Dynamic Clustering-based Query Answering in Peer-to-Peer Systems

Weining Qian† Shuigeng Zhou Yi Ren Aoying Zhou

Department of Computer Science and Engineering

Fudan University

{wnqian, sgzhou, ayzhou}@fudan.edu.cn

Beng Chin Ooi Kian-Lee Tan

Department of Computer Science

National University of Singapore

{ooibc, tankl}@comp.nus.edu.sg

Abstract

P2P computing has been employing in more and more application domains as the technology becomes mature. One popular and successful application area is file sharing. However, current file sharing systems support only or mainly key-based exact matching (e.g., Chord [27], CAN [25]) and keyword-based searching (e.g., Napster, Gnutella) for files discovery and location, which is not enough to meet the requirements of more advanced applications such as information retrieval and data management. In this paper, we propose a new query answering model for P2P applications, which is termed as *clustering-based query answering* (CBQA). In our definition, CBQA will retrieve the data objects that are in the same cluster of the query from the global dataset distributed over peers of a P2P system. Generally, CBQA may obtain more correct answers than similarity based query can, which means higher recall may be achieved. To implement the new query model, we first present a framework that support clustering based query answering, including general algorithms, lemmas and system architecture. Then we give three concrete algorithms for different clustering criteria, namely k-nearest-neighbor, distance-based, and density-based clustering, along with detailed analyses and discussions. Finally, implementation issues, especially dynamic neighbors selection and caching techniques to enable the scalability of our method are addressed. Theoretical analysis and preliminary experiments show that our method can guarantee to find desirable objects in the interested cluster with modest bandwidth overhead.

Index Terms

P2P computing, clustering-based query answering

I. Introduction

Peer-to-peer (P2P) has become a new wave of innovative Internet-based computing technologies, and is expected to be an alternative of the traditional Client/Server model in many applications. In some domains, P2P-based applications have

†The author is partially supported by Microsoft Research Fellowship

been proved to be successful and are attracting more and more attention. These include file sharing [1], [2], [6], message exchanging [4], [5], scientific computing [7], bandwidth sharing and Web search [3], etc. In this paper, our interest is P2P systems for file sharing, information retrieval, and data management applications.

It is well-known that current file sharing systems support only or mainly key-based exact matching (e.g., Chord [27], CAN [25]) and keyword-based searching (e.g., Napster [6], Gnutella [2]) for files discovery and location, which is not enough to meet the requirements of more advanced applications such as information retrieval and data management. A natural advancement of key-based exact matching and keyword-based searching is similarity-based query (or simply similarity query). Similarity query is to retrieve the data objects that have similarity to the query not less than a pre-specified similarity threshold. Obviously, the number of returned query answers relies on the similarity threshold value: the larger the similarity threshold value is set, the more query answers can be obtained. Considering the decentralization nature (autonomy of peers, dynamism and ad hoc of network, and lack of global metadata) of P2P environment, it is inefficient to conduct similarity query in P2P systems (without special declaration, in this paper we refer to P2P systems as unstructured P2P systems such as Gnutella). Current solutions to this problem are to develop efficient routing algorithms [20] or build up distributed indices [28].

Generally, data distribution implies a certain cluster structure. The clustering task is to expose the underlying structure by using different algorithms. And data objects in the same cluster inherently share more similarity that these in different clusters. With these in mind, in this paper we propose a new query model for P2P application. We term the new query model *Clustering-Based Query Answering* (CBQA).

Given a query (it can be an example, or a set of keywords for document retrieval), we define CBQA as to retrieve the data objects that are in the same cluster of the query from the global dataset distributed over peers of a P2P system. Comparing with similarity query,

1) CBQA can obtain results that are more conform to the nature of queried data because all query answers it retrieves are in the same cluster of the query. However, similarity query may retrieve some objects that do not belong to the same of the query if the similarity threshold is too small; on the contrary, it may excludes some objects that belong to the same cluster of the query if the similarity threshold is too large. In Section 2, we will depict the cases above more clearly by example.
2) CBQA is more general. Considering that data clusters may in arbitrary shapes, which means query results of CBQA may distribute in the interested cluster region of possibly arbitrary shape; while results of similarity query are limited to the hypershperical region with the query as center and the similarity threshold as radius. Particularly, when the

shapes of all clusters of a given dataset is convex and the distances between different clusters are far enough, results of similarity query may be equal to that of CBQA if the similarity threshold is properly selected. Generally, CBQA may obtain more correct answers than similarity query can, which means higher recall may be achieved.

To implement the new query model, we first present a framework that support clustering based query answering, including general algorithms, lemmas and system architecture. Then we give three concrete algorithms for different clustering criteria, namely k-nearest-neighbor, distance-based, and density-based clustering, along with detailed analyses and discussions. Finally, implementation issues, especially dynamic neighbors selection and caching techniques to enable the scalability of our method are addressed. Theoretical analysis and preliminary experiments show that our method can guarantee to find desirable objects in the interested cluster with modest bandwidth overhead.

A. Our contribution

In this paper, we propose the clustering based query answering model and give concrete solution to implement this model in P2P environments. Following aspects distinguish our work from other research on file sharing and information retrieval in P2P systems.

