Dynamic P2P Indexing and Search based on Compact Clustering

Mauricio Marin
Yahoo! Research Latin America
Santiago, Chile
mmarin@yahoo-inc.com

Veronica Gil-Costa
DCC, University of San Luis
San Luis, Argentina
gvcosta@unsl.edu.ar

Cecilia Hernandez
DCC, University of Chile,
Santiago, Chile
chernand@inf.udec.cl

***Abstract*—We propose a strategy to perform query processing on P2P similarity search systems based on peers and super-peers. We show that by approximating global but resumed information about the indexed data in each peer, the average amount of computation and communication performed to solve range queries can be significantly reduced as compared to alternative state of the art strategies based on local indexing at peer level. We illustrate our technique by using an indexing method based on compact clustering.**

***Keywords*- Peer to peer networks; Metric-Space Databases.**

I. INTRODUCTION

Similarity search over a collection of metric-space database objects distributed on a large and dynamic set of small computers forming a Peer-to-Peer (P2P) network has been widely studied in recent years. Currently we have efficient solutions for structured networks like those based on the general purpose CAN [22] and Chord [24] protocols [12], [13], [21], [23], [26]. In these cases the assumption is that neither particular peers are more important than others by executing, for instance, coordination roles, nor they have more computing capabilities than other peers in the network. This represents the fully distributed case in the spectrum of feasible P2P systems for metric-space databases. Another way to achieve this case is by defining a specific purpose network topology by, for instance, mapping a data structure and using the IP addresses of the peer computers as pointers [1]–[4], [8], [9], [15]–[18], [25]. In order to achieve efficient performance during query processing, the metrics to be optimized are the cost of communication and the number of peers involved in the process which, for a fully distributed P2P system, can be significantly large.

Super-peer systems [11], [27], [28] are believed to represent a good tradeoff between centralized and distributed architectures. They are also considered a reasonable tradeoff between unstructured and structured P2P networks. In this case the network is seen as a collection of stable (usually powerful) peers called *super-peers* to which normal peers can connect and initiate queries. The most recent work on this kind of P2P systems (and as such the state of the art work for this paper) is the *SimPeer* system proposed in [11].

SimPeer uses the iDistance indexing strategy proposed in [14] to support range search upon a hierarchical unstructured P2P network composed of super-peers and peers. The routing indexes strategy proposed in [10] is adapted and used to route queries among neighboring super-peers. The peers are assumed to maintain and index their own data. The super-peers know a limited number of neighboring super-peers and index information of the peers connected to them. They also index information from their neighboring super-peers. This defines a three level hierarchy where the standard K-means clustering algorithm is used to index data in the hierarchy. The resulting centers from the clustering applied at each peer are indexed in the respective super-peer using also K-means clustering. These, at super-peer level, are called hyper-clusters. Finally, statistical data and hyper-centers from neighboring super-peers are also clustered in a second index stored in each super-peer which is used as a routing index. The scheme allows super-peers to send queries to both the most relevant peers and super-peers during query processing. The clusters are accessed during query processing by using the iDistance index [14] which is intended to provide fast access to the K-means clusters.

To make the point of this paper, we put the above description of SimPeer in generic terms. The special objects (centers) resulting from the indexing process in each peer are communicated to their respective super-peers, where they are further indexed. A query first scans the super-peer index and the result is a sort of plan that tells the super-peer to which of its peers the query should be sent and, very importantly, in which sections of each peer index the query should be submitted. In addition, neighboring super-peers know about the special objects resulting from the indexing at super-peer level. They are also indexed to form what is called routing indexes. The super-peer uses its routing index to decide to which neighboring super-peer route the query. Again the query is submitted to specific locations of the respective index at the destination super-peer. The routing indexes (and even the super-peer indexes) can contain statistical information telling where is more promising to send the query when approximate solutions are feasible.

The proposal in this work is based on the observation that the above approach is constructed in a bottom-up manner where *local indexing* plays a major role. However, this contradicts the nature of what we called the *special objects* which one needs them to be as selective as possible in order to reduce the total number of distance evaluations

associated with the solution of queries and the total number of communication actions.

Intuitively the local indexing approach lacks ability to properly select the special objects which are the most selective ones considering the whole database. It lacks a global view of the data. Nevertheless the super-peers tend to solve this problem by indexing the special local objects to get global ones as a result. But this triggers an unnecessarily large number of distance evaluations at search time since the super-peer index has to be scanned to get the right local special objects to later be searched in the respective peers. Also as data at each peer is indexed with poor-quality special objects, queries tends to generate more distance evaluations in peers as well. This can be critical when we consider that peers are expected to be small computers.

At indexing time, our approach first tries to determine (almost) global special objects (LC centers as described below) by performing a sort of tournament among peers attached to a given super-peer and among neighboring super-peers. Then when agreement has been reached, each peer indexes its local objects by using the same global special objects. As they are the same at super-peer level, searching the super-peer index costs very few distance evaluations, just the comparison of the query against the respective special objects. Like in the local indexing case, queries can be then directly sent to specific sections of the indexes located in the selected peers. When the system ensures that all super-peers have been considered in the tournament we get the exact global special objects. However, our experimental results show that such an exhaustive scan is not necessary to achieve better performance than the local indexing approach. Also because super-peers agree on similar global special objects we do not need routing indexes. We just index neighboring special objects into the same super-peer index. Namely, we do not need a secondary index to route queries to neighboring super-peers.

