ISSN:0975-9646

P. Lalitha Kumari et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 2 (5) , 2011, 2078-2083

Efficient way of Data Managing for Range Queries in Unstructured Peer to Peer Networks.

P Lalitha Kumari
Dept. of CSE
ASR College Of Engg,
Tanuku,Andhrapradesh,India.

P Rama Rao
Dept. of MCA
Sri Sivani College of Engg,
Andhrapradesh,India.

Chinnam YuvaRaju
Dept. of CSE
ASR College Of Engg
Tanuku,Andhrapradesh,India

Abstract— **Peer-To-Peer (P2P) networks have become very popular in the last few years. Nowadays, they are the most widespread approach for exchanging data among large communities of users in the file sharing context. Efficient way of managing storage and retrieval of multidimensional data is achieved by proposed framework which ensures robust query evolution. This framework is based on peer to peer network, where large collection of Data to be stored. This data is divided into subparts and built up an index on set of each compressed data and this data is to be distributed across p2p network .This compressed data supports efficient data extraction of information .A replication mechanism provides appropriate coverage of index and metadata by considering network conditions and query workload .**

Keywords- multidimensional data, indexing, compression and p2p network

I. Introduction

PEER-TO-PEER (P2P) networks have become very popular in the last few years. Nowadays, they are the most widespread approach for exchanging data among large communities of users in the file sharing context.

In order to make participants really autonomous, they should be imposed no constraint on storage and [1] computational resources to be shared, as well as on the reliability of their network connection. These requirements make traditional distributed frameworks unsuitable and suggest the adoption of a solution based on an unstructured P2P network, where peers are neither responsible of coordination tasks (such as super peers, which are called for a certain amount of resources and reliability), nor imposed to host specific pieces of data (as in DHT-based networks).

Our aim is devising a P2P-based framework supporting the analysis of multidimensional historical data. Specifically, our efforts will be devoted to combining the amenities of P2P networks and data compression to provide a support for the evaluation of range queries, possibly trading off efficiency with accuracy of answers. [2] The framework should enable members of an organization to cooperate by sharing their resources (both storage and computational) to host (compressed) data and perform aggregate queries on them, while preserving their autonomy.

The management of compressed data on unstructured P2P Networks is an intriguing issue, but poses several research Challenges, which we are discuss in the following.

A. Compression

A compression technique must be devised which is able to create "prone-to-be-distributed" data synopses supporting the efficient evaluation of aggregates, possibly affected by tolerable error rates [3]. However, in this case, although the cost of disk storage is continuously and rapidly decreasing, it may still be difficult to find peers for which hosting replicas of synopses has a negligible cost, while autonomy is a requirement in our setting? Using traditional compression techniques, synopses providing reasonable error rates may have a non-negligible size (usually not under 1 percent of the size of the original data set, e.g., 1'' GB from a 1' TB data set). Although compressing the data certainly makes replication less resource consuming, [4]replicating the entire synopsis each time would require storage and network resources that could be saved if only some specific portion of the synopsis could be replicated [2]. We recall that replication is mandatory in the P2P setting, both to contrast the volatility of peers (which threatens data availability) and to prevent peers
from being overloaded (in the presence of many users interested in a data set, if the peers hosting these data were too few, they would be required to process a large amount of queries).

These drawbacks would be overcome if the compressed synopsis were subdivided into tiny sub synopses which are Independently replicated and disseminated on the network when needed. Peers would, therefore, be asked to host replicas of small chunks of data. This way, the autonomy requirement would not result in a limit on the overall size of the synopsis

B. Indexing

A better way to address this issue is to design an indexing mechanism that supports the efficient location of the sub synopses involved in the query evaluation. In the literature, there are several works proposing distributed indexing techniques, where indexes are variants of R-Trees which are partitioned and distributed among the nodes of the network. [1] According to these approaches, nodes of the networks are assigned groups of nodes of the R-tree, and maintain references to hosts which are assigned other nodes of the R-tree. The association between hosts and R-tree nodes is fixed and the maintenance of the index is centralized. These solutions, as they are, were devised for relatively static scenarios, and they are not suitable for the dynamic scenario addressed by our proposal, where in order to guarantee peer autonomy, peers cannot be constrained to host a certain portion of the index or to be always connected to the network; and [1]Peers are volatile, so the framework must be capable of promptly reacting to peer disconnections, [5] preventing dangling references in the index.