1) A new query model, i.e., *clustering-based query* is proposed, which can be seen an advancement of traditional similarity based query or a kind of combination of clustering and similarity based query processing.
2) A framework for implementing clustering-based query in P2P environments is provided, which includes general algorithms, lemmas to guarantee the correctness and completeness of query results, and system architecture based on BestPeer [21]. The proposed framework is independent from concrete clustering criteria, so that variety of clustering algorithms can be employed in it.
3) Three concrete algorithms for different clustering criteria, namely k-nearest-neighbor, distance-based, and density-based clustering, are developed, based on the framework mentioned above. Furthermore, the properties of the algorithms are discussed, and the correctness of the algorithms is proved.
4) The computation and bandwidth cost for the algorithms above is analyzed, which shows that the cost is near optimal. It is also shown that the algorithms need much fewer resources than the naive method.
5) The implementation issues, especially dynamic neighbor selection and neighbor suggestion, data caching techniques, are discussed. With these techniques, our method can be scaled up to large-scale P2P applications.
6) Preliminary experiments validate the efficient and effectiveness of the proposed method.

To the best of our knowledge, this is the first attempt to propose the clustering based query model and implement the model in P2P environments. The techniques introduced in this paper may become the foundation of high-level P2P applications, such as information search and retrieval, data management data mining, and so on.

B. Paper organization

The rest of the paper is organized as follows. In Section 2, clustering-based query is proposed and defined, while the research motivation is discussed. After the introduction to the framework to implement clustering based query in P2P environment in Section 3, we give the algorithms for three different clustering criteria in Section 4. The analyses of algorithm complexity are also provided. In Section 5, implementation issues are addressed. In Section 6, the related work on P2P systems and clustering techniques are surveyed. Finally, the paper is concluded in Section 7.

II. MOTIVATION AND PROBLEM STATEMENT

Clustering-based query is to find the cluster with respect to a specific query object. We discuss the applications of clustering-based query first in this section. Then, the problem is defined formally.

A. Applications of clustering in P2P systems

Clustering-based query answering problem exists in many real-life applications which need to find *all* similar objects with respect to a query object in the same cluster. Some of such typical applications in P2P environments are listed below.

Health care data management

Health care data are usually distributed on different computers. A doctor may have his stable group of patients in a period of time. The data, including the patient's information, symptom, and case history, are stored within the doctor's own computer with frequent updates. However, to analyze the affect of new medicines or to find case histories of particular diseases, data on different doctor's computers may be involved. The analysis task may be represented by queries like this, '*Retrieve all the case histories similar to a patient with symptom on blood pressure and cardiogram*', or '*Retrieve all the patients who have similar reaction to one kind of medicine to one specific patient's reaction*'. The target of the search or mining is usually of one particular object, e.g. a patient of certain symptom, or a kind of medicine, etc., which is the query object. However, the search condition of the related data cannot be defined in advance, and doctors may attend or leave the system while the data on each computer is dynamically changing. It is obvious that clustering-in-advance is not able to solve the problem. CBQA is a suitable model for such problems.

Personal digital collection sharing

Personal digital collection includes text, image, or video. Although traditional *Query-by-Example* (QBE) may find some related information, it cannot find *all* information of a query object. Consider a virtual community, in which each peer's host is a serious researcher. The researchers may report their new experimental result or the observation he or she collected. He may want to find *all* similar data about one outlier in his or her own dataset. Another example is a virtual community constructed by serious artists, each collects the art interested. One artist may want to find information about *all* other art with similar characteristics, e.g. color, or layout. In such P2P systems, the query condition will be figured out while the mining task is issued, and cannot be determined in advance. CBQA is a right way to abstract such problems.

Genome database mining

The discovery of new proteins need complex analysis on genomic data. Data mining techniques, namely classification and sequential pattern discovery, are widely used in this domain. Clustering can be used as preprocessing for these tasks. There exists several genomic databases, e.g. GenBank, SWISS-PROT and EMBL, and new data are produced everyday by research institutes in different places all over the world. These databases can be collaborated with each other and form a P2P system. To analyze a new protein may involve clustering on several databases. As in health care data management and personal digital collection sharing example, the clustering cannot be processed in advance, for the fast changing of the whole dataset. Furthermore, to collect sufficient data is important for further study. CBQA is an appropriate model for such tasks.

Three possible applications of clustering-based queries are discussed above. It is shown that they share the common characteristics in that,

- Data are distributed on different peers, while the whole dataset is large;
- *All* data similar to a query object should be retrieved, which means the data objects in the same cluster with the query object are interested;
- The clustering could not be processed in advance, because of the absence of in-advance clustering condition or the frequently changing dataset.

There are many other applications satisfied with these conditions, such as data caching, digital library, etc. They are all potential applications of clustering-based query.

We show the characteristics of CBQA, which is different with traditional similarity query answering, in the following example.

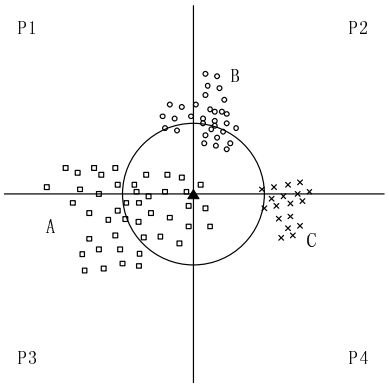

Fig. 1. The difference between clustering-based query and similarity query

Example 1: In Figure 1, there are four peers with data objects from three clusters. The object denoted by ▲ at the cross of the lines is the query. Traditional clustering techniques can be applied on the whole dataset, which need the transfer of all the data objects from all peers to one site. As for similarity queries, they can only find the data objects in the circle, as it is shown in the figure. Thus, many objects in cluster B would be sent before all the data objects in cluster A is identified.

The example shows that, similarity queries are not suitable for high-level analysis tasks, while traditional clustering methods yield high network transfer cost.

