Supporting Similarity Range Queries Efficiently by Using Reference Points in Structured P2P Overlays

Guanling Lee, Yi-Chun Chen, and Chung Chi Lee

Department of Computer Science and Information Engineering
National Dong Hwa University, Hualien, Taiwan, R.O.C
guanling@mail.ndhu.edu.tw

Abstract. In recent years, the research issues in peer to peer (P2P) systems have been discussed widely. In a P2P system, the role of each node is the same, and the nodes simultaneously function as both clients and servers to the other nodes on the network. Many studies have been proposed for solving different problems to improve the performance of P2P systems. To solve file availability and network flow problems, a method named distributed hash tables (DHT) has been proposed. However, these DHT-based systems are not able to support efficient queries such as similarity queries, range queries, and skyline queries.

In this paper, a novel method for supporting similarity searches in a structured P2P system is proposed. Compared to other existing works, our approach shows great improvement in precision and guarantees the file availability. The experimental results show the effectiveness of our approach.

Keywords: Peer to Peer, Similarity Search, Dimension Reduction, iDistance.

1 Introduction and Related Work

Structured P2P systems, such as CAN [6] and Chord [7], utilize a Distributed Hash Table (DHT) to direct searches to specific node(s) holding the requested data. This architecture mainly concentrates on the key values lookup to improve response efficiency and to guarantee file availability. Since keyword searches are not sufficient, how to support a general query in DHT-based systems has been widely discussed [4][9][10]. In [8] the problem of similarity discovery in a P2P system is discussed and a method called pSearch is proposed. It uses a peer Vector Space Model and peer Latent Semantic Indexing (LSI) to improve the traditional Vector Space Model. [1] proposed a scalable and distributed access structure for similarity searches in metric spaces. All the works discussed above proposed a certain mechanism to reduce the document dimension for supporting similarity queries in a structured P2P overlay. However, the *recall* of the query cannot be guaranteed. *Precision* and *recall* are two key statistics regarding the system's returned results for a query, and are usually used to measure the effectiveness of an information retrieval system. Precision is defined as the proportion of retrieved documents that are relevant and recall is defined as the proportion of relevant documents that are retrieved. In general, there is an inverse relationship between precision and recall.

R.-S. Chang et al. (Eds.): *Advances in Intelligent Systems & Applications*, SIST 20, pp. 645–652.
DOI: 10.1007/ 978-3-642-35452-6_65

To solve the problem of recall, the M-Chord is proposed in [5]. The structure takes advantage of the idea of a vector index method iDistance[2] in order to transform the issue of a similarity search into the problem of an interval search in one dimension. However, it used only one reference point to calculate the response data which results in poor precision.

In this paper, the problem of how to support a similarity query in a structured P2P overlay is discussed. In our approach, by using the idea of iDistance, *m* reference points are selected to reduce a high dimensional document vector into an *m* dimensional vector. Additionally, the document is published in an *m* dimension CAN by mapping the *m* dimensional vector into a specific point in CAN. In query processing, the user's query will be calculated with reference points, which will get *m* bounds in every dimension. Those bounds will be used to fetch a set of object in CAN. The advantages of our work are that the recall and precision of a query can be guaranteed.

The remainder of this paper is organized as follows: The problem definition is described in section 2. Section 3 presents the main idea of our approach. Experimental results and analysis are discussed in Section 4. Section 5 summarizes our work.

2 Problem Definition and Preliminaries

In the similarity search system, each object can be represented by a *k*-element vector and in the vector, each dimension is standing for a term, which is related to the object. The similarity between the objects can be represented by the corresponding vectors. There are several similarity measures in the traditional database. In this paper, Euclidean distance is used to evaluate similarity between objects. The Euclidean distance between objects $X=(x_1, x_2,\ldots, x_k)$ and $Y=(y_1, y_2,\ldots, y_k)$, in Euclidean space, is defined as: $\sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\ldots+(x_k-y_k)^2}$. In a similarity discovery system, the range query is defined as Query(q, r) and it will retrieve all objects within distance r to q. Express as $\{x \in objects : dist(q,x) \le r\}$. In general, the number of dimensions is between 80,000 and 110,000. Even if LSI is used to reduce dimensionality, k is still between 50 and 350. To further reduce the dimensionality, the idea of iDistance is proposed. The advantage of iDistance is that the recall of search results can be guaranteed. However, the precision in the system is not good enough. That is, too many unrelated objects are retrieved. To solve the problem, in this paper, a document is mapped into a multi-dimension structure by applying a similar idea of iDistance.

In iDistance, the data space is partitioned into n clusters. For each cluster C_i, the centroid of C_i, said p_i, is selected as its reference point. Each data object is assigned a one-dimensional iDistance value according to the distance between itself and the reference object of the cluster it belongs to. By using a constant c to separate an individual cluster, the iDistance value for an object $x \in C_i$ is $iDist(x) = dist(p_i, x) + i * c$.