C. Replication

A replication scheme capable of maintaining appropriate levels of coverage w.r.t. the evolution of user interests and network conditions must be designed, to ensure accessibility and robustness.

The main contributions of this paper and its organization may be summed up as follows [6].

- a compression technique for building an indexed aggregate structure over a multidimensional data population, prone to be distributed, and accessed across a P2P network ;
- a storage model which employs additional data structures to support efficient and robust query answering over compressed data in an unstructured P2P network ; and
- a dynamic replication scheme capable of maintaining appropriate levels of coverage w.r.t. the evolution of the query workload and the network conditions with proposal work.

II. Compression And Indexing DatA

This section consists of three subsections that are partitioning, compression, and indexing.

A. Partitioning. The aim of the partitioning step is to divide the data domain into non overlapping blocks. These blocks will be compressed separately, yielding distinct sub synopses. For each of them, a portion of the amount of storage space B chosen to represent the whole synopsis will be invested. The distribution of B among blocks will take into account the following requirements. B must be fairly distributed among blocks and each block must be assigned a “small” portion of B.

The assignment of different amounts of storage space to the blocks for representing their sub synopses should depend on the differences in homogeneity among the blocks. Intuitively enough, the more homogeneous the data inside a block, the smaller the amount of information needed to effectively accomplish its summarization.

The sub synopses over the blocks are the data that will be hosted by peers and exchanged across the P2P network. As explained in Section 1, [1] building sub synopses with “large” size would impose a significant constraint on the amount of storage space which should be made available by each peer. [7] On the contrary, defining small-size sub synopses results in limiting the storage and computational resources required at each peer for storing and querying data, as well as reducing both the download and upload traffic needed for supporting data exchange.

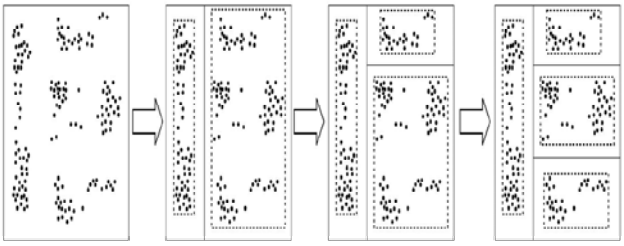


Fig. 1. Partitioning a 2D data population

B. Compression. Clustering-based Histogram (CHIST) exploits a density-based clustering algorithm to construct a set of (possibly overlapping) blocks covering the nonempty portions of the data domain [8]. For each block (called bucket, according to standard histogram terminology), its boundaries as well as some aggregate value summarizing its data are stored. In our current implementation, each bucket is associated with the result of evaluating the sum aggregate operator. This way, the summary data suffice to estimate range sum queries.

C. Indexing. At this step, an index is built on top of the sub synopses resulting from the compression step. This index will be exploited for locating the data involved in the queries across the network. [9] The aggregate R-tree indexing the sub synopses will be denoted as I.

