Aggregation Query Processing in P2P Networks

Ratnmala Bhimanpallewar and Pravin Metkewar

Department Of Computer Engg, Pune University, VIIT, Kondhawa, Pune
Symbiosis Institute of Computer Studies and Research, Pune
Maharashtra, India
ratnmalab@gmail.com, pravin.metkewar@sicsr.ac.in

Abstract

Peer-to-peer (P2P) databases are becoming prevalent on the Internet for distribution and sharing of documents, applications, and other digital media. The problem of answering large-scale ad hoc analysis queries, for example, aggregation queries, on these databases poses unique challenges. Exact solutions can be time consuming and difficult to implement, for the distributed and dynamic nature of P2P databases.

In this paper, we have presented novel sampling-based techniques for approximate answering of ad hoc aggregation queries in such databases. Computing a high-quality random sample of the database efficiently in the P2P environment is complicated due to several factors: the data is distributed (usually in uneven quantities) across many peers, within each peer the data is often highly correlated and moreover, even collecting a random sample of the peers is difficult to accomplish. To formulate these problems, developed software uses approach, based on random walks of the P2P graph, as well as block-level sampling techniques. We have presented here extensive experimental evaluations to demonstrate the feasibility of our solution.

The modules that are included in this project are:

1. Peer-to-Peer Node Construction

2. Random Selection of Node

3. Selection of Records

4. Performance Evaluation

Keywords: *Peer-to-peer (P2P) databases, Approximation, Aggregation, Distributed databases, Query processing, Distributed systems*

1. Introduction

Real-time databases are distributed over the entire network. Different nodes in network are called as peers in P2P network. In most of the real-time database applications we have to deal with Aggregation queries like SUM, AVG etc. While executing aggregation queries, execution time is one of the important factors to speed up the Query processing in distributed environment. Here Query is fired from single node called as Query-node to all peers (i.e. Database servers) query is processed at individual peer and only result is sent back to Query node to compute final aggregation result .Our approach is to implement AQP algorithm in such a way that it will give the result with desired accuracy in minimal time. We have used sampling method to retrieve data from peers.

The database is distributed in n-peers. The problem concerns database applications in P2P systems is: Increasingly sophisticated e-business and different applications have vast amount

of data within P2P databases. Aggregation queries have the potential of supporting applications in decision support, data analysis, and data mining. One of the challenges is: 'How aggregation queries on such databases can be answered '. This challenge is not answered yet.

Consider that required data is distributed over a P2P system, that is, the peers store horizontal partitions (of different sizes) of this database. An aggregation query may be introduced at any peer (here on query node). Although aggregation queries have been heavily tried in traditional databases, but those techniques cannot be easily adapted in the P2P domain. For example, decision support techniques such as online analytical processing (OLAP) commonly employ materialized views. But the distribution and management of such views appear difficult in a dynamic and decentralized domain. One of the alternatives of answering aggregation queries at runtime is "from scratch". Aggregation queries are executed here by crawling and scanning the entire P2P repository.

2. Problem Definition

Although all these goals can be achieved with alternative approaches, still e-commerce is not that much efficient and there are still some problems remain unanswered.

They include:

Need of fast computation: When we deal with real-time database applications user often expect approximate answers mostly very few applications need accurate answers. This approximation should within some limit with allowed error only. Time is the important factor in real-time applications, to feel the aliveness. So resultant computation should be fast enough.

Execution of aggregation Queries: While performing large transactions in real-time database applications mostly we need to handle aggregation queries (like SUM, AVERAGE, COUNT etc) along with simple queries. There are many approaches to faster the execution of simple queries but achieving fast execution of aggregation queries is a unique challenge.

Dynamic nature of P2P database and aggregation queries: P2P databases are becoming efficient for distribution and sharing documents, applications and other digital media. P2P database is popular as it shows dynamic and distributed nature. Computing solutions to aggregation queries on this database can be time consuming and difficult to implement.

Random-walk: We cannot introduce query on every node in peer-network as it will be time consuming. So to answer the queries we need to select some peer-nodes at random in order to speed-up the query processing. As adjacent peers may passes correlated database, no meaning in visiting adjacent peers. We will skip some in-between nodes on the basis of some logic. The logic should be developed in such a way that it will select nodes having probably asymmetric database which will be sufficient to compute result. An efficient random-walk algorithm can fulfill both of above requirements.

A novel sampling based technique: After selecting nodes randomly we cannot consider whole database for query execution, because data at each node is often highly correlated and main thing is that each peer-node has database of different size. Peer-nodes having large database will take more time to answer the query. To keep uniformity in execution time proper way is to fix the sample size in terms of number of tuples and select that fix sized sample for query processing from existing database.