B. Clustering-based query

For formal definition of the answer with respect to a query of CBQA, we should define the *global clustering* result as the baseline. Global clustering can be viewed as the *virtual* process executes on the whole dataset, that is the intention of the user. It can adopt some widely accepted clustering criteria, which are introduced in the next section.

Definition 1—Global Clustering: Given a dataset DB, and a clustering condition c, the **global clustering result w.r.t. c** is a set of sets C_i, for $i = 1, ..., n$, denoted by $\{C_i\}_c$, satisfied that $\bigcup_i C_i = DB$, and $C_i \cap C_j = \emptyset$ for $i \neq j$.

Thus, clustering-based query can be defined according to global clustering.

Definition 2—Clustering-based Query: Given a dataset DB, a global clustering condition c, and a query object o, the **o-based clustering result w.r.t. c** is a set C^o, satisfied that $C^o \in \{C_i\}_c$, and $o \in C^o$.

Since we usually cannot obtain the result of global clustering in P2P systems, only the clustering-based query result C'^o w.r.t certain clustering condition c' can be found. If $C'^o = C^o$, we call that clustering condition c and c' are **consistent on o**. Thus, the clustering-based query for o is *to find C'^o that is consistent to pre-defined clustering condition on o.*

It will be shown in the next section that, under specific conditions, we can obtain the CBQA result consistent with the global clustering condition without retrieve the whole dataset.

III. Framework for CBQA in P2P environments

In this section, the framework for finding the cluster according to the query is introduced. The properties of the framework are discussed, which become the base of the analysis in next section. Then, the system architecture is presented, while the requirements for platform support are discussed.

A. Overview

The procedures for CBQA are shown in Algorithm 1 and 2. The main procedure obtains the cluster C in which query object o falls in, based on the data it has already collected. The $Query()$ procedure tries to search the local database of other peers those data objects that potentially belong to the final cluster. The $mainCBQA()$ procedure initializes

Algorithm 1 $mainCBQA$

Input: query object o, peer p
Output: cluster C^o

1: $C \leftarrow \{o\}$; {data in cluster}
2: $D \leftarrow \emptyset$; {data not in cluster}
3: $E \leftarrow \emptyset$; {new clustering result}
4: $T \leftarrow LDB$; {all data collected on peer p, it includes the local database and the objects transferred from other peers}
5: $P \leftarrow$ {all p's neighbors}; {all neighbors of peer p}
6: **while** $C \neq E$ **do**
7: **for** each $q \in P$ **do**
8: $T \leftarrow T \cup q.Query(QueryFunction())$; {query the neighbors}
9: **end for**
10: $C \leftarrow E$;
11: $E \leftarrow Clustering(o, T)$; {get the new cluster based on the new T}
12: $D \leftarrow T - E$;
13: $P \leftarrow \emptyset$; {empty the neighbor list}
14: **if** $(D - LDB) \neq \emptyset$ **then**
15: $P \leftarrow$ {peer $r | r$ only has answers in E}; {need more data objects from these peers}
16: **end if**
17: **if** $P = \emptyset$ **then**
18: $P \leftarrow$ {all p's neighbors}; {each neighbor sends objects for one more time}
19: **end if**
20: **end while**
21: **return** C;

the cluster C that contains o (line 1). The data objects already collected, that do not belong to the cluster, are stored in D. All data objects that have been collected are stored in T, while E is used to store the new clustering result after the communication with neighbors, which are stored in P. If the new clustering result is not the same with the old one

Algorithm 2 $Query$ procedure on peers

Input: $QueryFunction()$
Output: result set R

1: $DB \leftarrow \{a | a \in LDB, a$ has not been sent$\}$; $\{DB$ contains the candidate data objects to be queried$\}$
2: $R \leftarrow QueryFunction(DB)$; {search the local database by using function that received}
3: **for** each object $a \in R$ **do**
4: $\quad Label(a,' has\_been\_sent')$; {each data object should be sent at most once for one query task}
5: **end for**
6: **return** R;

(line 6), the procedure retrieves new data objects from its neighbors based on $QueryFunction()$ (loop from line 7 to 9). $Clustering()$ procedure is executed on the new local dataset T, so that the clustering result is updated (line 11). Only peers with all answers in the cluster are queried further (line 15). But, if such peer does not exist, the peer cannot judge whether its neighbors have returned all related objects. Therefore, it needs to query all its neighbors (line 18).

When a peer receives a query from another peer, it looks up its local database for those data objects that have not been returned before (line 1 of $Query()$). $QueryFunction()$, which is sent with the query, is evaluated on those data objects. The data objects satisfied the condition are returned to the query peer (line 6), and labelled $'has\_been\_sent'$ in local site (line 4).

B. Properties of CBQA framework

We call the procedure $Query()$ is *order-consistent* if the data belong to the final cluster are returned before other data objects are returned. For the procedure $Clustering()$, if $(C^o \cap T_1) \subset (C^o \cap T_2)$ implies that $Clustering(o, T_1) \subseteq Clustering(o, T_2)$, we say that it satisfies *monotone*. It means that, the more data a peer collects, the larger the cluster would be. If $T_1 \subset T_2$ implies that, $(T_1 - Clustering(o, T_1)) \subseteq (T_2 - Clustering(o, T_2))$, we say that it satisfies *anti-monotone*. It means that, if an object is not in the cluster in T, it would not be put back to the cluster after more data objects are collected. For $Clustering()$ procedure, we call it is *locality-preservable* if it satisfies that $C^o \subseteq Clustering(o, T)$ and $C^o \subseteq T$ implies $Clustering(o, T) = C^o$.