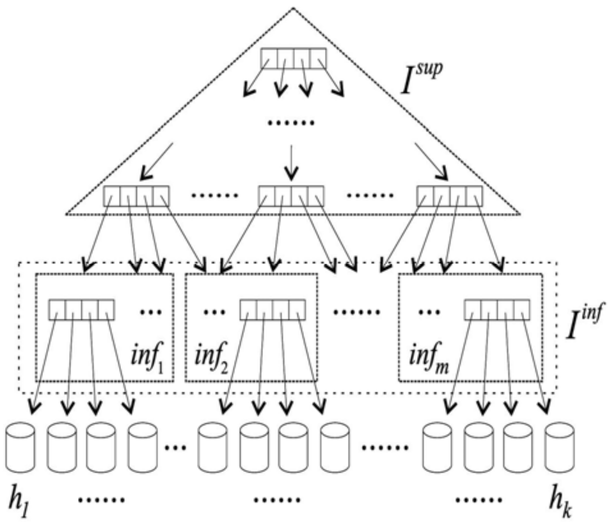


Fig.2. Partitioning the R-tree.

D. Partitioning the Index

After being populated, I is partitioned in “small”-size portions which are prone to be distributed across the network.

The reason for partitioning the index is the same as for limiting the amount of storage space invested for a single synopsis, that is, distributing small-size index portions across the network prevents peers from being overloaded in terms of upload and download traffic needed for supporting index replication.

III. PROPOSED METHOD

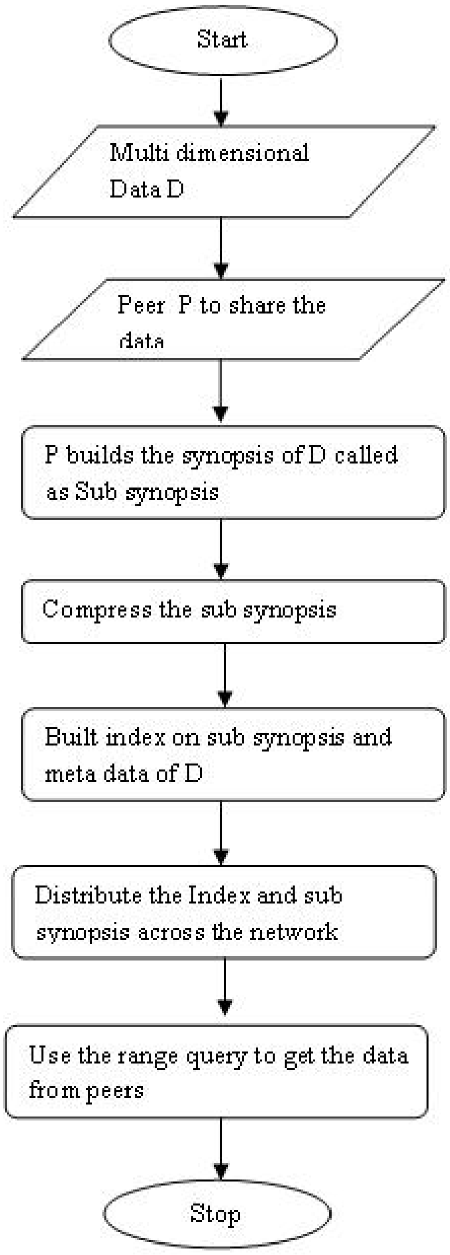


Fig 3 . proposed method flowchart

Our aim is devising a P2P-based framework supporting the analysis of multidimensional historical data. Specifically, our efforts will be devoted to combining the amenities of P2P networks and data compression to provide a support for the evaluation of range queries, possibly trading off efficiency with accuracy of answers. [1] The framework should enable members of an organization to cooperate by sharing their resources (both storage and computational) to host (compressed) data and perform aggregate queries on them, while preserving their autonomy.

