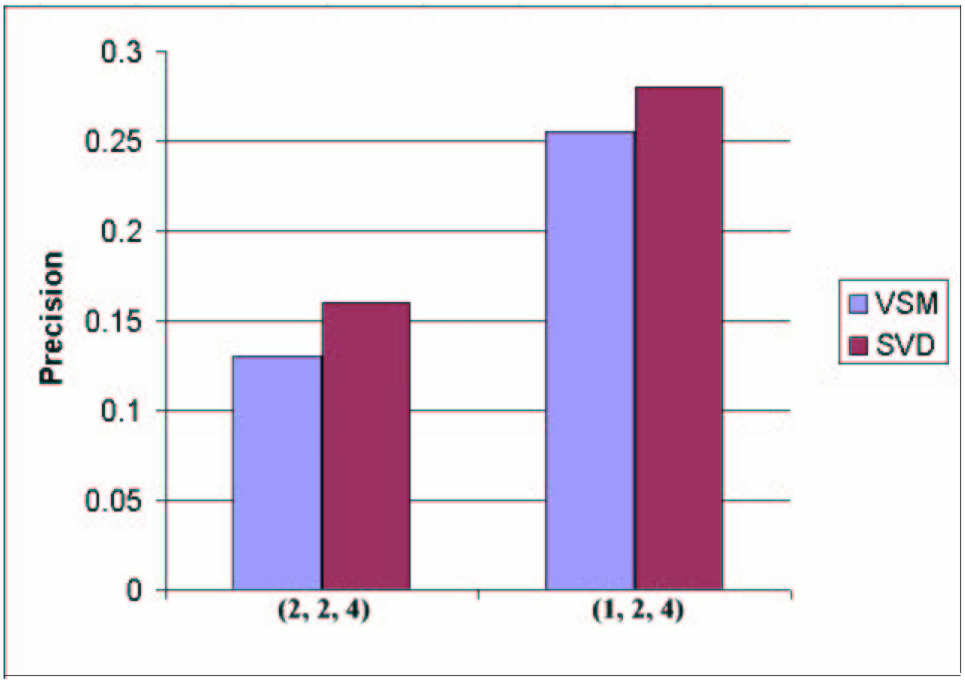

Fig. 11. Overall Hierarchical System Summary Precision.

C. Retrieval Efficiency

In this experiment, we simulate our system with 10,000 peers. A large set of synthetic documents were generated based on the distribution of terms in the real documents we used, with each peer having an average of 2000 documents. In our simulation, we only consider results from the first 1000 queries, though queries themselves are generated continuously and endlessly for better simulation.

We hope we could compare our system with other content-based P2P systems, but all these systems employed unstructured P2P architecture, which is certainly less efficient than our system, which is built on super-peer structure. Retrieval efficiency of our system is largely attributed to our novel summary hierarchy built on super-peer architecture. With indexes maintained in super peers, search becomes much more efficient than pure P2P network. Given an average number of 2,000 documents in a peer, only 20ms is needed for processing a query when VA-file is used, while about 50ms is needed when inverted file is used [4]. As this is expected, we put more focus on studying what factors are involved in a super-peer setting, which may potentially affect the retrieval efficiency.

Good indexing method and high network bandwidth are certainly beneficial for the retrieval efficiency.