Lemma 1: If procedure $Query()$ is order-consistent, while $Clustering()$ satisfies monotone or anti-monotone, and is locality-preservable, Algorithm 1 returns C^o.

Proof: First we prove that when $Clustering()$ is anti-monotone, the conclusion holds.

The excluded objects will not be put into the cluster. Therefore, we must prove that all objects in cluster will be collected. If a peer has already returned objects don't belong to the cluster, it will not contain any other data objects belong to the cluster, since $Query()$ is order-consistent and $Clustering()$ is anti-monotone. If the result returned by

one peer all belong to the cluster, the peer must have already returned all its data, otherwise the algorithm will not stop (line 15 in Algorithm 1). Therefore, all peers must have already sent all its data objects belonging to the cluster. Since $Clustering()$ is locality- preservable, the conclusion holds.

Then, assume that $Clustering()$ is monotone. All data objects put in cluster will not be dropped later. The size of the cluster increases until all data objects in cluster are included. Since $Query()$ is order-consistent, it will find all those objects, or the process will not stop. By the locality-preservable property of $Clustering()$, the conclusion holds.

End of proof

Lemma 2: If in Algorithm 1 and 2, $Query()$ is order-consistent, while $Clustering()$ is locality-preservable, then

1) The total network transfer is less than $\|C^o\| + \|P\| \times \|R\|$ objects, in which $\|C^o\|$ is the number of objects in C^o, $\|P\|$ is the number of peers that are involved in the clustering, and $\|R\|$ is $\max\{$ objects returned by one peer once$\}$;
2) The loop from line 6 to 20 in Algorithm 1 is executed at most $\frac{\|C^o\|}{\|R\|} + 1$ times.

Proof: Since $Clustering()$ is locality-preservable, while $Query()$ is order-consistent, each peer at most returns data objects not in cluster once. Therefore, the conclusion holds.

End of proof

Lemma 2 means that the wasted bandwidth, which is used for transfer of data objects that are not in the cluster, is $\|P\| \times \|R\|$. The smaller $\|R\|$ is, the smaller the network transfer is. On the other hand, the number of times for communication between the peer, who issues clustering task, and its neighbors, increases along with the decreasing of $\|R\|$. Establishing or stopping a communication needs additional overhead. Therefore, an appropriate $\|R\|$ should be chosen.

C. System architecture

As it is shown in the procedure $Query()$, a peer can send one function $QueryFunction()$ to another peer. The function will be evaluated at a remote peer so that interested data can be found. Agent-based technology is employed to implement such functions. Each $QueryFunction()$ is encapsulated in an agent. Note that there are several such functions for supporting different clustering criteria. The agents are management by an *agent management module*.

The system is implemented based on BestPeer platform [21]. BestPeer is a general purpose P2P platform which allows a peer to send an agent to remote peers for executing complex computation tasks. The agent-based mechanism enables our method to handle different clustering criteria under one uniform framework. Although the system is built on BestPeer, our method can also be applied over other platforms if a peer can execute computation tasks on remote peers.

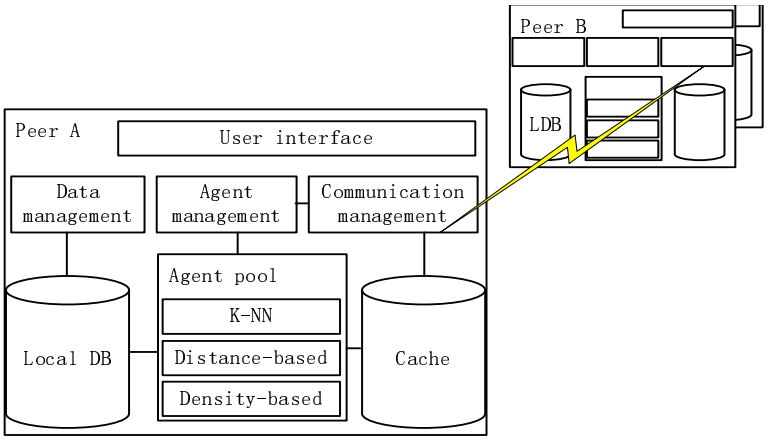

Fig. 2. The architecture of the system

The architecture of our system is shown in Figure 2. On each peer, a *local database* is maintained, while the space of cache for data transferred from other peers is preserved. An *agent pool* stores several agents, each for one clustering criteria. Data and agents are managed by *data management module* and *agent management module* respectively. Another important component is the *communication management module*, which is in charge of the agent sending/receiving, data exchanging, and other communication tasks.

When a clustering task is issued by the user, the peer starts a new clustering process in local. The process searches in the local database, while choosing an appropriate agent from the agent pool, and send it to neighbors. After several times communication, it would have sufficient data objects in local database, for local data, or in cache, for data transferred from other peers. Then, the process obtains the right result and send it back to user.

When a peer receives an agent of a query task, it first passes the agent to agent management module. The agent is executed through the data management module. The result is sent back to the peer who initiates the agent, through communication module.

An agent is killed when the clustering task is finished. Therefore, although the $Query()$ procedure may be executed several times, each peer will receive the corresponding agent at most once. Furthermore, note that the $Query()$ function should have *memory* to remember the history of data objects in one clustering task. The agent is in charge of this task.

IV. Clustering for different criteria

In this section, algorithms for three different clustering criteria, namely k-nearest-neighbor (k-NN), distance-based, and density-based clustering, are introduced. We prove that all these algorithms can find consistent result according to

(centralized) global clustering algorithms. The network transfer and computation cost of the algorithms are analyzed.