A framework with these characteristics can be useful in different application contexts. For instance, consider the case of a worldwide virtual organization with users interested in geographical data, as well as the case of a real organization
on an enterprise network. In both cases, even users who are not continuously interested in performing data analysis can make a part of their resources available for supporting analysis tasks needed by others, if their own capability of performing local tasks is preserved. [10] This is analogous to the idea on which several popular applications for public resource computing are based. For instance, within the project SETI@home [39], members of a worldwide community offer their CPU, when it is idle, to analyze radio telescope readings in search of nonrandom patterns, such as spikes in power spectra. In order to make participants really autonomous, they should be imposed no constraint on storage and computational resources to be shared, as well as on the reliability of their network connection. These requirements make traditional distributed frameworks unsuitable and suggest the adoption of a solution based on an unstructured P2P network, where peers are neither responsible of coordination tasks (such as super peers, which are called for a certain amount of resources and reliability), nor imposed to host specific pieces of data (as in DHT-based networks).

PEER-TO-PEER (P2P) networks have become very popular in
the last few years. Nowadays, they are the most widespread approach for exchanging data among large communities of users in the file sharing context specifically, no P2P-based solution has imposed itself as an effective evolution of traditional distributed databases. [11]This is quite surprising, as the huge amount of resources provided by P2P networks (in terms of storage capacity, computing power, and data transmission capability) could effectively support data management. Our aim is devising a P2P-based framework supporting the analysis of multidimensional historical data. The multidimensional data is stored in peer so that it can be shared in the network, for that built the synopses.

The synopsis is built in three steps

1. Partition 2. Compressing and 3.Indexing.

The aim of the partitioning step is to divide the data domain into non overlapping blocks. [12] These blocks will be compressed
separately, yielding distinct sub synopses.

An index is built on top of the sub synopses resulting from the compression step Index and these sub synopses are distributed across the network .Queries can be posed against the data. The queries can be any explorative or range queries. One of the first works dealing with the problem of supporting range queries in a peer-to-peer network is where data are ordered according to Hilbert curves, and then, distributed among the peers.

As seen in Section 2, the compression and indexing processes result in a synopsis organized into sub synopses, and a fragmented aggregate R-Tree over them. We now describe how the distribution of the synopsis and the index are performed.

A. System Primitives and Data Structures .

We assume the existence of two system primitives named search and send. Primitive search(N)—which is used by the
framework every time it is required to find sets of peers on the network—returns a set of N IP addresses of randomly chosen peers. In order to choose a peer randomly, it suffices to locate a peer by starting a random walk of length rather than $\log_f N$ (where N is the number of peers

in the network and f is the average fan-out) from the peer which invoked search. In fact, as shown in [1], a random walk of this length makes the probability of reaching any peer converge to a stationary distribution, which is uniform if the network graph is well connected. In our prototype, we set the length of the random walk to 1'. This allows us to randomly select peers from a network of up to $4^{1'}$ peers even in the pessimistic case that the network reaches a condition with average fan-out equal to 4. Primitive send(P; o) transmits s-block o from the peer p which invoked the primitive to the peers whose IP addresses are in set P. In our prototype, this primitive properly avoids overloading p when P is large. This is achieved through decentralized dissemination. instead of sending jPj copies of o, p sends o to a subset of the peers in P which, in turn, keep a copy of o and forward it to different subsets of the remaining peers in P, and so on. We assume that each s-block is uniquely identified throughout the system, and we denote their identifiers as id(I^{sup}), id(inf_i), and id(h_j). [13] Moreover, when needed to avoid confusion, we denote the s-blocks related to a

population D as D.I^{sup}, D.inf_i, and D.h_j. Finally, we assume that each s-block carries along metadata about the population it belongs to. These metadata are denoted as Dm and comprise the name of the population, the schema of the data (dimensionality, names, and ranges of dimensions), as well as some keywords which will be exploited to support search operations across the network.

Our proposed distribution scheme makes use of a set of data structures named as location tables. Each location table will be associated with a copy of an index portion and maintain correspondences between s-blocks and sets of peers. Specifically, the location table associated with I^{sup} will consist of a row for each leaf portion, plus a row for I^{sup} itself. Each row, in turn, will contain addresses of peers where copies of these index portions are hosted. This way, a peer hosting I^{sup} will be able to contact the peers hosting copies of the leaf portions by simply accessing its associated location table. The row for I^{sup} is employed to connect the set of peers that initially host copies of I^{sup} in a clique, i.e., each peer hosting a copy of I^{sup} knows the other peers which are assigned I^{sup} as well (indeed, the location tables of the peers which do not belong to this clique will not contain this row). This way, the survivability of populations can be tightly controlled through a mechanism that replaces a peer of the clique as soon as it exits the system. Further details will be provided in the following.