In the remaining of this paper we describe our approach and present experimental results. Section 2 describes the proposed methods to build a P2P index upon a scheme based on peers and super-peers and perform range search operations upon it. Section 3 presents experimental results that compare the same index under the context of local and global indexing. Finally section 5 presents conclusions.

II. Semi-global indexing and search

A. Index construction

We define B_i to be the collection of N database objects stored in the peer i. We assume an initial P2P system with N_s super-peers, each containing N_p peers with $N_s \ll N_p$. We define M to be the number of special objects which we call *centers*. Thus the index in each peer and super-peer has M centers (below we describe how to relax this to have a variable number of centers and objects in peers and super-peers). Like [11] we employ clustering to select the centers c_i in each peer and super-peer. However we pay attention to the heuristic used to select them which tries to widely cover the metric-space defined by the database objects. As clustering method we use the List of Clusters (LC) strategy presented in [7] and enhanced in [20].

We determine the set of M global centers by performing a tournament which starts at each peer associated with every super-peer. In a peer i we select as the next center a database object $o \in B_i$ that maximizes the sum of the distances $d(o, c_k)$ where c_k are the previously selected centers. This is repeated until getting M centers c_k. Also each time a new center c_k is obtained, the $K-1$ database objects that are the nearest ones to c_k are removed from the tournament with $K = N/M$. We then collect all $M{\cdot}N_p$ peer centers c_k in their respective super-peers, and select a new set of M centers from them using the same heuristic with $K = N_p$. After this, each super-peer gets the M centers obtained by its n neighboring super-peers and applies the heuristic over those centers and their own centers with $K = n + 1$ (this can be extended one or more hops away across neighboring super-peers or apply a kind of circulating ring strategy among communities of super-peers). In any case, upto this point, the cost in communication is identical to the local indexing one [11].

After each super-peer has obtained its M semi-global centers, they are broadcast to their peers and neighboring super-peers. The peers use these centers to index their local objects so that the K-nearest objects are associated with each center c_i with $K = N/M$. They are stored in buckets of size K. For each center c_i the peers return to their respective super-peer the tuple (i, p, r_m, r_x) where i is the identifier of the center, p the identifier (IP address) of the sending peer, r_m the distance between c_i and the nearest object to c_i stored in the respective bucket, and r_x the distance to the farthest one. In local indexing these tuples are also sent to the super-peers, so the extra-cost of the semi-global approach is no more than twice the cost of the local one because the communication of centers from super-peers to peers.

The super-peers store the tuples (i, p, r_m, r_x) in the respective buckets associated with the centers c_i. Also for each center c_i a tuple (i, s, r'_m, r'_x) is created where s is the super-peer address, r'_m the minimum r_m of the tuples (i, p, r_m, r_x) stored in the bucket and r'_x the maximum r_x in those tuples. Then all super-peers send to their neighbors their tuples (i, s, r'_m, r'_x). We denote the received tuples as $(i, s, r'_m, r'_x)^*$. Finally the super-peers include in their buckets the received tuples. At this point the respective center objects c_s are already stored in the super-peer from the previous step. Thus the received tuples are increased as $(i, c_s, s, r'_m, r'_x)^*$ before being stored in the bucket whose center c_i is the nearest one to c_s. Also the values r_m and r_x of the respective center tuple (c_i, r_m, r_x) are updated with the arriving r'_m and r'_x (if necessary).

In this way the index data structure kept at each super-

peer is formed by a sequence (c_i, r_m, r_x, b_i) with i= 1 ... M where b_i is a pointer to the bucket associated with each center c_i. And each bucket pointed to by b_i is composed of a sequence of tuples (i, p, r_m, r_x) and $(i, c_s, s, r'_m, r'_x)^*$ where the last tuple can be further enhanced by pre-computing the distance $d(c_i, c_s)$ which can be used to reduce the number of times the distance $d(q, c_s)$ has to be computed at search time as we explain below.

In the super-peers all centers from peers that are not global centers to the super-peer are discarded to release space. The data structure kept at each peer is simple since they all see the same set of centers and queries comes from the respective super-peer with already calculated information about what buckets are to be examined. The bucket kept at each peer for a center c_i contains the set of database objects which are the closet ones to c_i in the peer. Here further optimizations can be made to reduce the number of objects that must be compared against the query.

We can keep a table with a few columns containing distances to pivots, one per column. The first pivot is the center itself an the column stores the distance of each object in the bucket to the center. This column is kept sorted in ascending order of these distances. So two binary searches on the column determine the range of rows where the objects that can be potentially part of the answer to the query are located. The second column contains as pivot the center that is the farthest one to c_i. The third is the closet one to c_i, and so on interleaving. These additional columns are used to further narrow the number of objects to be compared with the query. In [6] it is shown that good pivots are the ones very far or very close to the objects being examined by the query.