Computing high quality random samples: Though we fix sample size, but in what way it should be selected from existing database. One approach is to select it randomly. But what is the guaranty that selected samples are high quality samples. Collecting high quality random samples in first scanning is unstructured distributed database of peer-network is difficult to accomplish. To counter this problem we need to verify the experimental results of first scan. If result is satisfactory we are lucky otherwise change input parameters (like sample size, jumping parameter etc.) such that we will get high quality samples surely. The exact problem addressed here is to design a system which will cater to the needs mentioned above.

3. Literature Survey

P2P systems are becoming very popular because they provide an efficient mechanism for building large scalable systems [6]. Most recent work has focused on Distributed Hash Tables (DHTs) [7]. Such techniques provide scalability advantages over unstructured systems (such as Gnutella); however, they are not flexible enough for some applications, especially when nodes join or leave the network frequently or change their connections often. Recent work has proposed different techniques for exact query processing in P2P systems. Most proposals use structured overlay networks (DHTs), such as CAN, Pastry and Chord. A hybrid system, Mercury [2], using routing hubs to answer range queries was also recently proposed. Methods to sample random peers in P2P networks have been proposed in [4] and [2]. These techniques use Markov-chain random walks to select random peers from the network. Their results show that when certain structural properties of the graph are known or can be estimated (such as the second eigenvalue of the graph), the parameters of the walk can be set so that a representative sample of the stationary distribution can be collected with high probability. In [2] it is shown that if the graph is an expander, then a random walk converges to the stationary distribution in $O(\log M)$ steps, where M is the number of peers in the network. There are known techniques for computing approximate aggregates in distributed settings (most notably, the Gossip protocol [3] and [5]). The technique works generally as a preprocessing step where all peers in a network attempt to mix data among adjacent peers, eventually converging upon a single value. The inability to contact all nodes in the network makes it exceedingly difficult to Gossip in the traditional sense.

Our work also generalizes to the P2P domain and previous work on AQP in relational databases. Recent work in [4] has developed powerful techniques for employing sampling in the database engine to approximate aggregation queries and to estimate database statistics. Recent techniques have focused on providing formal foundations and algorithms for block-level sampling and are thus most relevant to our work. Recent technique provides the results for hybrid Approach with simulator [1] which we have actually implemented in our paper.

4. Our Approach

All traditional methods for aggregation queries gives slow response and as we know the time and interactivity are more important factors than accuracy in many applications. Our approach gives fast response and gives approximate answer to the introduced aggregation query called as Approximate Query Processing (AQP). An efficient algorithm is introduced in this paper. Standard theorems are provided.

4.1 AQP Phases

Input:

Δ_{req} - Maximum allowed error.

t- Number of tuples to be sampled per peer.

j- Jump size.

Phase-I:

Execute the query introduced at Query node using given random sampling algorithm. Estimate both result and error. Compare estimated error with Δ_{req} and find the parameter

m' - (more) number of peers required to visit for desired output.

Phase-II:

Execute query on m' number of peers and get desired output.

Our approach works for standard SQL aggregation operators like SUM, COUNT, AVG, and so on and also can be extended for interesting statistical estimators. (Only COUNT is discussed in detail).

The Goal of this approach is, Approximating Aggregation Queries in P2P Networks. Given an aggregation query and a desired error bound at a query node peer, compute with "minimum cost" an approximate answer to this query that satisfied the error bound.

Where,

Δ_{req} - is the given desired error bound, varies as per application or need.

4.2 Basic Terms

Peer-to-Peer (P2P) Databases: A P2P network consists of numerous peer nodes that share data and resources with other peers on an equal basis.

Aggregation Queries: SELECT Agg-Op(Col) FROM T WHERE selection-condition

Agg-Op: any aggregation operator such as SUM, COUNT, AVG etc.

4.3. Challenges

Two main problems in P2P Network: We need uniform random samples from peers to get proper data for approximate result.

4.3.1 Picking a set of uniform random peers is a difficult problem.

4.3.2 Even if we could select a peer (or a set of peers) uniformly at random, it does not make the problem of selecting a uniform random set of tuples much easier.

5. Research Methodology

5.1 Unstructured P2P Network

An unstructured P2P network is formed when the overlay links are established arbitrarily. Such networks can be easily constructed as a new peer that wants to join the network can copy existing links of another node and then form its own links over time. In an unstructured P2P network, if a peer wants to find a desired piece of data in the network, the query has to be flooded through the network to find as many peers as possible that share the data. The main disadvantage with such networks is that the queries may not always be resolved. Popular content is likely to be available at several peers and any peer searching for it is likely to find the same thing. But if a peer is looking for rare data shared by only a few other peers, then it is highly unlikely that search will be successful. Since there is no correlation between a peer

and the content managed by it, there is no guarantee that flooding will find a peer that has the desired data. Flooding also causes a high amount of signaling traffic in the network and hence such networks typically have very poor search efficiency. Many of the popular P2P networks are of unstructured type.