It is shown that k-NN and distance-based clustering criteria is suitable for building low-cost algorithms under the framework introduced in last section. However, since density-based clustering criteria does not satisfy the order-consistent property, applying the framework directly may lost data objects belonging to the cluster. An algorithm based on distance-based clustering algorithm is introduced. The analysis show that it can find the cluster with quite low cost.

A. K-nearest-neighbor search

K-NN is the process to find the k nearest data objects to the given query object. K-NN is widely used in similarity search, proximity search, pattern recognition and information retrieval.

We apply the functions in Algorithm 3 and 4 to the framework introduced in last section. Here, α is a parameter whose domain is $[0, 1]$, which denotes the percentage of k that should be retrieved from each neighbor. $QueryFunction()$ searches in the local database for αk nearest data objects to o, and returns them to the peer who issue the task. $Clustering()$ finds in the DB, which is the set of data objects that have been collected, the k nearest neighbors to the query object. The result is the cluster.

Algorithm 3 $QueryFunction$ for k-NN clustering

Input: query object o, k, α, DB
Output: result set R

1: $R \leftarrow \{$the closest αk objects in DB to $o\}$;
2: **return** R;

Algorithm 4 $Clustering$ for k-NN clustering

Input: query object o, k, DB
Output: cluster C

1: $C \leftarrow \{$the closest k objects in DB to $o\}$;
2: **return** C;

Observation 1: By applying $QueryFunction()$ of Algorithm 3, The $Query()$ is order-consistent. The $Clustering()$ in Algorithm 4 is anti-monotone and locality-preservable.

It is obvious that on one peer, the objects close to the query object are returned before those objects that are farther away. Furthermore, if a data object has been found to be farther than other k objects, it will not become k-NN later. Therefore, $Clustering()$ is anti-monotone. Meanwhile, the k-NN in global must be the k-NN in the subset of the database, if they are included. Thus, by using Lemma 1, following lemma holds for our k-NN algorithm.

Lemma 3: The k-NN algorithm introduced above returns all and exact k-nearest-neighbors of o.

TABLE I
COST FOR k-NN CLUSTERING

Items	Cost	Remarks
Network transfer	$k + \alpha km$	α is the parameter for balancing between times of communication and the wasted bandwidth, m is the number of peers that return result
Times for communication	$1 + \frac{1}{\alpha}$	α is described above
Computation complexity	$O(mk \log k)$	m is described before

The cluster contains exactly k data objects, while each time a neighbor sends αk objects. By applying Lemma 2, the network transfer cost of the algorithm is analyzed as it is shown in Table I. Since the maintenance of the cluster after the communication is $O(m\alpha k \log k)$[1], the overall computation cost is $O(mk \log k)$, as it is also shown in Table I.

B. Distance-based clustering

Distance-based clustering is widely used in data mining [14]. Traditional global distance-based clustering use the number of clusters K or the minimum inter-cluster distance d as threshold [24]. We prefer the later for CBQA.

The clustering condition, which is similar to single-link algorithms, is defined as follows:

1) $o \in C^o$;
2) If $\min\{distance(p, q) | q \in C^o\} \leq d$, then $p \in C'$;
3) If $p \in C'$, then $p \in C^o$;
4) If $distance(o, p) \leq \max\{distance(o, q) | q \in C'\}$, then $p \in C^o$.

The first condition means that the query object itself belongs to the cluster. The second and third condition means that if a data object is *single-link-connected* to the query object, it belongs to the cluster. The fourth condition means that if a data object is closer to the query object than any single-link-connected data object, it belongs to the cluster. Note that the data objects satisfied the fourth condition would not be expanded to link more data objects into the cluster.

The procedures in Algorithm 5 and 6 are applied in the framework. As in k-NN clustering, $QueryFunction()$ looks in local database for several nearest objects to the query object, except that the number of objects is defined by parameter k, which is determined by users. $Clustering()$ searches in the set of objects collected based on the second and third clustering condition (loop from line 6 to 11), and the fourth (loop from line 15 to 19).

Observation 2: By applying $QueryFunction()$ of Algorithm 5, $Query()$ is order-consistent. $Clustering()$ in Algorithm 6 is monotone and locality-preservable.

[1] Assume that each peer has index on database at local site.

Algorithm 5 $QueryFunction$ for distance-based clustering

Input: query object o, k, DB
Output: result set R

1: $R \leftarrow$ {the closest k objects in DB to o};
2: **return** R;

Algorithm 6 $Clustering$ for distance-based clustering

Input: query object o, threshold d, DB
Output: cluster C

1: $C \leftarrow \{o\}$;
2: $F \leftarrow \{o\}$;
3: $T \leftarrow DB$; {all data objects}
4: **while** $F \neq \emptyset$ **do**
5: $F \leftarrow \emptyset$;
6: **for** each object $t \in T$ **do**
7: **if** $distance(t, C) < d$ **then**
8: $F \leftarrow F \cup \{t\}$; {$t$ should be added into the cluster, since it satisfies the third condition}
9: $T \leftarrow T - \{t\}$;
10: **end if**
11: **end for**
12: $C \leftarrow C \cup F$; {the cluster grows in batch}
13: **end while**
14: $max\_dis \leftarrow \max\{distance(o, s) | s \in C\}$;
15: **for** each object $t \in T$ **do**
16: **if** $distance(o, t) \leq max\_dis$ **then**
17: $C \leftarrow C \cup \{t\}$; {add t into the cluster since it satisfies the fourth condition}
18: **end if**
19: **end for**
20: **return** C;

The properties of $Query()$ are the same for k-NN and distance-based clustering, since their $QueryFunction()$ are essentially the same. Once an object is found to be within the range of d to query object o, it would not be dropped. This results in the conclusion that monotone holds for $Clustering()$. Furthermore, an object who is out of the range will never be wrongly put into the cluster. Therefore, locality-preservable holds for $Clustering()$. By Lemma 1, it is easy to get that:

Lemma 4: The distance-based clustering algorithm introduced above returns all and exact data objects in C^o with respect to distance-based clustering criteria.