In a location table associated with a copy of a leaf portion inf_i, each row will contain the addresses of the peers hosting copies of a sub synopsis pointed by inf_i .We denote the location tables associated with index portions as table (I^{sup}) and table (inf_i). At runtime, the local copies of these tables can be modified by the peers that host them; hence, when needed to avoid confusion, we will denote the tables at a peer p as p.table(I^{sup}) and p.table(inf_i).In addition, along with each sub synopsis and leaf portion, the address of one of the peers that point to it is stored. These reverse pointers allow for more efficient location of the peers involved in the query evaluation

Process..

B. Disseminating Data and Index

The distribution process is started by a peer p that is willing to publish a data population, and works as follows. First, for each sub synopsis hj (respectively, leaf portion inf_i), p invokes search(C_{min}) to find C_{min} peers which can host a copy of h_j (respectively, inf_i along with table(inf_i)). Then, for each inf_i and sub synopsis h_j referenced by inf_i, location table table(inf_i) is filled with the IP addresses of the peers which will host h_j. Correspondingly, each h_j is augmented with a reverse pointer to one of the peers which will host inf_i. A similar process is performed to find C_{min} peers which will host I^{sup} along with a location table, and to fill the table as well as the reverse pointers of leaf portions. In

particular, as explained before, the location table of each peer that will host a copy of I^{sup} is filled with the addresses of the other peers which will host copies of I^{sup}. After all of the location tables have been filled, the copies of s-blocks along with their associated location tables are sent to the appropriate peers. It is worth noting that distributing the copies of the s-blocks randomly across the network well suits the search of data in our unstructured scenario, where search will be performed by randomly navigating across the network. At the same time, the information provided by the location tables allows, once an s-blocks related to a data population D is located, to quickly locate all the other s-blocks that are needed to answer queries over D.

IV. RESULTS AND DISCUSSION

We performed several experiments to assess the effectiveness of our approach. Specifically, we studied the accuracy of query estimates and the performance of our replica management strategies in terms of generated network traffic, data reachability, and query performances.

A. Dynamic Replication :

Our dynamic replication scheme aims at both providing the appropriate coverage of s-blocks and balancing the load at the peers. To this aim, besides guaranteeing a minimum coverage for each s-block (so that published data remain accessible over time), our replication scheme provides adaptivity to the dynamic query workload by creating new replicas of an s-block each time it is queried and by removing less queried data through suitable aging policies.

In our framework, location tables encode links among s-blocks spread over the network. Thus, they are kept updated w.r.t. events causing data unavailability by deleting the addresses of the peers that no longer host these data. Our approach is independent of the way the unavailability of data is identified; in practice, this can be done through periodic pinging (as in our prototype) or notification protocols.

After the deletion of some entries in a location table, the system detects whether the minimum coverage (Cmin referenced copies for each s-block) is maintained .

B. Query-Based Replication

We now describe two replication strategies, called path based (PBS) and reactive (RS), that aim at increasing the availability of most queried data, also pursuing load balancing when facing large and dynamic query workloads.