We have abandon trying to pick true uniform random samples of the tuples, as such samples are likely to be extremely impractical to obtain. Instead, we consider skewed samples, provided that we can accurately estimate the skew during the sampling process. To get the accuracy in the query answer, our skewed samples can be larger than the size of a corresponding uniform random sample that delivers the same accuracy and they are much more cost efficient to generate. The main point that system parameters are relatively slow to change and thus do not have to be estimated at query time but the data contents of peers that changes more rapidly; hence, the random sampling process has to be done at runtime.

5.2. Query Processing

Graph :

The network design is considered as a graph as fallows

$G = (P, E),$

vertex set $P = \{p_1, p_2, \ldots, p_M\}$

E- Set of Edges in graph

$\deg(p)$ - in-degree + out-degree of peer p

$$prob(p) = \frac{\deg(p)}{2|E|}.$$

P2P graph are modeled as an $M \times M$ matrix

j- Jump parameter, determines in random peer selection how many peers can be skipped between selection of two peers.

Phase 1:

Here, we initiate a fixed-length random walk from the query node. This random walk should be long enough to ensure that the visited peers represent a close sample to retrieve certain information.

This information is then analyzed at the query node to determine the skewed nature of the data that is distributed across the network, such as the variance of the aggregates of the data at peers, the amount of correlation between tuples that exists within the same peers, the variance in the degrees of individual nodes in the P2P graph, and so on as fallows.

t - A fixed no of tuples to be selected from each selected peer. [If peer has t tuples consider entire data otherwise, from whole database consider t tuples randomly].

Note: If data at peers is highly correlated then number of peers to be visited are increased i.e. j value is decreased.

Δ_{req}- desired error threshold already specified.

CVError- cross validation error.

Example:

Practical approach for Aggrgation query COUNT:

Δ_{req} - desired error threshold

already specified.

$P = \{p_1, p_2, \ldots, p_M\}$ is the set of peers.

For a tuple u,

if u satisfies the selection condition, $y(u) = 1$

otherwise $y(u) = 0$

aggregate for a peer p

$y(p) = \sum_{u \in p} y(u)$. (For count)

exact answer for the query

$y = \sum_{p \in P} y(p)$

y' - estimated count by our algorithm

Then,

$$|y - y'| \leq \Delta_{req}.$$

Let there are m samples from m peers,

$$S = \{s_1, s_2 \ldots s_m\}.$$

Random-walk applied in m rounds then result is,

Random-walk applied in m rounds then result is,

$$y'' = \frac{\sum_{s \in S} \frac{y(s)}{prob(s)}}{m}.$$

Expected result is

$$E[y''] = y,$$

For m=1,variance is defined as,

$$C = \sum_{p \in P} \left(\frac{y(p)}{prob(p)} - y \right)^2 prob(p).$$

$$Var[y''] = C/m.$$

Divide m peers into two m/2 parts,

Cross validation error is,

$$CVError = |y_1'' - y_2''|.$$

Expected Cross validation error is,

$$E[CVError^2] = 2E[(y'' - y)^2].$$ ……(proved).

For result with desired accuracy, more no peers to be visited are m'

$$m' = (m/2) * \left(\frac{CVError^2}{\Delta^2_{eq}}\right)$$

Same query is repeatedly fired in same P2P network, at some stage we will get accurate parameters m' and t, which needed for Phase-II. Thus first phase may recommend that the best way to answer this query is to visit m' peers and, from each peer, randomly sample t tuples.

Phase 2:

Visit m' peers; we will get result with desired accuracy. A random walk is reinitiated, and tuples are collected according to the recommendations made by the first phase.

The aggregate value is computed at each peer, the local aggregates at all visited peers are returned to the query node, which are then composed into a final answer.

In a network for visiting peers randomly Gossip protocol can be used. Gossip protocol is executed in rounds. For each round, participating peers select adjacent peers uniformly at random sharing information. The Gossip protocol exploits a communication mechanism where peers diffuse local aggregates with adjacent peers. In this the average of all of the sums of individual peers is the correct average, and the sum of all of the weights is n as the number of passes of the Gossip protocol increases, values of participating peers are increasingly diffused through the network therefore, sampling-diffused values provide a better representation of the values contained in the network as compared to a single peer.

Key issues are:

We introduce the important problem of AQP in P2P databases, which is likely to be of increasing significance in the future.

The problem is analyzed in detail, and its unique challenges are comprehensively discussed.

Hybrid sampling technique maximizes per-peer in network computation building upon the Gossip protocol.

Adaptive two-phase sampling-based approaches are proposed based on well-founded theoretical principles.

