Journal of Computer Science 10 (7): 1186-1196, 2014
ISSN: 1549-3636
© 2014 Science Publications
doi:10.3844/jcssp.2014.1186.1196 Published Online 10 (7) 2014 (http://www.thescipub.com/jcs.toc)

AN EFFICIENT HYBRID ALGORITHM TO REDUCE LATENCY IN AD-HOC AGGREGATION

Rajkumar, N., N.K. Senthil Kumar, P. Subha and R. Kavitha

Department of CSE, Vel Tech Dr. RR and SR Technical University, Avadi, India

Received; 2013-11-16 Received; 2013-11-29, Accepted; 2014-02-18

ABSTRACT

A data warehouse is a collection of data gathered and organized so that it can easily be analyzed, extracted, synthesized and also be used for the purpose of further understanding data. Peer to Peer networks are used for distribution and sharing of documents. In traditional techniques, when aggregate functions like average, sum and count are encountered, the aggregate operation is performed by considering all the nodes and tuples, which reduces the efficiency of the query processing system. Exact solutions can be time consuming and difficult to implement, given the distributed and dynamic nature of P2P networks. The problem is overcome in this project by selecting random peers and random tuples from P2P networks and then performing the aggregation operation, thus the speed is increased and latency is reduced. Though accuracy is compromised to small extent, efficiency is achieved. Thus, this kind of approximate query processing will be beneficial to the areas where efficiency plays a main role than accuracy. Adaptive Hybrid approach based on random walk is used to achieve the efficiency in the performance of aggregation operation.

Keywords: Aggregate Function, Peer to Peer Networks, Distributed Databases, Distributed Database Query Processing and Gossiping

1. INTRODUCTION

1.1. Peer-to-Peer Database

A P2P network consists of several peer nodes that share data and resources with other peers on an equal basis. The most gripping applications on P2P systems to data have been file sharing and retrieval. For example, P2P systems such as Napster (2006); Gnutella (2006); Kazaa (2006); Freene (2006) are mainly known for their file sharing capabilities.

1.2. Aggregation Queries

The huge amount of data within P2P databases creates a different challenge that has not been adequately researched thus far, that is, how aggregation queries on such databases can be replied. Aggregation queries have the possible wrong applications in decision support, data analysis and data mining. Sensor networks (Zhao *et al*., 2003) can directly profit from aggregation of traffic analysis data by contribution a more efficient means of computing various network-based aggregates such as the average message size and maximum data throughput within the network, with smallest energy consumption and reduced response times. We make the problem more precise as follows.

Consider a single table T that is distributed over a P2P system; that is, the peers store horizontal partitions (of varying sizes) of this table. An aggregation query such as the following may be initiated at any peer.

Aggregation query: SELECT Agg-Op(Col) FROM T WHERE selection-condition In the above query, the Agg-Op may be any aggregation operator such as SUM, COUNT, AVG and so on, Col may be any numeric measure column of T or even an expression involving multiple columns and the selection condition decides which tuples should be involved in the aggregation.

Corresponding Author: Rajkumar, N., Department of CSE, Vel Tech Dr. RR and SR Technical University, Avadi, India

Although aggregation queries have been deeply investigated in traditional databases, it is not clear that these techniques will simply adapt to the P2P domain.

1.3. Approximate Query Processing (AQP)

A variety of AQP (Babcock *et al*., 2003) techniques have been developed:

The most accepted ones are based on random sampling, where a small random sample of the rows of the database is drawn, the query is executed on this small sample and the results are extrapolated to the complete database. In addition to effortlessness of implementation, random sampling has the forceful advantage that, in addition to an estimate of the aggregate, one can also offer confidence intervals of the error, with high probability.

generally, two types of sampling-based approaches have been considered: (1) precomputed samples, where a random sample is precomputed by scanning the database and the same sample is reused for several queries and (2) online samples, where the sample is drawn "on the fly" upon encountering a query.

1.4. Existing System

The existing system requires aggregation of large volume of data. It involves scanning of entire P2P repository as shown in **Fig. 1**. As there is no centralized storage it is not possible to store the precomputed samples. It is time consuming and difficult to implement when the P2P databases are of dynamic nature. It focuses on specific application domains and ignores the semantics of data. The process is also resource intensive and expensive since it considers all the peers and all the tuples in it.

1.5. Proposed System

Approximate Query Processing (AQP) is introduced in P2P databases to overcome the limitations of existing system by improving the efficiency. Hybrid algorithm is used which involves two phases with gossiping thereby selecting peers randomly and then random selection of tuples is done. Finally cross validation is performed. Thus the latency is reduced sacrificing the accuracy to some extent.