C. Range Queries

The answer of a range query is computed at the requesting peer after receiving the answers of all the (sub)queries submitted to peers hosting data blocks overlapping the query range. The cost of a sub query can be measured from two standpoints, which take into account network-and computation-related costs.

a. Number of hops:

This is at least 1 for a query on I^{sup}, 2 for a sub query on a leaf portion (as this kind of query is generated after one hop for accessing a peer hosting I^{sup} and requires one more hop to reach a peer hosting the appropriate leaf portion), and 3 for a subquery on a subsynopsis (as one more hop is needed). These values are lower bounds, due to peer volatility, data replacement—which yield dangling references—and (in the case of RS) overloading— which triggers the unloading mechanism.

b. Overall wait in queue:

As every (sub)query SQ is enqueued at the peer p' where it will be evaluated, it has to wait for the requests preceding it. The overall wait in queue of SQ is the sum of the enqueuing position of SQ at p^1 and the overall wait in queue of the (sub)query which generated SQ (if any). For instance, if SQ is a sub query on a sub synopsis, its overall wait in queue is the sum of: 1) its enqueuing position at p'; 2) the enqueuing position of the sub query SQ' which generated SQ; and 3) the enqueuing position of the query Q which generated sq'.

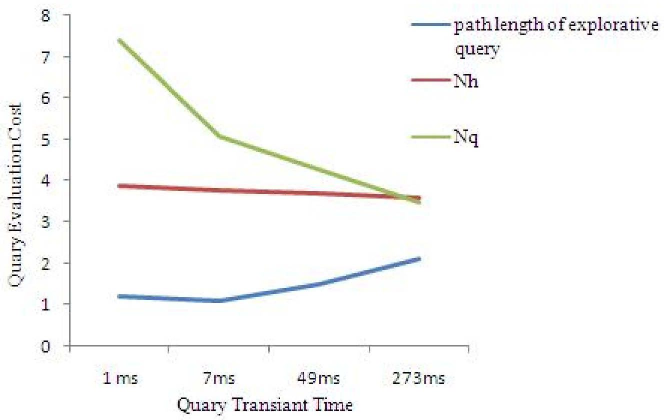
Thus, an upper bound on the overall time needed to complete the evaluation of a range query Q can be obtained by considering the following quantities:

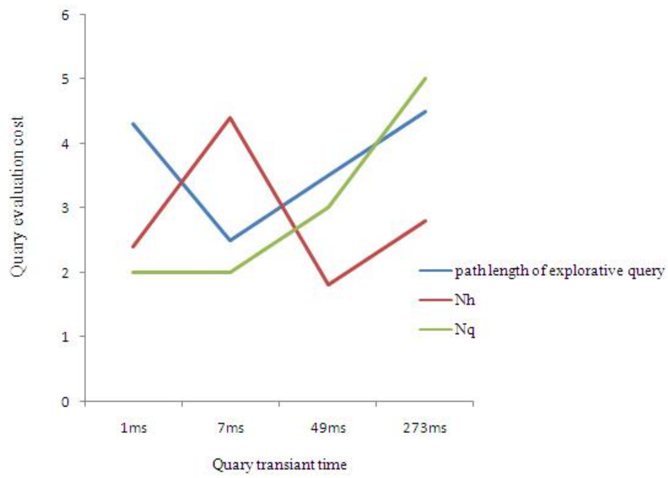
N_h: the maximum number of hops performed to get the answer of a sub query of Q; and

N_q: the maximum overall wait in queue for a subquery of Q.

The diagrams in Fig. 10 depict N_h and N_q versus query frequency for different values of Mt(p). Fig. 1'a shows that as query frequency increases, N_h slightly increases. This can be explained as follows: in the case of PBS, a more intensive query workload yields a more frequent data replacements, which increases the likelihood of finding dangling references, and thus, of performing more hops to reach the needed data. In the case of RS, increasing query frequency causes a larger number of peers to be overloaded when they are called to evaluate queries. Thus, the unloading mechanism is triggered, and requests are forwarded to further peers, thus increasing N_h. The increase in query frequency also negatively impacts on N_q (Fig. 1'b). This effect is less evident with PBS, as compared to RS, the higher coverage allows requests to be distributed among a larger number of peers. As expected, for both N_h and N_q, the behavior of RS depends on $M_t(p)$, as RS saturates queues before making replications: thus, waits in queue get longer as $M_t(p)$ increases (Fig. 1'b), whereas the maximum number of hops for answering a sub query decreases (Fig. 1'a) since the unloading mechanism, yielding the forwarding of query requests to further peers, becomes less frequent as the capacity of queues increases.