6. Random-walk Algorithm

The following algorithm shows the complete procedure for random walk in Peer-to-Peer network and efficient query processing. This algorithm is implemented in Java frame work. This algorithm is used for selection of active peers in network randomly and execution of query on selected peers.

Predefined Values:

M : Total number of peers in a network

m : Number of peers to visit

j : jump size for random walk

t : max tuples to be sub sampled per peer

Inputs:-

Q : Query with selection condition

Sink: Peer where query is initiated

Output: - Query result to Sink(Peer where query is initiated)

1. Start
2. Check number of active nodes
3. If number of peers = 1
   3.1. Execute query on that peer
4. Else perform random walk
5. Curr = Sink; Hops = 1;
6. While (Hops < j * m) {
7. If (Hops % j)
8. Visit (Curr);
9. Hops ++;
10. Curr = random adjacent peer
11.}
12. Visit (Curr){
13. If (# tuples of Curr) <= t){
14. Execute Q on all tuples
15. Else
16. Execute Q on t randomly sampled tuples
17.}
18. Return result to sink
19. Compute Error rate, Exact Processing time, Approximate Processing time
20. Return this result to Sink
21. End

Determination of Error Rate:

For a tuple u,

Let,

y(u)=1 if u satisfies the selection condition, and y(u)= 0 otherwise.

Let the aggregate for a peer p be

y(p) = $\sum_{u \in p} y(u)$.

Let y be the exact answer for the query, that is,

y(p) = $\sum_{u \in p} y(u)$. T

The query also comes with a desired error threshold Δreq .

The implication of this requirement is that if y' is the estimated count by our algorithm, then

|y- y' |<= Δreq

7. Proposed Modules

Module1: Peer-to-Peer Node Construction

Module2: Random Selection of Node

Module3: Selection of Records

Module4: Performance Evaluation

Module 1: Peer-to-Peer Node Construction:

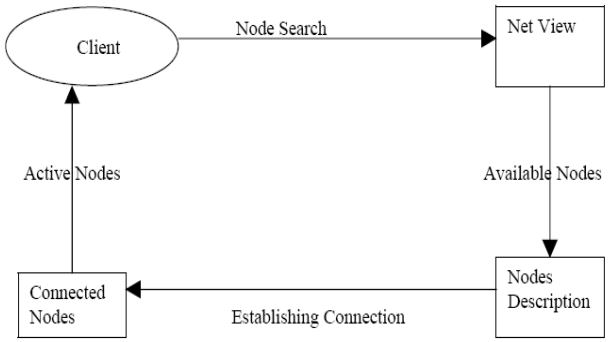

Figure 1. Peer-to-Peer Node Connections

Above figure shows Peer to peer node connection. Client node is one from which the user makes the query to process from other node called servers (Database servers). Simply query-processor is termed as client. This will capable to act as a Distributed Model. The server has maintained all information about the client. In this module client search for all nodes in network and get the names of nodes those are active nodes.

Module 2: Random Selection of Nodes:

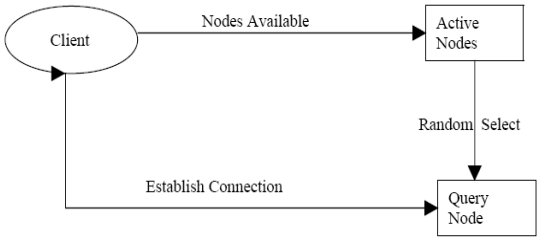

Figure 2. Random Selection of Active Node

From all active peers select some of active peers randomly. This is done using random walk algorithm. It is well known that if this walk is carried out long enough, then the eventual probability of reaching any peer p will reach a stationary distribution. To make this more precise,

Let ,

P = {p1; p2; . . . ; pM} be the entire set of peers,

E- be the entire set of edges,

deg(p)- the degree of a peer p

Then, the probability of any peer p in the stationary distribution is

Prob (p) = deg(p)/2|E|

Module 3: Selection of Records:

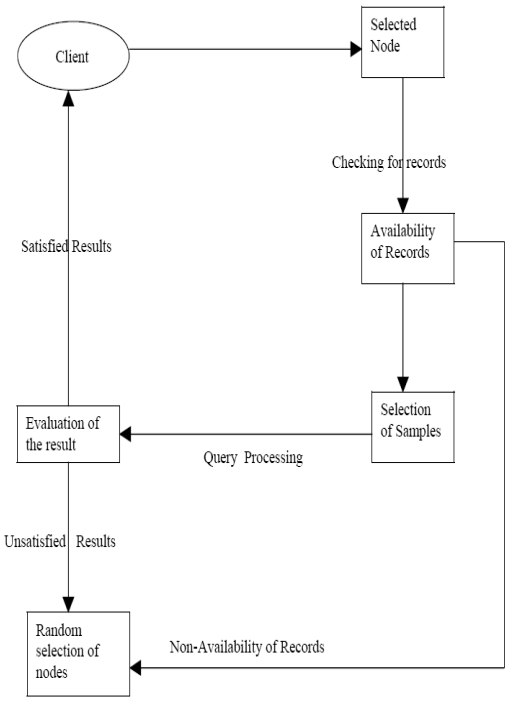

Figure 3. Selection of Records

Consider a fixed-size sample of peers S (s1, s2 . . . sm); where each si is from P. This sample is picked by the random walk in the first phase. We can approximate this process as that of picking peers in m rounds, where in each round, a random peer si is picked from P, with probability prob(si). We also assume that peers may be picked with replacement; that is, multiple copies of the same peer may be added to the sample, as this greatly simplifies the statistical derivations below.

$$Y = E(y(s)/prob(s))/m$$

Where m is the number of peers visited.

Module 4: Performance Evaluation

Query node results are compared with error bound and execution time is noted.

Error-bound varies as per application Ex. In Astronomical applications accuracy is more important than time so the error-bound is minimum.

8. Experimental Evaluation

This section provides experimental justification for our methods. We have implemented our algorithms on real-world topologies by using various degrees of data clustering and topology structures.

Our algorithm is implemented in JDK 1.6.0_10 , JRE 6 with SQL Server 2005. Our implementation includes both static and dynamic samples. All of our experiments were run on Intel Core 2 Duo / AMD 2.6 GHz processors with 3 GB of RAM. We generate P2P network on the basis of j-jump size parameter. It can be changed as per the need of application.

8.1. Evaluation Metrics

Our algorithms are evaluated based on the cost of execution and how close they get to the desired accuracy. As discussed earlier, we use latency as a measure of our cost, noting that in our case, it is proportional to the number of peers participating in the random walk. In fact, if the number of tuples to be sampled is the same for all peers (which is true in our experiments), then latency is also proportional to the total number of sample tuples drawn by the overall algorithm. Thus, we use the number of sample tuples as alternative for latency in describing our results.

8.2 Input Parameters

We evaluate the accuracy, the use of network resources, the size of sample acquired, and the total number of tuples sampled from the network. We define each of the user defined inputs as follows:

1. Required Accuracy (Δreq): This parameter defines the maximum allowed error for the estimated answer.

2. Tuples Sampled per Peer (t): This parameter defines the number of tuples to be sampled from each selected peer.

3. Jump Size (j): This parameter defines the number of peers to be passed over before selecting the next peer for sampling.

4. Initial Sample Size (R): This parameter defines the initial number of tuples to be acquired from the database to execute the first phase. (Thus, R=t * m) where m is the number

of peers visited in the first phase. In our experiments, the local databases are always large enough to ensure that sub sampling always takes place.

Parameter 1 is provided by the user for each query.

Parameters 2-4 may be provided by the user or may be set via a preprocessing step.

Parameters 1-4 all are dependent on Application.

9. Experiments

This section presents an extensive experimental evaluation. After successful execution of program which is implemented in java following Login window appear on screen shown in Figure 4.

9.1. P2P Databases

We have used single-attribute tuples.

Sql Server 2005/2000 is used for database creation. In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMS. This consistent interface is achieved through the use of "plug-in" database connectivity modules, or drivers. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on it. To gain a wider acceptance of JDBC, Sun based JDBC's framework on ODBC.

9.2. A Typical model for AQP in P2P Network

Query Processing in P2P network is achieved through RMI technique and Java Swing as API. Query Processing is on java file contain 'main' function from where the execution of application starts. Before that Server should be started. After successful execution following window will appear on screen.

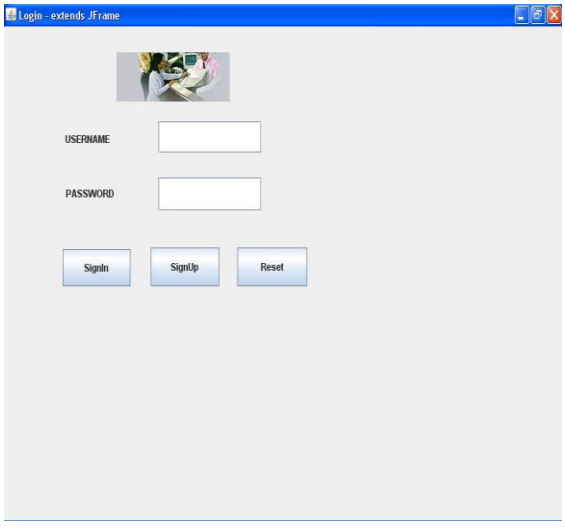

Figure 4. Login Window

After successful login, user is ready for execution of Approximate Query Processing in peer to peer network. We can use from 10 to 1000 nodes for processing.