1.6. Two Phase Sampling Approach

Our approach has two major phases (Das *et al*., 2007). In the first phase, 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 from the underlying stationary distribution (the appropriate length of such a walk is determined in a preprocessing step) (Gribble *et al*., 2001; Gupta *et al*., 2002). We then retrieve certain information from the visited peers, such as the number of tuples, the aggregate of tuples (for example, SUM, COUNT, AVG and so forth) that satisfy the selection condition and send this information back to the query node. 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 (recall that the degree has a bearing on the probability that a node will be sampled by the random walk) and so on. Once this data has been analyzed at the query node, an estimation is made on how much more samples are required (and in what way should these samples be collected) so that the original query can be optimally answered within the desired accuracy, with high probability. For example, the first phase may recommend that the best way to answer this query is to visit m^0 more peers and from each peer, randomly sample t tuples. We mention that the first phase is not overly driven by heuristics. Instead, it is based on underlying theoretical principles such as the theory of random walks as well as statistical techniques such as cluster sampling, block-level sampling and cross validation.

The second phase is then straightforward: A random walk is reinitiated and tuples are collected according to the recommendations made by the first phase. Effectively, the first phase is used to "sniff" the network and determine an optimal-cost "query plan," which is then implemented in the second phase. For certain aggregates such as COUNT and SUM, further optimizations may be achieved by pushing the selections and aggregations to the peers; that is, the local aggregates instead of raw samples are returned to the query node, which are then composed into a final answer. In addition, we explore in-network techniques for dissemination of values throughout the network. We accomplish this through a hybrid technique building upon the Gossip protocol. A Gossip protocol is executed in rounds.

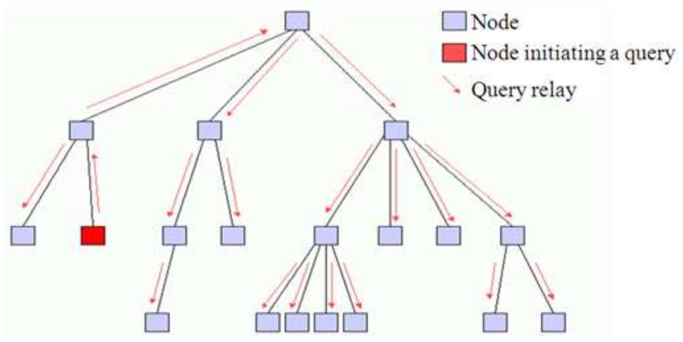


Fig 1. Existing P2P network

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. This process relies heavily upon mass conversation, which describes that the average of all of the sums of individual peers is the correct average and the sum of all of the weights is n. In general, as the number of passes of the Gossip protocol increases, values of participating peers are increasingly diffused through the network (in our case, the local groups); therefore, sampling-diffused values provide a better representation of the values contained in the network as opposed to a single peer. The contributions of this study are summarized as follows:

- 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 network computation building upon the Gossip protocol
- Adaptive two-phase sampling-based approaches are proposed based on well-founded theoretical principles
- We present an adaptive approach for computing aggregates such as COUNT, SUM, AVERAGE and MEDIAN

1.7. Two-Phase Algorithm

In this section, we present details of our two-phase algorithm for approximate answering of aggregate queries. For illustration, we focus on approximating COUNT queries (it can be easily extended to SUM, AVERAGE and MEDIAN queries).

2. COUNT

First, we perform a random walk on the P2P network, attempting to avoid skewing due to graph clustering and vertices of high degree. Our walk skips j nodes between each selection to reduce the dependency between consecutive selected peers. As the jump size increases, our method increases overall bandwidth requirements within the database, but for most cases, small jump sizes suffice for obtaining random samples.

Second, we compute aggregates of the data at the peers and send these back to the query node. In reality, the local databases at some peers can be quite large and aggregating them in their entirety may not be negligible as compared to the overhead of visiting the peer. In other words, the simplistic cost model of only counting the number of visited peers is inappropriate. In such cases, it is preferable to randomly subsample a small portion of the local database and apply the aggregation only to this subsample. Thus, the ideal approach for this problem is to develop a cost model that takes into account cost of visiting peers, as well as local processing costs. Moreover, for such cost models, an ideal two-phase algorithm should determine various parameters in the first phase, such as how many peers should be visited in the second phase and how many tuples should be subsampled from each visited peer. We have taken a somewhat simpler approach in which we fix a constant t (determined at preprocessing time via experiments) such that if a peer has at most t tuples, then its database is aggregated in its entirety, whereas if the peer has more than t tuples, then t tuples are randomly selected and aggregated. Subsampling can be more efficient than scanning the entire local database, for example, by blocklevel sampling, in which only a small number of disk blocks are retrieved. If the data in the disk blocks are highly correlated, then it will simply mean that the number of peers to be visited will increase, as determined by our crossvalidation approach at query time. Third, we estimate the CVError of the collected sample and use that to estimate the additional number of peers that need to be visited in the second phase. For improving robustness, steps 2-4 in the cross-validation procedure can be repeated a few times, as well as the average squared CVError computed.

Once the first phase has completed, the second phase is then straightforward. We simply initiate a second random walk based on the recommendations of the first phase and compute the final aggregate.