Expecting that c is large enough, all objects in cluster i are mapped to the interval $[i*c, (i+1)*c]$. According to the basic concept of range query, for Query(q, r), only the clusters whose coverage overlap the query range may contain the satisfied objects. That is, only the cluster C_i which satisfies the inequality $dist(p_i, q) - r \le r_i$, where r_i is the radius of C_i, should be checked. Moreover, because iDistance maps each object into a one dimensional space, the search range of the satisfied cluster C_i can be summarized as [MAX($i*c$, $dist(q,p_i)$-r+$i*c$), MIN(dist(q,p_i)+r+$i*c$, r_i+$i*c$)].

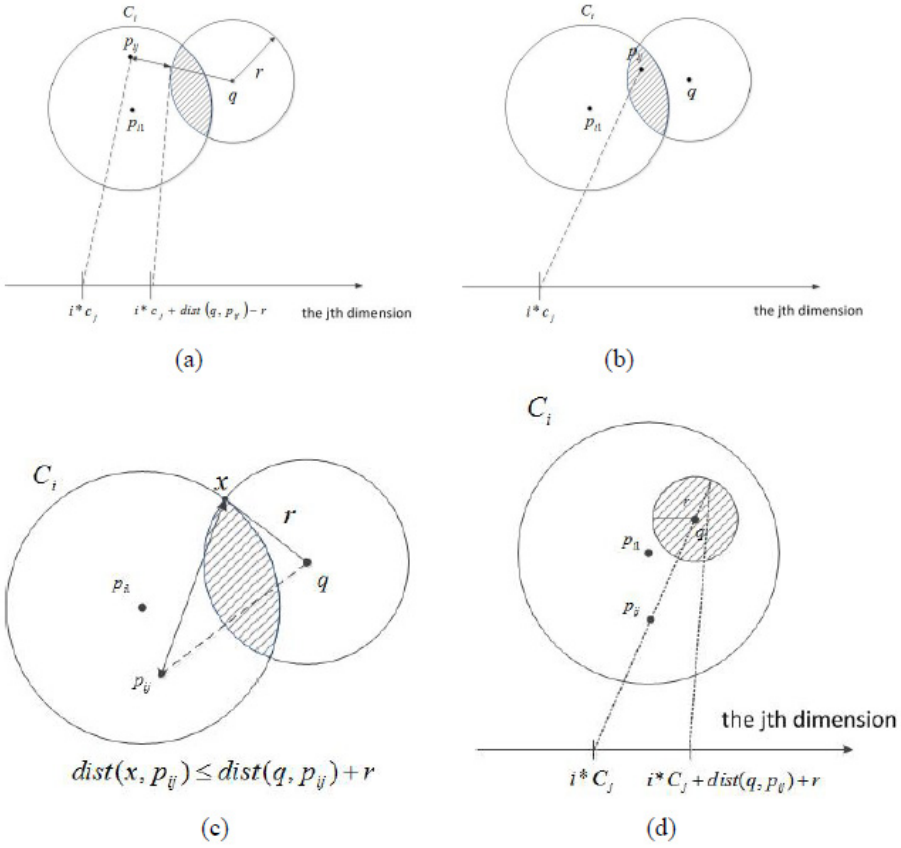

Fig. 1. Search bounds of reference point p_{ij}

Our system is based on CAN [6], which is designed to store and retrieve individual data objects and support exact match queries. To support a similarity search, in our approach the idea of iDistance is extended to map the document into CAN. Moreover,

to increase the precision, for each C_i, m reference points $\{p_{i1}, p_{i2},\ldots, p_{im}\}$, where p_{i1} is centroid of C_i, are selected in this approach. For an object x belonging to cluster C_i, the m-dimension vector $< dist(x, p_{i1}) + i * c_1, dist(x, p_{i2}) + i * c_2, ..., dist(x, p_{im}) + i * c_m >$ can be used to represent x in an m dimensional space. In the equation, c_j is a constant used to separate the range covered by the clusters in dimension j. And after the transformation, x can be assigned into m-dimension CAN easily.