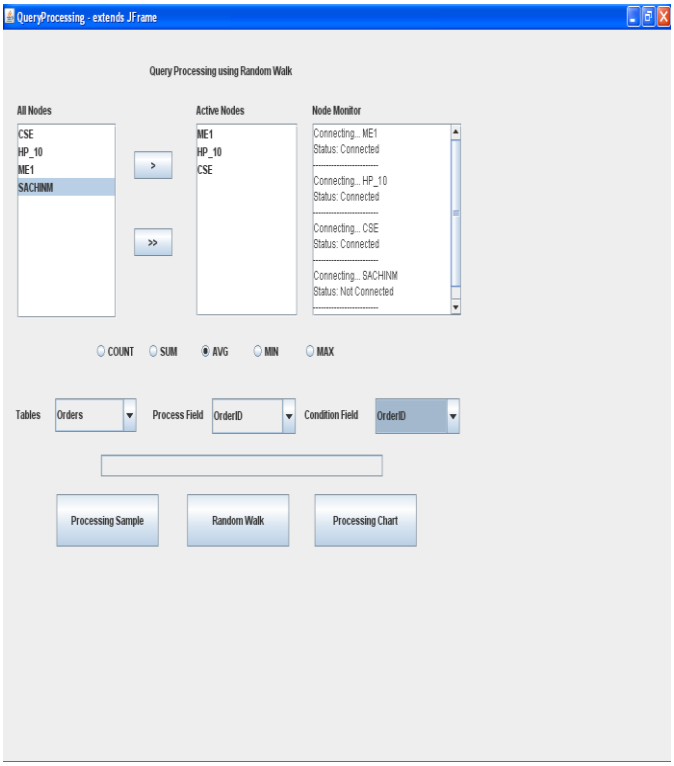

Figure 5. Approximate Query Processing Window

Figure 5 shows list of all peers in Peer to Peer network. This list is present in text area 'All Nodes'. CSE, HP_10, ME1, SACHINM are the peers in network. From this list of peers CSE, HP_10, ME1 are the active peers. These active peers are listed in 'Active Nodes' text area above. Text area 'Node Monitor' is status of peers Connected or Not Connected. Radio Buttons COUNT, AVG, SUM, MIN, MAX are used for selection of aggregation queries. Following is format and explanation of COUNT and MAX query in Sql.
Create tables in database using Sql ex.:

Table 1. Supplier

Supplier_ID	Supplier_Name	State
1	IBM	CA
2	Microsoft	US
3	NVIDIA	UK

1. COUNT :The COUNT function returns the number of rows in a query. The syntax for the COUNT function is:

SELECT COUNT (expression) FROM tables WHERE predicates;

The COUNT function will only count those records in which the field in the brackets is NOT NULL.

For example,

1.If you have the following table called suppliers:

Select COUNT(Supplier_ID) FROM suppliers;

The result for this query will return 3.

2.MAX: The MAX function returns the maximum value of an expression.

The syntax for the MAX function is:

SELECT MAX(expression) FROM tables WHERE predicates;

Simple Example

For example, you might wish to know the maximum salary of all employees.

SELECT MAX (Supplier_Id) FROM Suppliers;

The result of this query will return 3.

User selects the query which is presented by radio buttons in figure 5. After selection of query that user wants to fire, select 'Processing Sample' button. After clicking this button text area 'Table' will show the list of tables present in the database. Select table name from that list on which query has to process. Then select Process Field and Condition Field for that particular table. Finally click button Random Walk which will display the result of query fired by user with following window.

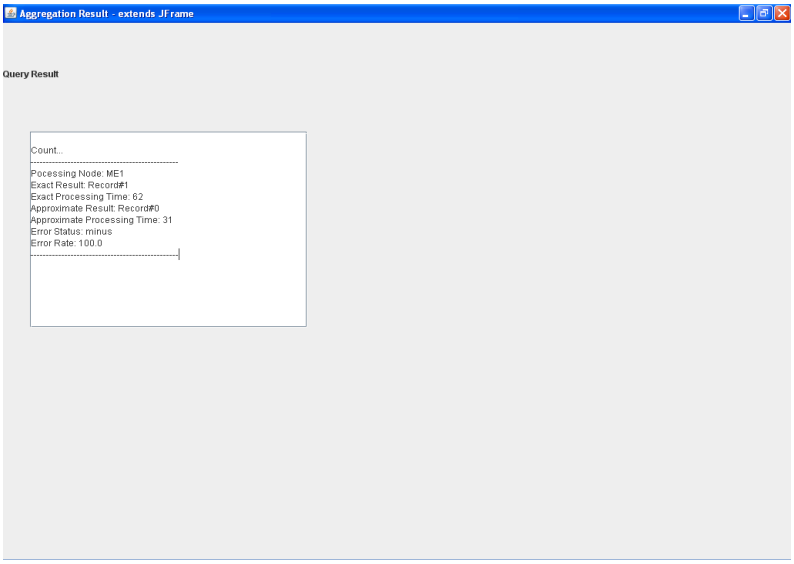

Figure 6. Result of Approximate Query Processing

Text Area in Figure 6 window shows the result of users query. It contains the processing node which is selected randomly by random walk algorithm. It indicates Exact Processing time, Approximate Processing time, Error Status and Error rate.