Notice that when one uses one column of the above table we have a situation that can be considered as equivalent to the B+ Tree in the iDistance approach discussed in [11], [14]. However as we include more columns our approach becomes more effective in reducing the number of candidate object which finally do make into the results of the range query (we call this *effectiveness* and show results about this in the section of experiments).

B. Range search algorithm

A range search (q, r) with object q and radius r consists on retrieving all the objects located within the query ball. We assume that the queries arrive to any super-peer. The centers (c_i, r_m, r_x, b_i) with i= 1 ... M of the super-peer are traversed and each time the inequality $d(q, c_i) + r \geq r_m$ and $d(q, c_i) - r \leq r_x$ holds true, the bucket b_i is traversed meaning that the query ball intersects the range $(c_i, r_x) - (c_i, r_m)$. Traversing a bucket b_i means dealing with tuples $t_p = (i, p, r_m, r_x)$ and $t_s = (i, c_s, s, r_m, r_x)$.

Upon a tuple t_p, the algorithm computes the inequality $d(q, c_i) + r \geq t_p.r_m$ and $d(q, c_i) - r \leq t_p.r_x$ and if it holds true, then the super-peer sends a message (q, r, i) to the peer p. At this point the distance $d(q, c_i)$ has been already calculated. When the message is received in the peer, the objects in the bucket associated with the center c_i are compared against the query. Each object o for which $d(q, o) \leq r$ holds true, is placed in the result set for the query (q, r).

Upon a tuple $t_s = (i, c_s, s, r_m, r_x)$, it is necessary to compute $\ell = d(q, c_s)$ and evaluate the inequality $\ell + r \geq t_s.r_m$ and $\ell - r \leq t_s.r_x$ and if it holds true, then the super-peer sends a message (q, r, i, ℓ) to the neighboring super-peer s so that the search can continue in the bucket associated with the center i in the super-peer s, and so on recursively.

C. Dynamism

Every time a new peer wants to participate in the system it sends a request to one of the known super-peers which passes back the global centers (c_i, r_i, s_i), where r_i is the average radius calculated considering the covering radii of the current peers attached to the centers c_i and s_i the standard deviation. The peer indexes its local data using the centers by attaching its database objects to the first center (in the construction order indicated by the subscripts of c_i) for which the object is within the range defined by (c_i, r_i^*). We allow a tolerance of $r_i^* = 1.5\, s_i + r_i$ over the average r_i. We use this method of insertion to detect cases in which new peers provide to the system some objects that can potentially become global centers because they are significantly different from the current global ones.

The local database objects that cannot be inserted in one of the clusters defined by the global centers are stored in an overflow area. The center of this area is the current object that maximizes the sum of the distance to all global centers. This center can be replaced by another object if during the process of index construction in the peer a new object arrives to the overflow area which has a larger value for the cumulative sum of the distance to all global centers. This center is sent to the super-peer and is included in a respective overflow area of the super-peer index.

The super-peer handles its overflow area as it were a local index. This is done upto a threshold value for the size of the overflow area is achieved, in which case a tournament is effected on these local centers to elect one or more global centers. This time, however, the tournament is slightly different because we need to estimate the number of new global centers that must be calculated. This because the objects in the overflow area can belong to very different regions of the metric-space. To this end we form groups of centers by considering the degree in which the centers intersect each other.

The degree $S_{1,2}$ at which cluster (c_1, r_1) intersects cluster (c_2, r_2) is calculated as follows: **(i)** they intersect each other if $d(c_1, c_2) \leq r_1 + r_2$, and the degree $S_{1,2}$ is set to an initial value of 1 (or 0 otherwise), **(ii)** one is contained into the another if $d(c_1, c_2) \leq |r_1 - r_2|$, **(iii)** if contained and

$r_1 < r_2$, the degree $S_{1,2}$ is reduced by $S_{1,2} = (r_1/r_2) \cdot S_{1,2}$, **(iv)** otherwise if $r_1 > r_2$ the $S_{1,2}$ is increased to $S_{1,2} = 1 + r_2/r_1$, and **(v)** if they in fact do not contain each other, they just intersect, then $S_{1,2}$ is decreased to $S_{1,2} = (\,|r_1 - r_2|/d(c_1, c_2)\,) \cdot S_{1,2}$.

The values $S_{i,j}$ are used to do clustering of the centers stored in the overflow area of the super-peer. This clustering starts by selecting one of the centers at random, say center (c_1, r_1). All centers k for which $S_{k,1}$ is 0 are considered candidates to become new global centers (c_k, r_k). The remaining centers (c_j, r_j) are associated to the candidate center (c_k, r_k) for which the value $S_{j,k}$ is maximum. After this, for each group of clusters one of them is selected as a new global center. We select the one that maximizes the sum of the distances to all current global centers. These centers are communicated to all peers and neighboring super-peers which are used to re-index their respective overflow areas.