To process range query Query(q,r), only the cluster C_i which satisfies the inequality $dist(p_i, q) - r \le r_i$, where r_i is the radius of C_i, should be checked. The lower bound of the search range is the minimum distance from reference point p_{ij} to the overlap region which is $dist(q, p_{ij})$-r+i*c_i (Fig. 1(a)) or i*c_j (Fig. 1(b)). The upper bound of the search range is the longest distance from p_{ij} to the overlap region. Refer to Fig. 1(d), when query range is fully covered by the cluster C_i, the longest distance can be computed as $dist(p_{ij}, p_{i1}) + r_i + i * c_j$. Otherwise, refer to Fig. 1(c), the longest distance is bounded by $dist(q, p_{ij}) + r + i * c_j$. Summarizing the above discussions, the search range for C_i in dimension j, i.e., the dimension corresponding to p_{ij}, can be computed as follows:

$$[MAX(i * c_j, dist(q, p_{ij}) - r + i * c_j), MIN(dist(q, p_{ij}) + r + i * c_j, dist(p_{ij}, p_{i1}) + r_i + i * c_j)] \quad (1)$$

3 Our Approach

3.1 File Publishing

Initially, a distributed cluster method proposed in [3] is used to partition the objects into clusters. After getting the clusters, m reference points will be generated for each cluster. A reference point p_{ij} denotes the jth reference point in cluster i. Moreover, p_{i1} is the centroid of C_i. By using the equation proposed in section 2, a high dimensional object x can be transformed into an m dimensional vector, that is, $< dist(x, p_{i1}) + i * c_1, dist(x, p_{i2}) + i * c_2, ..., dist(x, p_{im}) + i * c_m >$. With the m dimensional vector, object x can be published in an m dimension CAN by using the publish function supported in CAN directly.

3.2 Range Query Processing

To process range query Query(q,r), we first find out the clusters whose coverage overlap the query range. That is, the cluster C_i which satisfies the inequality $dist(p_{i1}, q) - r \le r_i$ is a candidate cluster. And then, for each candidate cluster C_i, equation (1) proposed in section 2 is calculated to get m search bounds. By searching the objects within the search bounds in the corresponding dimension in CAN, the objects satisfy Query(q,r) are obtained. The detailed algorithms are shown in Fig. 2 and 3.

```
Algorithm: Range Query(q,r)
Input: query point q
     query range r
Output: result S
1: S={}
2: For each cluster C_i
3:  If (dist(p_i1,q)-r≦r_i)  // C_i overlaps Query(q,r)
4:    S←S ∪ GetClusterCandidate(C_i,q,r)
5:  End If
6: End For
7: Return S
```

Fig. 2. Range query

```
Algorithm GetClusterCandidate(C_i,q,r)
Input: cluster C_i
     query point q
     query range r
Output:result S
1: S={}
2: For each p_ij  // reference points of C_i
// calculate the search bound
3:  L_j←MAX(i * c, dist(q, p_ij) − r + i * c_i)
4:  U_j←MIN(dist(q, p_ij) + r + i * c, dist(p_ij, p_i1) + r_i + i * c_i)
5: End For
6:S←Search the range <L_1,U_1>, <L_2,U_2>,…,<L_m,U_m> in CAN
7: Return S
```

Fig. 3. GetClusterCandidate

3.3 Reference Points Generation

The precision of the result is truly affected by the reference points. Therefore, how to generate m reference points is an important issue. In this approach, we propose three methods to generate reference points. Their efficiency will be discussed in section 4.

Random selection: Select m-1 objects in cluster i randomly to be the reference points $p_{i2},\ldots, p_{im}$.

Cluster selection: When m is 2, the one reference point is the centroid of the cluster, and the other is selected randomly. When $m > 2$, the objects in the cluster are further partitioned into m-1 subclusters. The centroids of the subclusters are selected to be reference points.

Coverage selection: When m is 2, the one reference point is the centroid of the cluster, and the other is selected randomly. When $m > 2$, the cluster is partitioned into m-1 equal coverage space areas. And the centers of the m-1 areas are selected to be reference points.

4 Experimental Results

4.1 Simulation Setup

All programs were written in Java and run on a PC with 3.0G Pentium 4 processor and 1G memory. The objects are generated synthetically with normal distribution. The function which is generally used to measure precision in a traditional IR system is shown below. Moreover, because our approach can achieve a 100% recall, we do not show the result here. Table 1 lists the parameters setting of our simulation.

$$\text{Precision}=\frac{\text{Number of items retrieved that are relevant}}{\text{Total number of file retrieved}} \tag{2}$$

Table 1. Parameter settings

	default	range
Number of Documents	100,000	-
Number of Peers in CAN	2,000	-
Number of reference points	4	2~8
Reference points generation methods	Random selection	Random selection Cluster selection Coverage selection

4.2 Experimental Results

Fig. 4 shows the average hop counts for processing range query with different number of reference points. In CAN, average hop-count decreases as dimension increases. Because the routing protocol is based on CAN, Fig. 4 shows the similar result.