Graphical representation of Exact Processing time and Approximate Processing time can be shown by clicking Processing Chart button which is shown in Figure 7.

Following window shows processing chart and exact processing time is 63ns and Approximate processing time is 31ns.

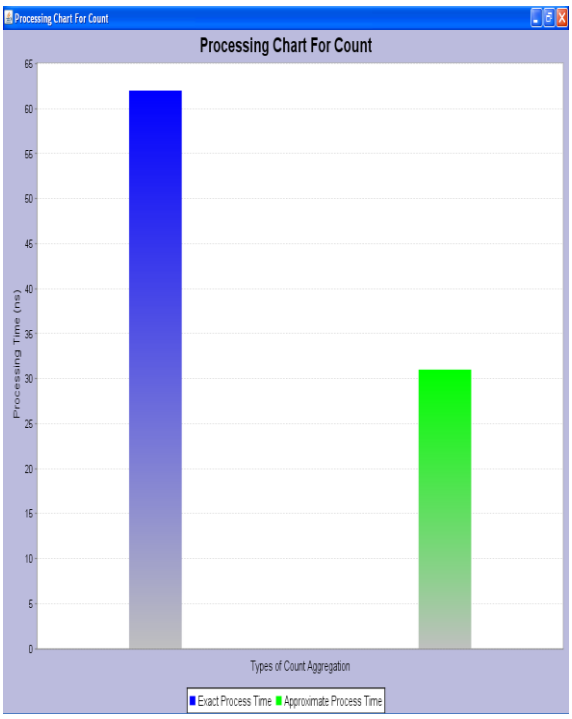

Figure 7. Processing Chart

9.3. Graph Plots

Synthetic topology is created through the process of connecting sub-graphs by using the graph generation tool. It consists of 10,000 peers and 100,000 edges.

Parameters are defined as follows

Cluster Level (CL):

If CL = 0 data is perfectly clustered otherwise randomly permuted.

Skew (Z):

The skew determines the slant in frequency distribution of distinct values in data.

Jump Size (j):

Number of peers to be passed before selection of next peer for sampling.