Assume the cluster contains N data objects. The number of objects a neighbor sends to the peer is k, as it is defined in Algorithm 5. By applying Lemma 2, the network transfer cost of the algorithm is analyzed as it is shown in Table II. The cost for maintenance of the cluster is linear to the size of the updated dataset. Therefore, the computation cost is $O(N)$.

TABLE II
COST FOR DISTANCE-BASED CLUSTERING

Items	Cost	Remarks
Network transfer	$N + k \times m$	N is the size of C^o, k is the number of objects returned each time, m is the number of peers who return result
Times for communication	$1 + \frac{N}{k}$	N and k are described above
Computation complexity	$O(N)$	N is described above

C. Density-based clustering

As long as distance-based clustering, density-based clustering is a kind of popular clustering criteria. The cell-based criteria, which is one of the density-based criteria [24], is adopted in our clustering-based query. The clustering condition is as follows:

1) All data objects fall in o's cell belong to C^o;
2) If cell c's density is larger than a threshold d, and c is the neighbor of a cell whose objects all falls in C^o, then all data objects in c belong to C^o.

Different with k-NN or distance-based clustering, it is hard to construct $QueryFunction()$ and $Clustering()$ directly from the clustering condition that satisfies the order-consistent, monotone/anti-monotone and locality-preservable respectively. We adopt another approach that employs the framework to solve the problem of density-based clustering.

First, the clustering condition similar to distance-based CBQA, that is easy to be implemented under the framework, is defined.

1) $c_o \in C'^o$, in which c_o is the cell o falls in;
2) If $\min\{distance(c, q)|q \in C'^o$ and $density(c) > d\}$, then $c \in C'$;
3) If $c \in C'$, then $c \in C'^o$;
4) If $distance(c_o, p) \leq \max\{distance(c_o, q)|q \in C'\}$, then $p \in C'^o$.

Here, a cell $c \in C$ means all data objects fall in c belong to cluster C. The distance between two cells means the distance of the vectors for coordinates of two cells. It is obvious that distance-based clustering algorithm can be applied to solve the clustering problem defined by above condition. Thus, the density-based clustering algorithm is shown in Algorithm 7. The cells are clustered by using the new clustering condition in line 2, by using distance-based clustering criteria, while line 7 in Algorithm 6 is replaced by:

if $distance(t, C) \leq 1$ and $density(t) > d$ **then**

After the data objects that may fall in the cluster are collected, $DensityClustering()$ clusters them based on density-

based clustering criteria. This is the centralized process. We omit the details here.

Algorithm 7 $DensityClustering$ procedure

Input: query object o, DB
Output: cluster C

1: $c_o \leftarrow$ the cell contains o;
2: $C_c \leftarrow mainClustering(c_o, this)$; {clustering using the new condition over P2P system}
3: $C \leftarrow LocalDensityClustering(o, C_c)$; {clustering using the old condition in local}
4: **return** C;

Lemma 5: Algorithm 7 returns all and exact data objects in C^o with respect to density-based clustering criteria.

Proof: First, after the modification, the lemma for distance-based clustering (Lemma 4 still holds, since $Clustering()$ is still monotone and locality-preservable, while $Query()$ is not affected.

Therefore, all data objects satisfied the modified condition have been collected by line 2. The cells that are connected to the query object are still connected by the query object, otherwise, a cell like this must be connected by at least one of the other cells, that is connected to query object, containing objects that are not collected. This is not true, as it has already been proved. So the lemma holds.

End of proof

From the analysis to distance-based clustering, the network transfer and times for communication of density-based clustering is linear to the size of the final cluster.

D. Experimental result for density-based CBQA

We simulate the experiments of density-based clustering. One thousand peers are built, and 100,000 data objects in two-dimensional Euclidian space are assigned to the peers randomly. The data objects are divided into two groups, in which one group is the cluster we want to find, that is density separated with other data. We define the separation of the cluster to other data as $sep(C^o) = \frac{\|C^o\|}{\|C'\|}$. Each time, one object in the cluster is chosen randomly as query object, and the cost for network transfer is recorded. For each separation value and k value pair, ten query objects are chosen to get the average result. The relationships between separation and wasted data transfer for different k is shown in Figure 3. It is shown that the more clear the cluster is separated from other data, the less wasted bandwidth is spent. Furthermore, it is shown that for different k, which is the number of cells that should be sent in every time of the communication, the cost for network transfer is different. The larger k is, the more the network transfer is wasted. The result is consistent with our analysis before.

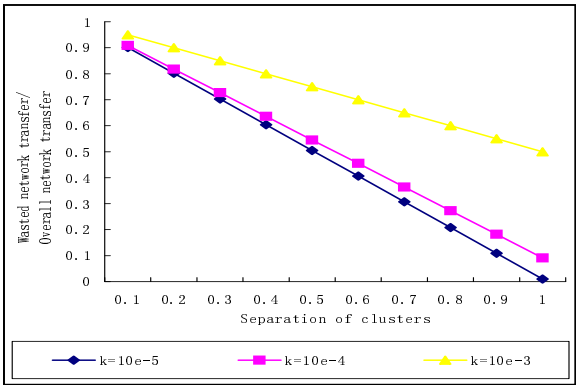

Fig. 3. Network transfer of density-based CBQA

V. Implementation Issues

The algorithms introduced above is developed under the ideal condition that the resource of each peer is not limited, so that each peer can maintain the connection to all other peers easily. However, this is usually not true. The scalability of the peers is one basic request for the P2P systems. Furthermore, a user may not be patient enough until all result data objects are returned by all the peers. In this section, the techniques for handling these problems are discussed.