D. Dynamism in SimPeer

Unfortunately in the description presented in [11] (below we call it *KM strategy*) no precise details are given on how their scheme handles the arrival of new peers to the system. Thus in this section we describe how we have modified the implementation of the strategy described in [11] to include this case. We assume there are N_p peers connected to N_s super-peers. When a new peer P_i is joining the P2P network, we apply the KM strategy upon its database objects and obtain a list of M cluster descriptions (i.e., a sequence of M centers represented by the pairs (c_i, r_i) where c_i is the center object and r_i is the covering radius). We must also connect the peer to its locally closest super-peer. For that, the new peer sends its list of M centers (c_i, r_i) to a contact super-peer SP_c of the network. The super-peer SP_c computes the intersection degree between the clusters (c_i, r_i) of the peer and its own clusters. SP_c also computes the intersection degree with the clusters of its neighbor super-peers. Finally, the closest super-peer is selected and P_i is assigned to it. To emulate an actual distributed system, the super-peer SP_c is selected uniformly at random from the N_s super-peers.

We define the closest super-peer SP_j to a peer P_i as the one with the greatest cumulative sum of the quantities $|d(c_i, c_j) - r_{c_i} - r_{c_j}| \forall\, c_j \in SP_j$ and $c_i \in P_i$. In the rare event that the cumulative sum is zero for all SP_c's neighboring super-peers, then the peer P_i is assigned to the contact super-peer SP_c. When this happens the clusters of SP_c are rebuilt by using the K-means algorithm.

III. Experimental results

To obtain performance results we used the Metric Spaces Library SISAP (http://www.sisap.org/Home.html) for generating synthetic vector data collections (uniform and Gaussian) and queries. Another data set representing image objects was generated synthetically as follows. We took a collection of images from a NASA data set containing 40,701 images vectors (dimension 20), and we used it as an empirical probability distribution upon which we generated our random image objects. We call this data set NASA. The query log for this data set was generated by taking randomly selected objects from the same set. These data sets contain 3,000,000 objects each. We use the Euclidean distance with these collections to measure the similarity between two objects. We used range search radii of R1= 0.1, R2= 0.6 and R3= 0.7 for the uniform data set (dimension 16); R1= 0.7, R2= 1.5 and R3= 5.0 for the Gaussian data set (dimension 16); and R1= 0.01, R2= 0.07 and R3= 0.7 for the NASA data set. These radii were selected experimentally to obtain a reasonable number of results. Figure 10 shows the all-object-pairs distance distribution for the data sets.

Below we show results normalized to 1 in order to better illustrate comparative performance in terms of percentage differences among the strategies. Also when we write *LC strategy* we refer to the proposal described in Section II which is based on our approximation to global indexing. For the purpose of comparing global and local indexing under the same setting, we have also implemented a realization of the LC strategy which indexes local centers coming from peers in the same way as it is made in [11], that is, indexing in a bottom up fashion. We call *KM strategy* the implementation of the strategy presented in [11] which has been enhanced with the procedure described in Section II-D.

We use a system composed of 30 super-peers and 1,000 peers, each peer starts up with 10 centers and we use 10 global centers. For super-peers we use two interconnection topologies: an all-to-all and a ring topology (Figure 9 shows a comparison between the two). The topology affects in the same way to the LC and KM strategies since in both cases indexing is constrained to each super-peer and its immediate neighboring super-peers. In the first set of experiments the database objects are distributed uniformly at random onto the peers (Figures 1, 2, 3, and 4 and Table I).

For the second set of experiments the database objects are distributed onto the peers by trying to focus them on different regions of the metric-space. We employ the pivot selection heuristic proposed in [5] which produces pivots evenly distributed in the metric-space. One pivot is selected for each peer and the database objects that are the closest to the pivot are stored in the same peer. We study two configurations. In the first one, the peers are clustered by similarity to decide to which super-peer they are connected (Figures 5, 6 and 7). The second one connects each peer to any super-peer selected at random (Figure 8 presents a comparison between the two cases).

A. Constant number of peers

To show the advantages of trying to approximate global indexing we show in Figure 1 results for the LC strategy under global and local indexing. These are results for the three radii r used in range queries (q, r). The Figure shows

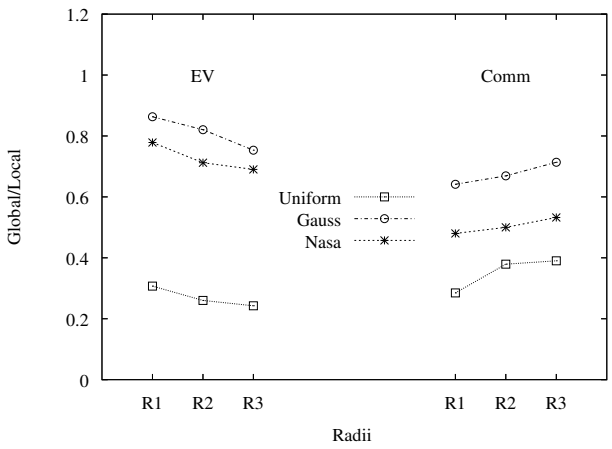

Figure 1. Total number of distance evaluations and messages for global and local indexing by using the LC strategy.

Table I
PERCENTAGE OF EFFECTIVENESS.