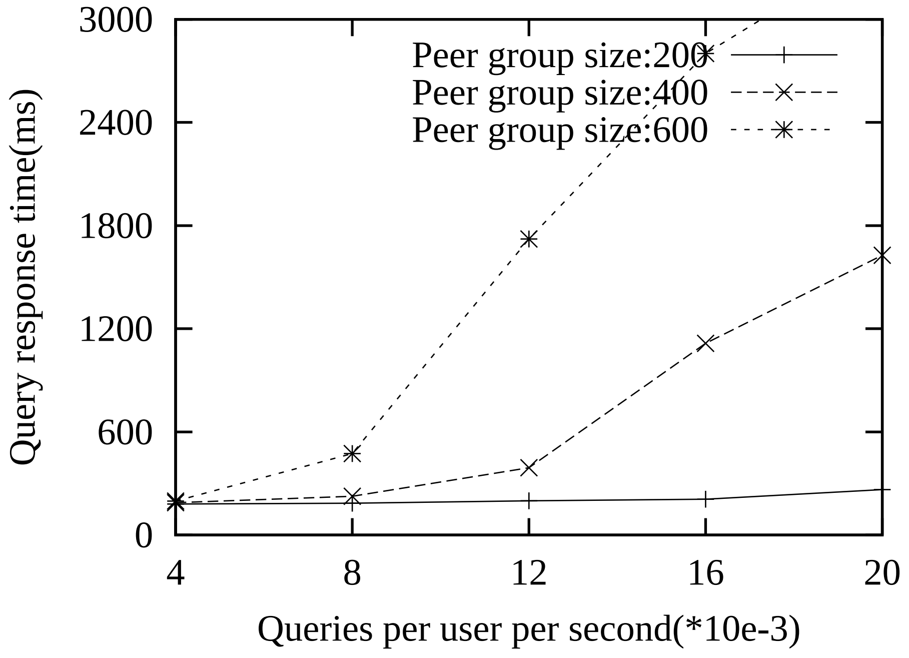

Fig. 12. Effect of Peer Group Size on Query Response Time.

Though network bandwidth is an uncertain factor in a real setting, which can be varied from several Kbps to hundreds of Mbps, for our simulation, we only use two kinds of bandwidth: LAN (10Mbps) and WAN (56Kbps). Since experiments results from LAN and WAN have the same trend, due to space limitations, we only present the results for WAN network.

Still, there are some other factors, whose effect may not be so apparent as the above, such as peer group size. To study all this in more detail, several experiments were conducted. First, we study the effect of peer group size on Query Response Time, given a certain query scheduling rate; second, the relationship between peer group size and the system load is thoroughly analyzed; and finally, we study the role of super peer capability in the retrieval efficiency, when the peer group size is increased (in our experiment, capability is modelled by the number of messages peer can process simultaneously and the number of network connections).

In Figure 12, each super peer's capability is 5 times than an ordinary peer's, which means the number of messages the super peer can process and the number of network connections are 5 times than an ordinary peer's. From the figure, we can see that as query rate increases, Query Response Time will increase correspondingly. This can be explained by the following: larger query rate results in less time interval for query scheduling, thus increasing larger possibility of query messages to compete for network and computing resources. If there are not enough resources, some messages must wait until other messages are processed first.

Meanwhile, we vary peer group size to study its effect on the Query Response Time. From experiments, we found that larger peer group size may result in worse Query Response Time. In Figure 12, three peer group size configurations are compared: *200*, *400* and *600*. Their differences in Query Response Time become more apparent as query rate increases: the Query Response Time for peer group size 600 is much longer than the one when the peer group size is 200. With larger peer group size, a super peer is more inclined to be overloaded for processing queries coming from its peer group. And higher query rates further make more queries generated and competed for super peer resources within a time interval, thus further increasing the delay.

System Load is also analyzed, especially, Load on super peers, with varying peer group size. Here, Load is measured by the following aspects:

- Total Messages Transmitted over the whole network (L1);
- Average Messages Received by a Super Peer(L2);
- Average Message Queue length of a Super Peer, measured in per 20ms(L3);

Among the above, the first one (L1) represents the overall network load, the other two (L2, L3) represent the load on a super peer, i.e., how many messages are received by a super peer, how many are still waiting for a super peer to process.

	size=200	size=400	size=600	size=800	size=1000
L1	227419	149028	123218	108821	99542
L2	575	850	1092	1309	1597
L3	0.015	0.038	0.071	0.116	0.225

TABLE IV
SYSTEM LOAD (THE CAPABILITY OF SUPER PEER IS SAME AS PEER'S).