A. Handling a giant number of peers

The algorithms introduced in previous sections assume that the peer, who initiates the query, can maintain all peers, which may contain the result objects, as neighbors. However this is hardly true in most applications, since the system may involve more than thousands of peers. Keeping the connection to all the possible neighbors costs a lot of resource of a peer.

For handling the real-life applications, we adopt a *dynamic neighbor suggestion* mechanism. In line 15 of Algorithm 1, for each neighbor who did not return any answer belonging to the cluster in the last few iterations, it must be swapped out after suggesting one of its neighbors.

Heuristic 1: The suggestion is based on following strategies:

1) Each peer sends the mean of the objects in its local database to its neighbor, when they are establishing connection;
2) The peer suggests the neighbor whose mean is the nearest one to the query object.

Note that the request peer does not drop the data objects returned by the swapped-out neighbors. Therefore, the swapped-out peer still has chance to be chosen back (line 15 of Algorithm 1).

Furthermore, it is usually impossible to wait until all peers are traversed, especially in a wide-area-network or Internet

environment. The peer, who issues the query, sets up a time-out threshold, so that the query can be answered within an acceptable time.

B. Dynamic initial neighborhood selection

A self-organizing approach is adopted for choosing the initial peers. The peer firstly send the query to the current neighbors. Instead of processing the query right away, these peers searches in its own neighbors. A peer would suggest the peer p whose mean m minimizes $\|DB_p\| \times d(m, o)$. The suggested peer will replace the one who suggested it. The process is executed several iterations. In each iteration, a *better* peer is suggested. By using this approach, the initial neighbors are determined.

This method is based on the idea that the later the initial neighbors are swapped out, the better the performance is. Note that each peer has already known the mean and size of its neighbors, this method would not increase the overhead much, since only when a neighbor is suggested, additional communication is needed. The overhead is linear to the number of neighbors times the iteration that are needed. Usually, both of them are small.

C. Caching

In real applications, a peer may issue different queries at different time. Therefore, it may have already collected some result objects in the previous queries. In implementation, each peer has a cache, in which the previous returned result can be stored. The cache size can be customized by the user, and may be different between different peers.

VI. Related work

The P2P model is firstly introduced in application domains [1], [2], [4], [6]. Currently, it has been successfully applied in instant message services [4] or file sharing [1], [2].

Researchers with different backgrounds study the P2P systems from different perspective. The routing and resource location problems has been studied in networking community, as they are reported in [26], [27], [25], etc. Some researchers has analyzed the architecture, protocol of P2P systems [30], [21]. As it is stated before, the BestPeer framework introduced in [21] is employed in our system.

Much work has been done on providing services in P2P systems. Two collections have been published reporting some of the results, as they are in [23], [17]. The state of the art in P2P-based information system includes searching [31], storage [26], retrieval [20], query processing [22], and schema mediation [15]. Some advanced applications include message exchanging [5], computing [7], Web caching [29] and online analytical processing [18].

Although some issues of data mining in P2P systems have been mentioned in [17], to the best of our knowledge, clustering in P2P systems has not been researched before.

BestPeer [21] is a general purpose framework designed for advanced P2P-based applications. Several systems based on BestPeer have already been proved to be successful in experimental environment [20], [29], [18], [22]. As in those systems, BestPeer is the supporting platform of our clustering method. However, as it is discussed in Section 3, our method can be applied on any P2P platform supporting computing ability sharing on peers. Furthermore, the analysis and conclusions in this paper is BestPeer independent.

Clustering is one of the basic techniques in data mining [19] and pattern recognition [9]. Much work has been done on centralized site, from different perspective, e.g. effectiveness, efficiency, robustness, and applications, as they are surveyed in [19], [13], [24]. Some famous algorithms include optimization-based algorithm k-means [19], distance-based agglomerate algorithm CURE [14], and density-based algorithm DBSCAN [11]. All these methods assume that the dataset is available and accessible with quite low cost. Although some incremental algorithms, e.g. IDBSCAN [10], and sampling based algorithms, e.g. DataBubble [8], have been developed, they still assume the data are accessible with low cost. Furthermore, these algorithms only fit for one specific clustering criteria, i.e. density-based clustering, which limits their applications in P2P environments. On the other side, all current clustering research focus on centralized clustering. Many results reported are useful for improvement of clustering on one peer. Building multi-dimensional index on each peer, for example, can improve the performance on each peer for *Query()* function. Since the limit of space, these issues will not be discussed in this paper.

Cluster validation [16] is a technique for finding interested clusters from clustering results. Cluster validation shares the same idea with CBQA in that, only some of the clusters are interested by users, so that only these parts should be retrieved. However, clustering-based query differs with cluster validation in three aspects. First, cluster validation is a post-clustering processing, while clustering-based query itself is a variation of clustering. Second, the research on clustering-based query is driven by the request of saving bandwidth while preserving cluster quality. The cluster validation is studied to decrease the affect of noises on clustering result. The last but not the least, cluster validation methods may access the whole dataset, which is almost impossible in P2P systems.