Gauss		
radius	LC	KM
0.7	55.7	0.2
1.5	55.6	2.5
5.0	62.1	10.5

Nasa		
radius	LC	KM
0.01	56.0	0.2
0.07	59.8	2.5
0.1	61.4	20.8

Uniform		
radius	LC	KM
0.1	58.0	0.2
0.6	61.4	2.5
0.7	64.7	17.4

results for the total number of distance evaluation and total amount of communication. For each metric we show the ratio global to local indexing (as described above global indexing is the LC strategy with global centers determined considering large set of peers whereas local indexing is the LC strategy with local centers determined as proposed in [11]). In all cases global indexing outperforms local indexing. Notice that we observed that the KM strategy [11] performed about 10 times more distance evaluations than the LC with our proposal for global indexing. This strategy has to pay more distance evaluation since it also indexes the centers of its neighboring super-peers.

In addition, our technique of using the pivoting table with two or more columns allows the LC to be more effective in reducing the number of candidate objects that are finally compared with the query. In this respect, the pivoting strategy is more efficient than the iDistance B+ Tree used in [11]. This measure is captured in Table I which shows the percentage of objects that are compared with the query *and* become part of the query answer (a larger value indicates a greater effectiveness).

B. Increasing the number of peers

In the following experiments we evaluate the performance of the global LC strategy and compare it with the KM strategy under a dynamic P2P system. The P2P system is initialized with 50% of the N_p peers and the remaining 50% join the system in two groups to reach $0.75\,N_p$ and N_p with N_s kept constant, and for each group the same number of queries are processed with both strategies. For each set of queries we determine three metrics given by the total number of distance evaluations, the total amount of communication and the percentage of effectiveness as described for the results in Table I.

Figure 2 shows the number of distance evaluations performed by the LC and KM strategies as we increment the number of peers in the network. The x-axis shows the percentage of peers involved during the query searches and y-axis shows the normalized number of evaluations performed. As new peers join to the network the algorithms require more distance evaluations to processes queries, because the system has more information and more candidate clusters may be selected. Figure 3 shows the communication in bytes required to process a batch of Q query in different intervals of number of peers with the same tendency as Figure 2. In both graphics the LC strategy outperforms the KM strategy.

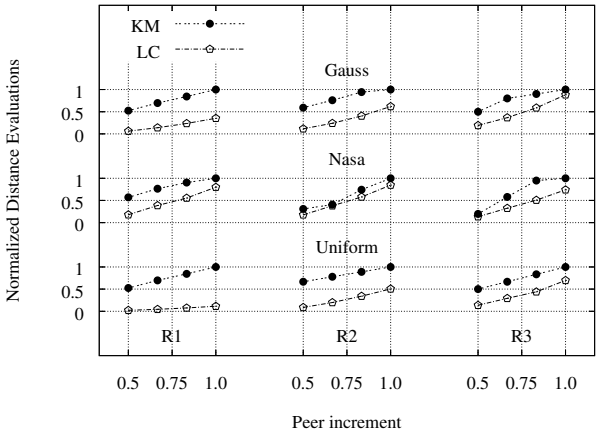

Figure 2. Distance evaluations for $0.5N_p$, $0.75N_p$ and N_p peers.

Figure 4 shows the effectiveness of the algorithms for the same experiment. This last figure shows the rate of the average number of peers reporting results for queries over the total number of peers visited. The LC strategy exhibits the most efficient performance.

The next set of experiments show performance results for the case in which peers are destinated to specific regions of the metric space and each peer is assigned to its closet super-peer in terms of similarity. We compare this case against the case in which objects are distributed at random among the peers. We call the two cases as *Sim* and *Random* respectively. Figure 5 shows results obtained by the LC and KM strategies for the two cases (the values for LC and KM are relative to themselves). Figure 6 shows the communication rate for the same experiment and Figure 7 shows the effectiveness. In overall, those results show that the performance of LC tends

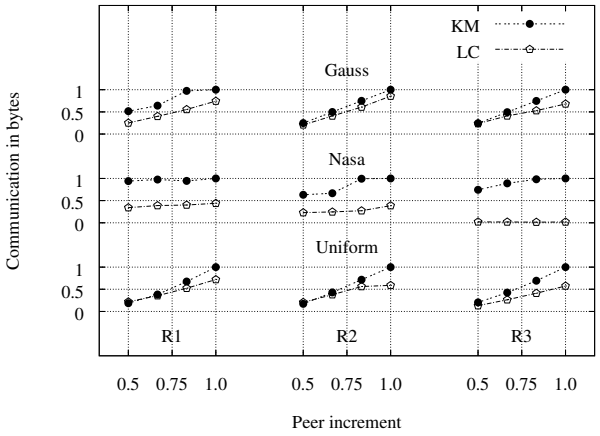

Figure 3. Communication in bytes for $0.5N_p$, $0.75N_p$ and N_p peers.