	size=200	size=400	size=600	size=800	size=1000
L1	324110	323848	321272	274668	235062
L2	722	1419	2056	2436	2884
L3	5.517e-6	3.31e-5	8.113e-5	0.0003	0.0009

TABLE V

SYSTEM LOAD (THE CAPABILITY OF SUPER PEER IS IMPROVED BY 5).

	size=200	size=400	size=600	size=800	size=1000
L1	324273	324218	324167	324093	323745
L2	723	850	2070	2700	3500
L3	0	0	0	0	0

TABLE VI

SYSTEM LOAD (THE CAPABILITY OF SUPER PEER IS IMPROVED BY 10).

From Table IV, we find larger peer group size imposes more load on the super peer. This is expected as more messages are received, and more messages remained in the process queue. It is also observed that the total messages transmitted over the network is decreased. This can be explained by the following: In theory, varying peer group size should have no effect on total messages transmitted in the network, since the number of queries scheduled only depends on query scheduling rate, and, the path length of each query message are same. Therefore, if the simulation continues with no end, total messages $transmitted$ over the network for different peer group sizes should be same. However, our simulation ends when waiting times of the first 1000 queries are expired. When the simulation is terminated, there may be some messages still waiting for super peer's processing. The more load on the super peer, the more messages in the super peer's process queue, thus, the less total messages transmitted over the network when the simulation is terminated.

To offset the effect of larger peer group size, one way is to increase super peer capability, making more capable super peer cope with more load resulted from larger peer group size. We do two sets of experiments, first, each super peer's capability is improved by 5; second, each super peer's capability is improved by 10. Table V and Table VI are experiment results from these two sets respectively.

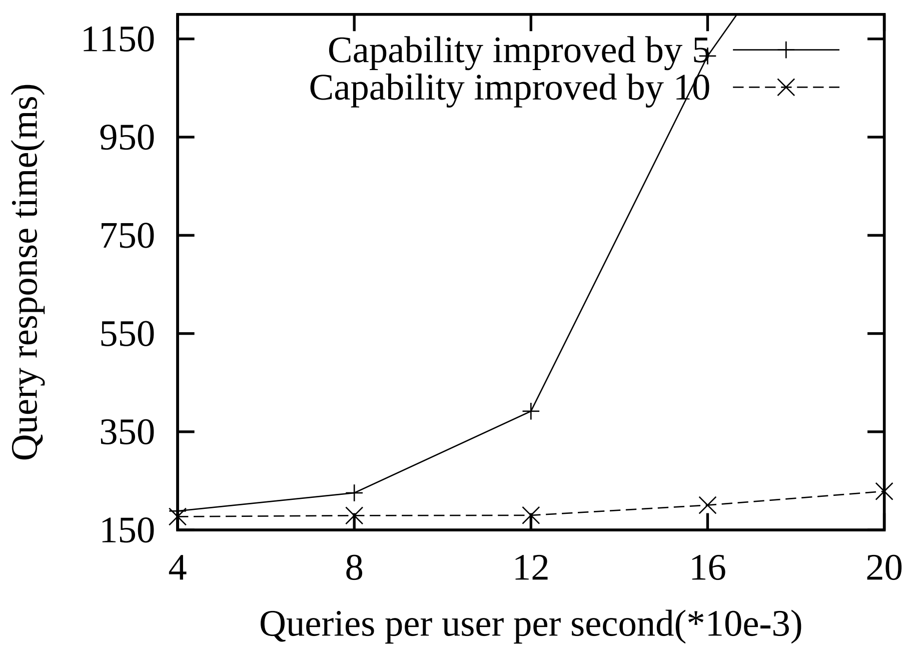

Fig. 13. The effect of super peer capability on search(peer group size=400).

By analyzing these two tables, it is easy to see, by improving the super peer's capability, fewer messages are left in the process queue. Also, we may expect the query response time to improve as super peer's capability improves, since more messages can be handled by more powerful super peers.