The results mentioned above are summarized in the figure 4 and 5 .(in the case $M_t(p)$ = 4), where the cost of explorative queries (in terms of path length per query) is taken into account as well, thus providing an insight on the overall performance of the query answering process in our framework. To summarize, on the one hand, with PBS, sub queries are more likely to be served first, and the number of hops for getting the "slowest" answer of a sub query is slightly lower. On the other hand, with PBS, explorative queries require longer walks over the network to find the needed data, and the network traffic due to the replications needed to support these performances is much larger than that required by RS (as seen previously), thus making RS a much preferable choice.





CONCLUSION AND FUTURE WORK

As the importance of peer to peer network is increasing, the data shared in network to be stored and retrieved very efficiently .we propose this framework to manage the multidimensional data. The data is shared and retrieval in unstructured p2p network. The people who are interested in sharing their data , make their resources available for all peers in network .So that they can access data by posing range queries .We adopt mechanism for data summarization ,data indexing and data distribution and replication by preserving autonomy of peers .this experiment proves fast and accurate query answers and ensuring the robustness .

Future work: adopting these mechanisms to other aggregate operators rather sum. And need to devise suitable compression, indexing techniques and data distributing techniques for better robustness assurance in the network.

REFERENCES

[1] Giuseppe Massimiliano Mazzeo, and Andrea Pugliese Filippo Furfaro, "Managing Multidimensional Historical Aggregate Data in Unstructured P2P Networks," *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, vol. 22, no. 9, pp. 1313-1330, September 2010.

[2] V. Poosala, and S. Ramaswamy S. Acharya, "Selectivity Estimation in Spatial Databases," in *ACM SIGMOD*, 1999.

[3] A. Andrzejak and Z. Xu, "Scalable, Efficient Range Queries for Grid Information Services," in *Proc. Second Int'l Conf. Peer-to-Peer*, 2002.

[4] M. Mihail, and A. Saberi C. Gkantsidis, "Random Walks in Peerto-Peer Networks," in *Proc. 23rd IEEE INFOCOM*, 2004.

[5] S. Chaudhuri and U. Dayal, "An Overview of Data Warehousing and OLAP Technology," *Sigmod Record*, vol. 26, no. 1, pp. 65-74, mar 1997.

[6] M. Demirbas and H. Ferhatosmanoglu, "Peer-to-Peer Spatial Queries in Sensor Networks," in *Third Int'l Conf. Peer-to-Peer Computing*, 2003.

[7] http://www.gnutella.com, 2008.

[8] G. Das, D. Gunopulos, and V. Kalogeraki B. Arai, "Approximating Aggregation Queries in Peer-to-Peer Networks," in *Proc. 22nd Int'l Conf. Data Eng*, 2006.

[9] A. Gupta, D. Agrawal, and A. El Abbadi O.D. Sahin, "A Peer-to-Peer Framework for Caching Range Queries," in *Proc. 20th Int'l Conf Data Eng*, 2004.

[10] A. Krishnamurthy, and R.Y. Wang C. Zhang, "Skipindex:Towards a Scalable Peer-to-Peer Index Service for High Dimensional Data," Princeton Univ, Technical Report TR-703-04, 2004.

[11] UCI KDD Archive. http://kdd.ics.uci.edu, 2010.

[12] (2008) http://setiathome.ssl.berkeley.edu.

[13] A. Gupta, D. Agrawal, and A. El Abbadi O.D. Sahin, "A Peer-to-Peer Framework for Caching Range Queries," in *Proc. 20th Int'l Conf.*, 2004.