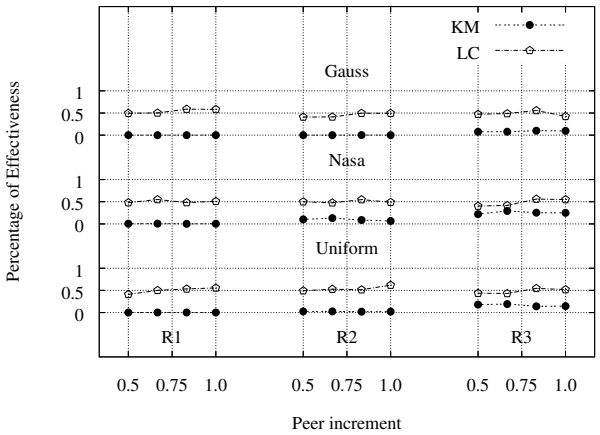

Figure 4. Quality of the indexes for $0.5N_p$, $0.75N_p$ and N_p peers.

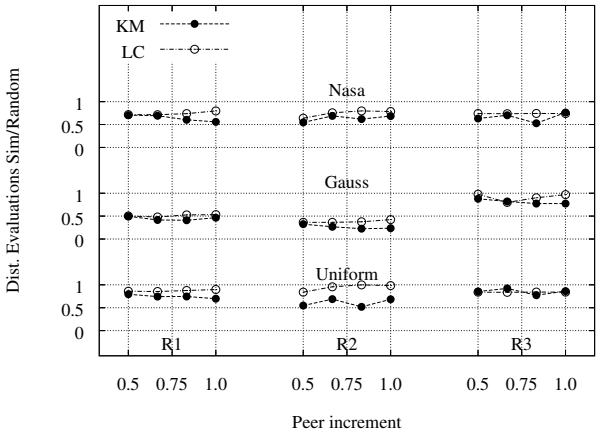

Figure 5. Difference in the number of distance evaluations performed using random and sim assignment.

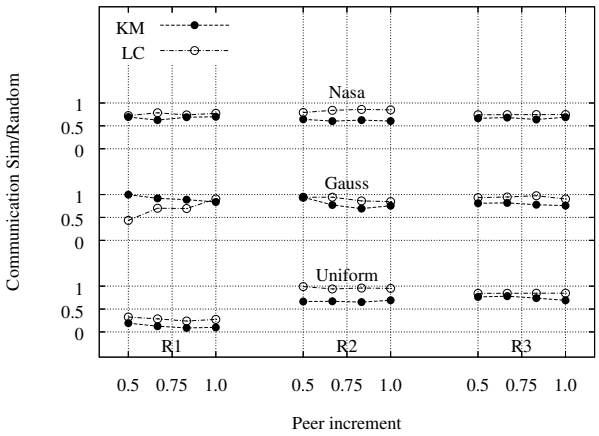

Figure 6. Difference in the communication performed by the LC and K-M algorithms using random and sim assignment.

to improve in the Sim case, and it improves in a fairly larger degree than the improvement observed in the KM strategy.

Figure 8 shows the performance of the LC algorithm using the Uniform collection, when the peers are assigned to its closest super-peer (*Closest*) and when peers are assigned at random (*Random*). Except for communication, those results shows that the LC strategy is less sensitive to the actual super-peer assignment. What is more relevant is the effect of focusing peers to specific regions in the metric-space.

Figure 9 compares the performance of the LC algorithm under runs using two types of connections among super-peers, namely ring and all-to-all. Those results show the LC strategy is not very sensitive to the communication topology apart from the fact that, as expected, all-to-all generates a higher message traffic.

Table II shows the number of peers out of 1,000 peers involved in the solution of range queries. The columns show the peers that were contacted and contributed with no results (N.R), the peers that contributed with results (R) and the total. Global indexing effectively reduces contacted peers.

IV. Conclusions

In this paper we have shown that our realization of a clustering index data structure based on global indexing can be an efficient strategy to perform query processing in P2P systems composed of super-peers and peers. Our strategy adapts itself well to the cases in which peers join dynamically the P2P system, which is an important requirement for this kind of systems.

We have shown that our proposal consistently outperforms the state of the art strategy for this type of P2P infrastructure [11]. Super-peers are expected to be stable computers which are used as coordinators whereas peers can be small user computers that intermittently participate in the system by making available their local collection of database objects to other users. In the local indexing approach proposed in [11] the new peers joining the system locally index their database objects using K-means clustering and send the cluster centers to the super-peers to be further indexed at that level.

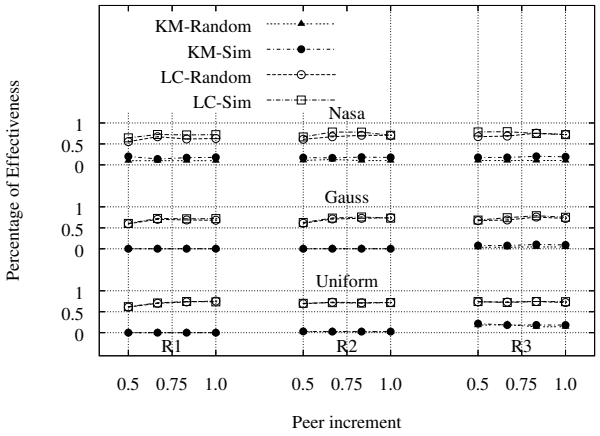

Figure 7. Quality achieved by the indexes under random and sim assignment.