Top/bottom-N queries in distributed databases are studied in [12]. The problem we studied in this paper is different in that the query condition can be extended to distance-based or density-based clustering criteria. Furthermore, the scalability problem met in P2P systems are discussed.

VII. Conclusion

A novel query model, namely, *clustering-based query answering*, is studied intensively in this paper. The research is driven by the applictions, e.g. file sharing and information retrieval, in peer-to-peer environments. CBQA meets the requests of these applications in that, 1) the model generalizes several traditional clustering criteria, so that it can be used to solve more analysis problems than tranditional similarity queries. 2) Only a small portion of dataset need to be transferred over network under certain conditions, while the query result is proved to be consistent with the global clustering. In other words, the query result is guaranteed, while the cost is low. 3) The query model can be supported by current P2P platforms easily.

The algorithm framework for CBQA is introduced and the properties are studied. Furthermore, we discussed the application of the model on three different, but widely employed, clustering criteria, i.e. k-NN, distance-based, and density-based clustering, in details. The theoretical analysis and preliminary experiments show the effectiveness and efficiency of the algorithms.

The implementaion details are discussed. We discussed the neighborhood suggestion, initial neighborhood selection, and caching mechanism adopted in our system. These techniques enable our method to be applied in real-life P2P systems.

Though current method is optimized for network transfer, it doesn't consider the index existing on peers. The utilization of information on other peers remains as our future work.

Acknowledgement

The authors would like to thank Wee Siong Ng, for providing us the source code of BestPeer.

References

[1] Freenet homepage. *http://freenet.sourceforge.com/*.

[2] Gnutella developement homepage. *http://gnutella.wego.com/*.

[3] Hyperbee homepage. *http://www.hyperbee.com/*.

[4] Icq homepage. *http://www.icq.com/*.

[5] Jabber homepage. *http://www.jabber.org/*.

[6] Napster homepage. *http://www.napster.com/*.

[7] Seti@home homepage. *http://setiathome.ssl.berkeley.edu/*.

[8] M. M. Breunig, H.-P. Kriegel, P. Kroger, and J. Sander. Data bubbles: Quality preserving performance boosting for hierarchical clustering. In *Proceedings of ACM SIGMOD 2001 International Conference on Management of Data (SIGMOD'2001)*, 2001.

[9] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, 2nd edition, 2000.

[10] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, and X. Xu. Incremental clustering for mining in a data warehousing environment. In *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB'98)*, pages 323–333, 1998.

[11] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of 2nd Int'l Conf. on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press, 1996.

[12] R. Fagin. Combining fuzzy information from multiple systems. In *Proc. ACM SIGMOD/SIGACT Conf. on Princ. of Database Syst. (PODS)*, 1996.

[13] D. P. Fasulo. An analysis of recent work on clustering algorithms. Technical report, Department of Computer Science and Engineering, University of Washington, 1999.

[14] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. In *Proc. of ACM SIGMOD Int'l Conf. on Management of Data*, pages 73–84. ACM Press, 1998.

[15] A. Halevy, Z. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. In *To appear in Proceedings of IEEE Conference on Data Engineering (ICDE'2003)*, 2003.

[16] Z. Huang and T. Lin. A visual method of cluster validation with fastmap. In *Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data-mining (PAKDD'2000)*, pages 153–164, 2000.

[17] F. Kaashoek and A. Powstron, editors. *Electronic Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS'2002)*. Available at: http://www.cs.rice.edu/Conferences/IPTPS02/, 2002.

[18] P. Kalnis, W. S. Ng, B. C. Ooi, D. Papadias, and K.-L. Tan. An adaptive peer-to-peer network for distributed caching of olap results. In *Proceedings of ACM SIGMOD 2002 International Conference on Management of Data (SIGMOD'2002)*, 2002.

[19] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[20] B. Ling, Z. Lu, W. S. Ng, B. C. Ooi, K.-L. Tan, and A. Zhou. A content-based resource location mechanism in peeris. In *Proceedings of IEEE Workshop on Web Information Systems and Engineering (WISE'2002)*. IEEE Press, 2002.

[21] W. S. Ng, B. C. Ooi, and K.-L. Tan. Bestpeer: A self-configurable peer-to-peer system. In *Proceedings of IEEE Conference on Data Engineering (ICDE'2001)*. IEEE Press, 2002.

[22] W. S. Ng, B. C. Ooi, K.-L. Tan, and A. Zhou. Peerdb: A p2p-based system for distributed data sharing. In *To appear in Proceedings of IEEE Conference on Data Engineering (ICDE'2003)*. IEEE Press, 2003.

[23] A. Oram and et. al, editors. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, 2001.

[24] W. Qian and A. Zhou. Analyzing popular clustering algorithms from different viewpoints. *To appear in Journal of Software*, 2002.

[25] S. Ratnasamy, P. Francis, K. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM 2001*, 2001.

[26] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 2001.

[27] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM 2001*, 2001.

[28] Q. Sun and H. Garcia-Molina. Partial lookup services. Technical report, Stanford University, 2003.

[29] X. Wang, W. S. Ng, B. C. Ooi, K.-L. Tan, and A. Zhou. Buddyweb: A p2p-based collaborative web caching system. In *Proceedings of Peer-to-Peer Computing Workshop (Networking 2002)*. IEEE Press, 2002.

[30] B. Yang and H. Garcia-Molina. Comparing hybrid peer-to-peer systems. In *Proceedings of the 27th International Conference on Very Large Databases (VLDB'2001)*, 2001.

[31] B. Yang and H. Garcia-Molina. Efficient search in peer-to-peer networks. In *Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS'2001)*, 2001.