Figure 13 proves our guess, larger super peer's capability does offset the delay resulted from larger peer group size and higher query rate. By increasing average super-peer capability by 5 times to 10 times, query response time improves a lot.

From above experiments, we can see that, besides two factors we listed at the beginning of this section, three factors may affect retrieval efficiency, i.e., query rate, super peer capability, and peer group size. Query rate is usually user-decided. A super peer with more capability is always desirable, though it may not be much helpful for small load within the network. In our experiments, for a certain configuration, we found though the Query Response Time improves a lot when increasing super-peer capability by 10, there's little improvement when we increase super peer capability further. Much consideration needs to be given in selecting appropriate peer group size. Though experiments show larger peer group size is not beneficial for retrieval efficiency, this does not mean that a small peer group size is preferred. There is a tradeoff: Smaller peer group size results in more peer groups, thus more communication overhead when building super-level indexes.

D. Updating Effect

Next, we examine the updating issue on our hierarchical summary indexing method. As we have presented in Section V, the super peers use the parameter AIR to indicate the degree of information updating in the network. Only when AIR exceeds some pre-defined threshold, super peer will rebuild the index. We have studied the changes of precision and the AIR values when new peers join the network. We divided the documents of the MED dataset into 11 peers randomly. Using the 30 queries available with the MED dataset, we simulated the performance when peers join the network incrementally, and recorded the changes of average precision and AIR values. Figure 14 shows the changes of precision and AIR value when the ratio of the number of newly joined peers with the the number of indexed peers increases. We can see from Figure 14 that the precision drops as the AIR value increases. When the ratio of newly joined peers reaches 0.5, i.e., the AIR reaches 0.4, the precision still remains over 0.6. Hence if the AIR threshold is chosen properly, the precision can be kept in the acceptable range in the ad-hoc network environment while keeping the frequency of rebuilding index low.

E. Cost Reduction

In this experiment, we examine the flexibility of reducing the cost of summary building by SVD. We use the simple random sampling technique to see its effect on the efficiency and precision so as to achieve a good trade-off between both in P2P environment. We use two parameters: *relative precision* (precision with sampling over precision without sampling) and *relative cost* (time with sampling over time without sampling).

Figure 15 depicts the trends of relative precision over the changing of sampling rate on MED dataset. As we can see, the relative precision drops slowly as the sampling rate is reduced until sampling rate reaches 0.15. This suggests that applying SVD on 15% random samples of original datasets can achieve

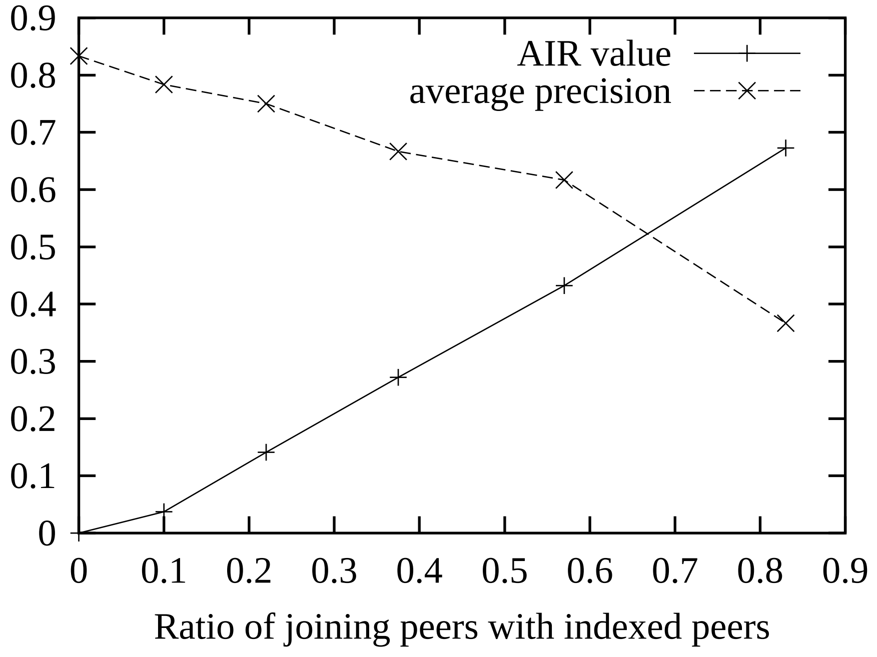

Fig. 14. Join Effect on Precision.