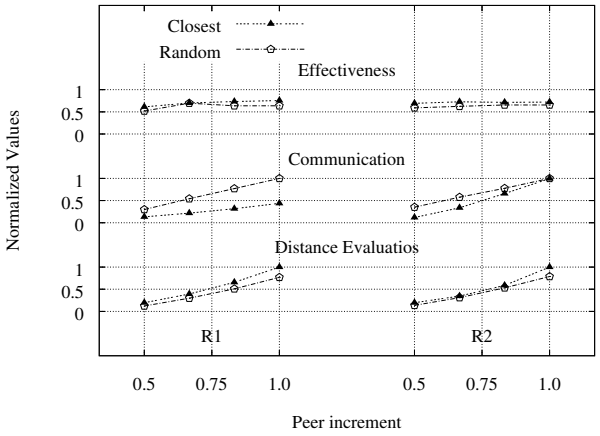

Figure 8. Performance of the LC algorithm when peers are assigned to the closest super-peer vs. when peers are assigned at random with the Uniform collection.

Essentially our approach is the reverse situation, namely super-peers communicate the relevant centers to the joining peers which makes the crucial difference in performance since those centers are expected to be more effective in driving queries to the relevant peers holding candidate objects for the answer to queries. We have implemented this strategy on top of the List of Clusters [7] index data structure enhanced as proposed in [20]. Our results with different databases show that this scheme performs well in a variety of cases, and it outperforms global and local indexing implemented using K-means clustering.

ACKNOWLEDGMENTS

We would like to thank Dr. Rodrigo Paredes for helpful discussions on the design of the dynamic LC-P2P method. We would also like to thank an anonymous reviewer for useful comments to improve the presentation of results and clarification of several details on the experiments.

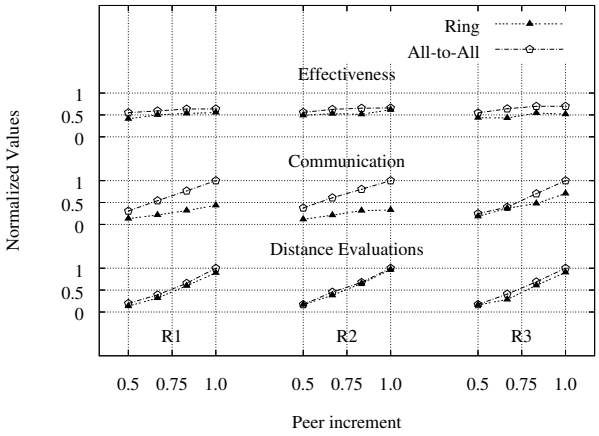

Figure 9. Performance of the LC algorithm under two network topologies and the Gauss collection.

Table II
AVERAGE NUMBER OF PEERS HIT BY QUERIES OUT OF 1,000.

Gauss						
LC	Local			Global		
radius	N.R.	R.	Total	N.R.	R.	Total
0.7	102.81	44.34	147.15	26.73	39.35	66.08
1.5	144.78	52.06	196.84	24.92	41.25	66.17
5.0	232.88	54.33	287.21	28.88	48.10	76.98
Nasa						
LC	Local			Global		
radius	N.R.	R.	Total	N.R.	R.	Total
0.01	24.01	54.65	78.66	22.82	36.84	59.66
0.07	24.3	58.02	82.32	23.71	36.63	60.34
0.1	32.04	78.5	110.54	24.8	38.99	63.79
Uniform						
LC	Local			Global		
radius	N.R.	R.	Total	N.R.	R.	Total
0.1	39.73	172.4	212.13	35.33	122.07	157.4
0.6	40.25	193.14	233.39	36.32	133.02	169.34
0.7	40.68	194.28	234.96	36.76	160.69	196.85

REFERENCES

[1] F. Banaei-Kashani and C. Shahabi. SWAM: a family of access methods for similarity-search in peer-to-peer data networks. In Proceedings of CIKM'04, pages 304-313, 2004.

[2] M. Bawa, T. Condie, and P. Ganesan. LSH forest: self-tuning indexes for similarity search. In Proceedings of WWW05, pages 651-660, 2005.

[3] A. R. Bharambe, M. Agrawal, and S. Seshan. Mercury: supporting scalable multi-attribute range queries. In Proceedings of SIGCOMM'04, pages 353-366, 2004.

[4] I. Bhattacharya, S. R. Kashyap, and S. Parthasarathy. Similarity searching in peer-to-peer databases. In Proceedings of ICDCS'05, pages 329-338, 2005.

[5] N. R. Brisaboa, A. Farina, O. Pedreira, and N. Reyes. Similarity search using sparse pivots for efficient multimedia information retrieval. In Proceedings of ISM'06, 2006.

[6] C. Celik. Effective use of space for pivot-based metric indexing structures. In Proceedings of SISAP'08, 2008.

[7] E. Chavez and G. Navarro. A Compact Space Decomposition for Effective Metric Indexing. Pattern Recognition Letters 26(9) pp. 1363–1376, 2005.