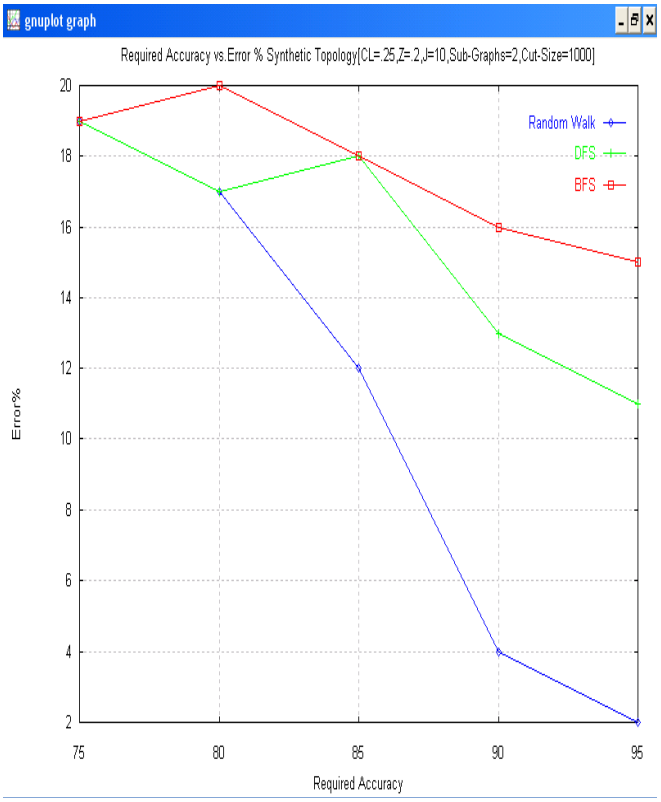

Figure 8. Random Walk Perform Better Performance than BFS and DFS

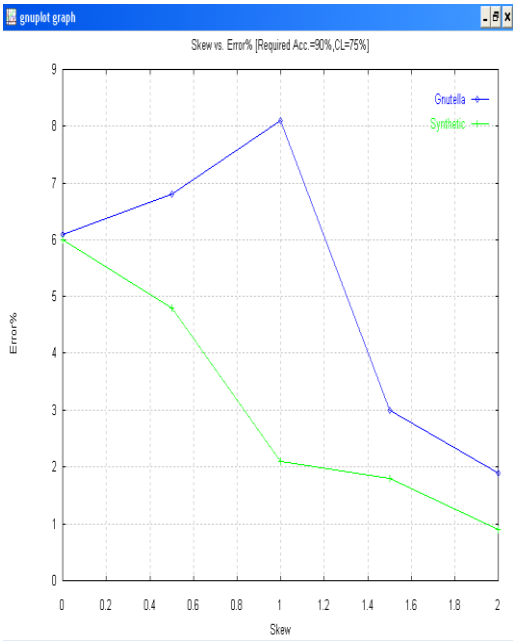

Figure 9. Effects of Clustering on Error % for SUM Technique

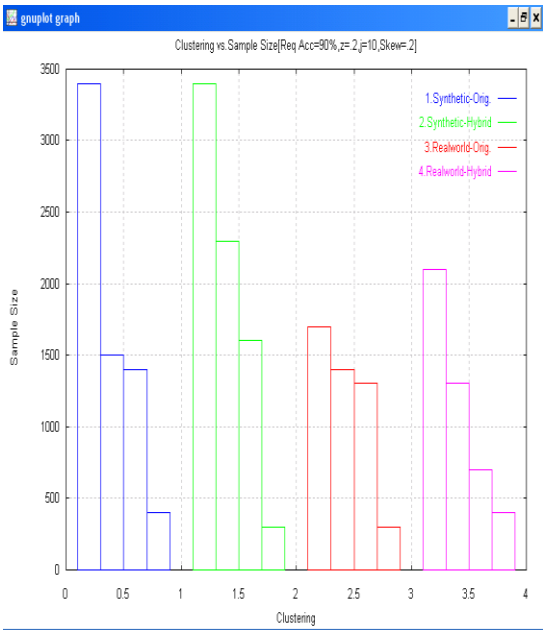

Figure 10. Effects on Clustering on Samples Size for the COUNT Technique

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

10. Conclusion

AQP presents adaptive sampling-based techniques for the approximate answering of adhoc aggregation queries in P2P databases. Our approach requires a minimal number of messages [5] sent over the network in order to achieve higher quality of service, efficient resource utilization, increased productivity, reduced business risks. It also tracks the business opportunities. By using this approach efficiency can be improved in this dynamic and competitive business environment.

This project provides a powerful technique for approximating aggregates but comes with limitations based upon structure and connectivity of given topologies. By varying a few parameters, our algorithm successfully computes aggregates within a given required accuracy. The extensibility feature helps to dynamically add new nodes as per the changing business needs. We presented here extensive experimental evaluations to demonstrate the feasibility of our solutions.

References

[1] B. Arai, G. Das and V. Kalogeraki, "Efficient Approximate Query Processing in Peer-to-Peer Networks", IEEE Transactions on Knowledge and and Data Engineering, vol. 19, no. 7, **(2007)** July.

[2] A. R. Bharambe, M. Agrawal and S. Seshan, "Mercury: Supporting Scalable Multi-Attribute Range Queries", Proc. ACM Ann. Conf. Applications, Technologies, Architectures, and Protocols for Computer Comm. (SIGCOMM '04), **(2004)**.

[3] S. Boyd, A. Ghosh, B. Prabhakar and D. Shah, "Analysis and Optimization of Randomized Gossip Algorithms", Proc. 43rd IEEE Conf. Decision and Control (CDC '04), **(2004)**.
[4] C. Gkantsidis, M. Mihail and A. Saberi, "Random Walks in Peer-to-Peer Networks", Proc. IEEE INFOCOM '04, **(2004)**.
[5] D. Kempe, A. Dobra and J. Gehrke, "Gossip-Based Computation of Aggregate Information", Proc. 44th Ann. IEEE Symp. Foundations of Computer Science (FOCS '03), **(2003)**.
[6] D. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins and Z. Xu, "Peer-to-Peer Computing", HP Technical Report HPL-2002-57, **(2002)**.
[7] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker, "A Scalable Content-Addressable Network", Proc. ACM Ann. Conf.

Authors

Miss Ratnmala N. Bhimanpallewar

She has completed M.E. in Computer Engineering from PICT College under Pune University. She has 2 years of experience as an Assistant Professor.

Dr. Pravin Metkewar

He has received his Ph.D. degree from SRT University, Nanded in computer science under the faculty of science in Aug 2005. He has 14 years of experience in the field of teaching, RnD (research) centre and industry. His specialization in Information Systems, Neural networks, OOAD and UML. He has presented and published 15 research papers and 1 book in his account. He is a research guide of Symbiosis International University, Pune. He is a CSI life member.