as high as 85% of original precision. Figure 16 depicts the trends of relative cost over the changing of sampling rate on the same dataset. We can see that the relative cost is almost linear to the sampling rate. When the sampling rate drops to 0.15, the time to produce the summary is almost 1/10 of original time. Recall that the time of building summary is $O(N * D^2)$ where N and D are the number of vector and the dimensionality of vector.

VIII. Conclusion

In this paper, we have examined the issues of supporting content-based searches in a distributed peer-to-peer information sharing system. We have proposed the first general and extensible hierarchical framework for summary building and indexing in P2P network. Based on this framework, we have presented an effective two-step summarization technique to transform large size representations of documents, peers, and super peers into small high-dimensional points, and extend known indexing technique to index transformed points at corresponding level for efficient peer and document search. A prototype and a simulated large-scale network have been designed to evaluate the system performance. Our experiments showed that such a hierarchical summary indexing structures can be easily adopted and our prototype system achieves

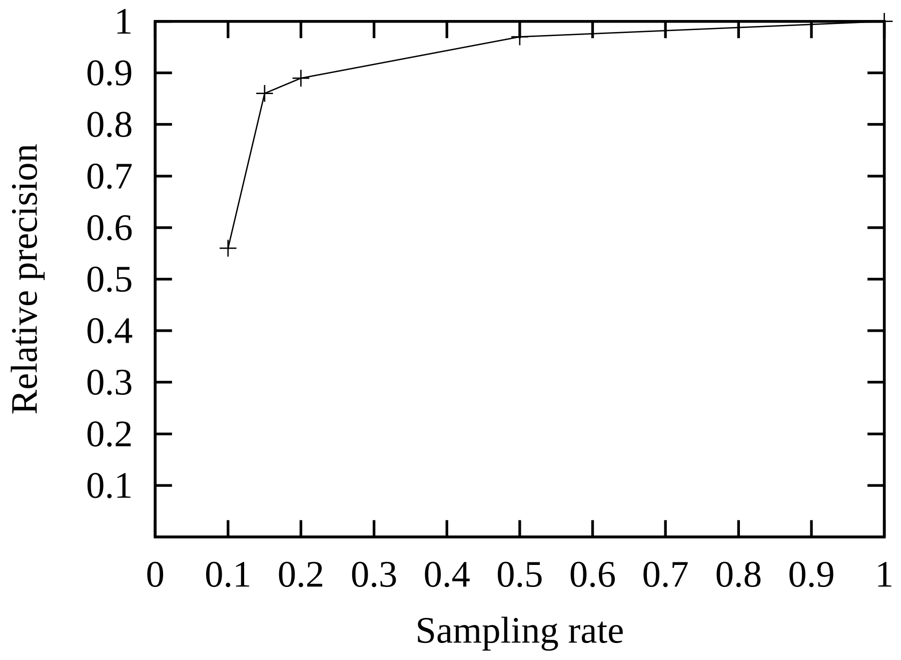

Fig. 15. Sampling Effect on Precision.