[8] A. Crainiceanu, P. Linga, J. Gehrke, and J. Shanmugasundaram. P-tree: a P2P index for resource discovery applications. In Proceedings of WWW'04, 2004.

[9] A. Crainiceanu, P. Linga, A. Machanavajjhala, J. Gehrke, and J. Shanmugasundaram. P-ring: an efficient and robust p2p range index structure. In Proceedings of SIGMOD'07, 2007.

[10] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In Proceedings of ICDCS'02, page 23, 2002.

[11] C. Doulkeridis, A. Vlachou, Y. Kotidis, and M. Vazirgiannis. Peer-to-peer similarity search in metric spaces. In proceedings of VLDB'07, 2007.

[12] F. Falchi, C. Gennaro, and P. Zezula. A content-addressable network for similarity search in metric spaces. In Proceedings of DBISP2P'05, pages 126-137, 2005.

[13] P. Haghani, S. Michel, K. Aberer. Distributed similarity search in high dimensions using locality sensitive hashing. In Proceedings of VLDB'07, 2007.

[14] H. V. Jagadish, B. C. Ooi, K. Tan, C. Yu, and R. Zhang. iDistance: An adaptive B+ tree based indexing method for nearest neighbor search. TODS, 30(2), 2005.

[15] H. V. Jagadish, B. C. Ooi, and Q. H. Vu. Baton: a balanced tree structure for peer-to-peer networks. In Proceedings of VLDB'05, pages 661-672, 2005.

[16] H. V. Jagadish, B. C. Ooi, Q. H. Vu, R. Zhang, and A. Zhou. VBI-tree: A peer-to-peer framework for supporting multi-dimensional indexing schemes. In Proceedings of ICDE'06, page 34, 2006.

[17] P. Kalnis, W. S. Ng, B. C. Ooi, and K. L. Tan. Answering similarity queries in peer-to-peer networks. Inf. Syst., 31(1):57-72, 2006.

[18] B. Liu, W.-C. Lee, and D. L. Lee. Supporting complex multi-dimensional queries in P2P systems. In Proceedings of ICDCS'05, pages 155-164, 2005.

[19] T. Luu, G. Skobeltsyn, F. Klemm, M. Puh, I. Podnar Zarko, M. Rajman, K. Aberer. AlvisP2P: Scalable Peer-to-Peer Text Retrieval in a Structured P2P Network. In Proceedings of VLDB'08 2008.

[20] M. Marin, G.V. Costa, R. Uribe. Hybrid Index for Metric Space Databases. In Proceedings of ICCS'08, 2008.

[21] D. Novak and P. Zezula. M-Chord: a scalable distributed similarity search structure. In Proceedings of InfoScale'06, page 19, 2006.

[22] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In Proceedings of SIGCOMM'01, pages 161-172, 2001.

[23] O. D. Sahin, F. Emekci, D. Agrawal, and A. E. Abbadi. Content-based similarity search over peer-to-peer systems. In Proceedings of DBISP2P'04, pages 61-78, 2004.

[24] 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 SIGCOMM'01, pages 149-160, 2001.

[25] C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. In Proceedings of SIGCOMM'03, 2003.

[26] E. Tanin, A. Harwood, H. Samet. Using a distributed quadtree index in peer-to-peer networks. VLDB J. 16(2): 165–178 (2007)

[27] A. Vlachou, C. Doulkeridis, Y. Kotidis, and M. Vazirgiannis. SKYPEER: Efficient subspace skyline computation over distributed data. In Proceedings of ICDE'07, 2007.

[28] B. Yang and H. Garcia-Molina. Designing a super-peer network. In Proceedings of ICDE'03, pages 49-59, 2003.

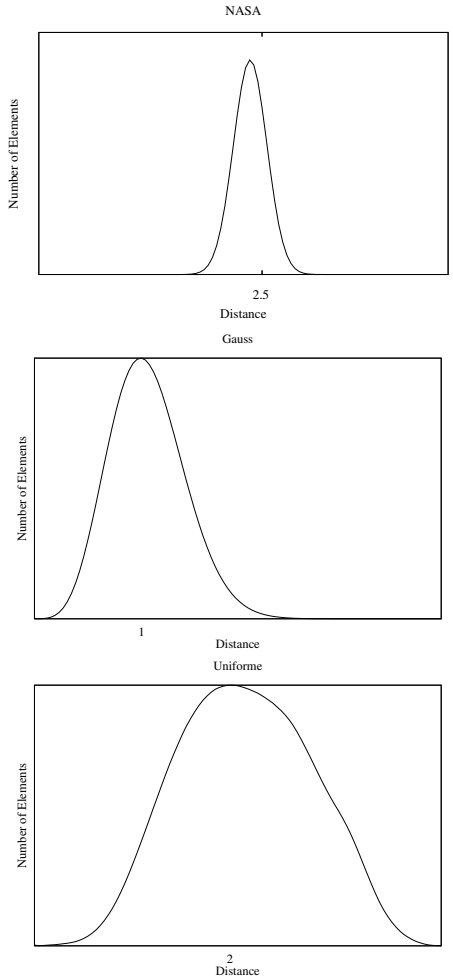

Figure 10. All-pairs distance distribution.