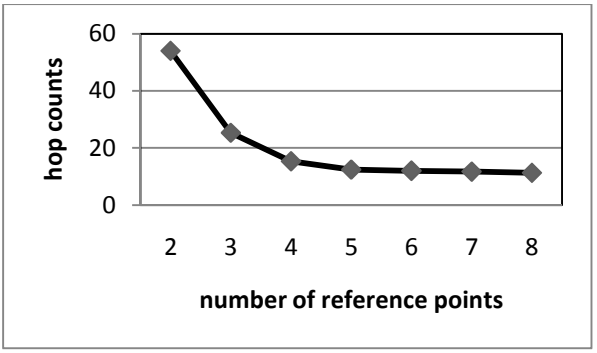

Fig. 4. Hop counts with different dimensions

Fig. 5 shows the precision with different number of reference points. As shown in the result, the precision of random selection and cluster selection methods increase as the number of reference point increases. And they always outperform M-chord. Moreover, random selection method performs almost the same as cluster selection method when the number of reference points larger than 5. This is because increase the number of reference points, the probability that the selected points cover the cluster increase. The precision of coverage selection method is very poor. The reason is that the distribution of objects in a cluster may not be uniform. Therefore, use the centers of the equal partition areas to be the reference points may lead to poor discrimination of the objects in the same cluster.

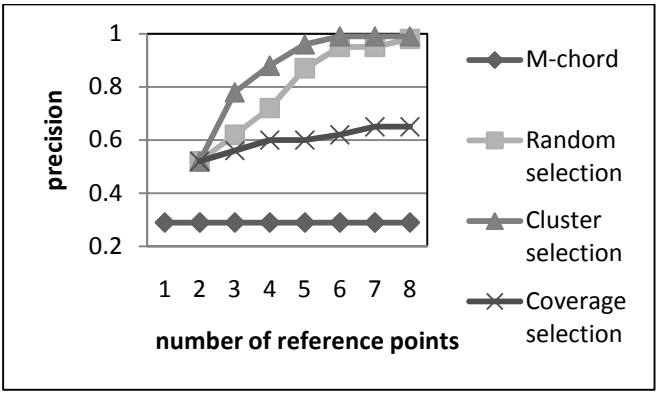

Fig. 5. Precision with different number of reference points

5 Conclusions

In this paper, the problem of how to support a similarity range query method in structured P2P overlays is discussed. In our approach, by using the idea of iDistance, m reference points are selected to reduce a high dimensional vector into an m dimensional vector which can be located in CAN. To process a range query, m search bounds are calculated according to m reference points. By using the bounds, the satisfied objects can be retrieved in CAN efficiently. In the simulation, a set of evaluations is performed to show the benefits of our approach. In comparison with previous approaches, our approach has a great improvement in precision and guarantees file availability.

References

1. Falchi, F., Gennaro, C., Zezula, P.: A Content–Addressable Network for Similarity Search in Metric Spaces. In: Moro, G., Bergamaschi, S., Joseph, S., Morin, J.-H., Ouksel, A.M. (eds.) DBISP2P 2005 and DBISP2P 2006. LNCS, vol. 4125, pp. 98–110. Springer, Heidelberg (2007)
2. Jagadish, H.V., Ooi, B.C., Tan, K.-L., Yu, C., Zhang, R.: iDistance: An adaptive B+-tree based indexing method for nearest neighbor search. ACM Transactions on Database System (TODS 2005) 30, 364–397 (2005)
3. Li, M., Lee, G., Lee, W.-C., Sivasubramaniam, A.: PENS: An Algorithm for Density-Based Clustering in Peer-to-Peer Systems. In: Proceedings of INFOSCALE 2006. IEEE Computer Society, Hong Kong (2006)
4. Mass, Y., Sagiv, Y., Shmueli-Scheuer, M.: KMV-peer: a robust and adaptive peer-selection algorithm. In: Proceedings of the 4th ACM International Conference on Web Search and Data Mining, Hong Kong, China, pp. 157–166 (2011)
5. Novak, D., Zezula, P.: M-Chord: A Scalable Distributed Similarity Search Structure. In: Proceedings of the 1st International Conference on Scalable Information Systems, New York, USA, pp. 1–10 (May 2006)
6. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A Scalable Content-Addressable Network. In: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, San Diego USA, pp. 161–172 (August 2001)
7. Stoica, I., Morris, R., Karger, D., Kaashoek, M., Balakrishnan, H.: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In: Proceedings of the ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, San Diego, USA, pp. 149–160 (2001)
8. Tang, C., Xu, Z., Mahalingam, M.: pSearch: Information Retrieval in Structured Overlays. ACM SIGCOMM Computer Communication Review 33, 89–94 (2003)
9. Tang, Y., Xu, J., Zhou, S., Lee, W.C.: A Lightweight Multidimensional Index for Complex Queries over DHTs. IEEE Transactions on Parallel and Distributed Systems 22(12), 2046–2054 (2011)
10. Tang, Y., Zhou, S., Xu, J.: Light: A Query-Efficient Yet Low-Maintenance Indexing Scheme over Dhts. IEEE Transactions on Knowledge Data Engineering 22(1), 59–75 (2010)