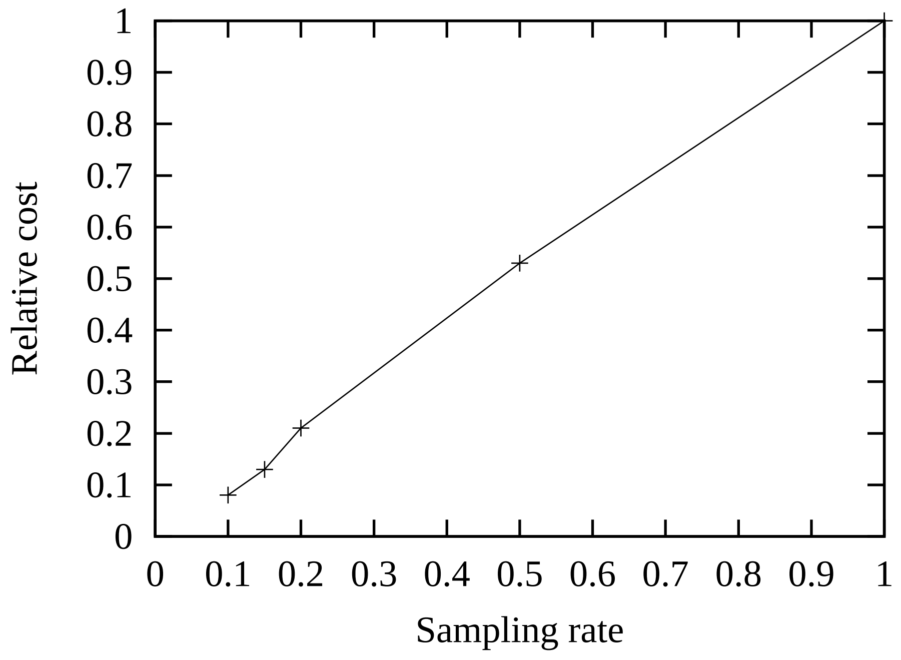

Fig. 16. Sampling Effect on Efficiency.

remarkable achievements.

REFERENCES

[1] S. Berchtold and D. A. Keim. Indexing high-dimensional spaces: Database support for next decade's applications. *ACM Computing Surveys*, 33(3):322–373, 2001.

[2] C. Buckley, A. Singhal, M. Mitra, and G. Salton. New retrieval approaches using smart. In *TREC 4.*, pages 25–48., 1995.

[3] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *28th Intl. Conf. on Distributed Computing Systems*, 2002.

[4] F. M. Cuenca-Acuna and T. D. Nguyen. Text-based content search and retrieval in ad hoc p2p communities. In *International Workshop on Peer-to-Peer Computing*, 2002.

[5] Freenet. *http://freenet.sourceforge.com/*.

[6] Gnutella. *http://gnutella.wego.com/*.

[7] Napster. *http://www.napster.com/*.

[8] SVD Package. *http://www.netlib.org/svdpack/*.

[9] C. Palmer and J. Steffan. Generating network topologies that obey power law. In *GLOBECOM*, 2000.

[10] C.H. Papadimitriou, H. Tamaki, P. Raghavan, and S. Vempala. Latent semantic indexing: A probabilistic analysis. In *PODS*, 1998.

[11] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *SIGCOMM*, 2001.

[12] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM*, 2001.

[13] P. Triantafillou, C. Xiruhaki, M. Koubarakis, and N. Ntarmos. Towards high performance peer-to-peer content and resource sharing systems. In *CIDR*, 2003.

[14] R. Weber, H. Schek, and S. Blott. A quantitative analysis and performance study for similarity search methods in high dimensional spaces. In *VLDB*, pages 194–205, 1998.

[15] S. K.M. Wong, W. Ziarko, V. V. Raghavan, and P. C.N. Wong. On modeling of information retrieval concepts in vector spaces. In *TODS*, 1987.

[16] B. Yang and H. Garcia-Molina. Comparing hybrid peer-to-peer systems. In *VLDB'2001*, 2001.

[17] B. Yang and H. Garcia-Molina. Improving efficiency of peer-to-peer search. In *28th Intl. Conf. on Distributed Computing Systems*, 2002.

[18] B. Yang and H. Garcia-Molina. Designing a super-peer network. In *ICDE*, 2003.