2.1. SUM and AVERAGE

Although the algorithm has been presented for COUNT queries, it can be easily extended to other aggregates such as the SUM and AVERAGE by modifying the y (Curr) value specified on line 8, phase 1 of the algorithm. For the SUM, no changes are required and for the AVERAGE, (#tuples/#process Tuples) is removed from y (Curr), since no scaling is required.

2.2. Algorithms for Choosing a Random Peer

We consider two types of algorithm for choosing a random peer (King and Saia, 2004) among the neighbour peers:

- Peer Count Algorithm
- Arc Length Algorithm

Here we consider the peer count algorithm as follows:

2.3. Peer Count Algorithm

A peer p initially calls FindParameters () to determine values for dminp and tp and sets λ to dminp/tp. Then the algorithm enters a loop in which it selects a random number r from (0, 1). It moves clockwise around the circle to the next peer until a peer p^1 is encountered such that dist(r, p^1)<λ num(r, p^1) or tp peers have been examined.

If such a peer is found, it is returned; otherwise, the loop is repeated. One execution of a loop is referred to as a round.

2.3.1. Algorithm

```
1. $T_p$,$dmin_p \leftarrow$ FindParameters ();
2. $\lambda \leftarrow dmin_p/t_p$;
3. While TRUE do:
4. R $\leftarrow$ random number in [0, 1];
5. First $\leftarrow$ h (r); T $\leftarrow$ dist(r, first)-λ;
6. Repeat tp times or until T<0:
7. T $\leftarrow$ T+dist (first, next (first))-λ;
8. First $\leftarrow$ next (first).
9. If T<0 return first;
```

In this algorithm used in Hybrid approach to select peer randomly from the neighbour peers.

3. HYBRID SOLUTION

In order to further improve the quality of our random sampling process, we have employed a hybrid sampling technique by allowing individually selected peers (Peer count Algorithm) to perform additional sampling in parallel with the random sampling phase. We exploit the fact that during a random walk, previously selected peers can perform further independent processing while waiting for the final peer to be selected for sampling during the random-walk phase. The ability to execute in-network computation is a valuable tool for maximizing sample quality and reducing the required jump size for individual queries. Our hybrid technique can be utilized for many aggregate types including SUM, AVERAGE and COUNT queries.

3.1. Two Phase Adaptive Algorithm

3.1.1. Predefined Values

M:Total number of peers in network E:
Total number of edges in network M:
Number of peers to visit in phase 1 J:
Jump size for random walk
T: max #tuples to be subsampled per peer Inputs Q:
COUNT query with selection condition:
Sink: peer where query is initiated
Δreq: desired max error
Phase 1

```
        // perform random walk
        Curr = Sink; Hops = 1;
        while (Hops < j * m)
        {
            if (Hops % j) Visit(Curr);
            Hops++;
            Curr = Random adjacent peer
    }
```

```
// Visit peer
  V visit (Curr)
  {
        if (NO of tuples of Curr)<= t) Execute Q on all tuples else
        Execute Q on t randomly sampled tuples
        y (Curr) = (no of tuples/no of processed tuples)*result of Query;
      Return (y (Curr); deg (Curr)) to Sink
  }
// Cross validate at sink
Let S = {s1,s2, . . . , $s_m$} be the visited peers Partition
      S randomly into halves: S1 and S2

      Compute $y''1 = \sum_{s\in s1}\left(\frac{y(s)}{\Pr ob(s)}\right) / \left(\frac{m}{2}\right)$
              $y''2 = \sum_{s\in s2}\left(\frac{y(s)}{\Pr ob(s)}\right) / \left(\frac{m}{2}\right)$

where, prob(s) = deg(s)/2E
Compute CVError = $|y''_1 - y''_2|$
Return m'   =   (m/2)* $(CVError^2/\Delta^2_{req})$
Phase 2
   1)   Visit m' peers by using random walk
   2)   Let S' = {s1, s2,... $s'_m$) be the visited peers
   3)   Return $y' = \sum_{s\in S'}\left(\frac{y(3)}{\Pr ob(3)}\right) / (m')$
```

First Phase

First, perform a random walk on the P2P network, attempting to avoid skewing due to graph clustering and vertices of high degree. Our walk skips j nodes between each selection to reduce the dependency between consecutive selected peers. As the jump size increases, our method increases overall bandwidth requirements within the database, but for most cases, small jump sizes suffice for obtaining random samples. Second, compute aggregates of the data at the peers and send these back to the query node. Note that in Section 3, we had not formally discussed the issue of subsampling at peers: This was primarily done to keep the previous discussion simple. In reality, the local databases at some peers can be quite large and aggregating them in their entirety may not be negligible as compared to the overhead of visiting the peer. In other words, the simplistic cost model of only counting the number of visited peers is inappropriate. In such cases, it is preferable to randomly sub sample a small portion of the local database and apply the aggregation only to this sub sample. Thus, the ideal approach for this problem is to develop a cost model that takes into account cost of visiting peers, as well as local processing costs. Moreover, for such cost models, an parameters in the first phase, such as how many peers should be visited in the second phase and how many tuples should be sub sampled from each visited peer. In this study, we have taken a somewhat simpler approach in which we fix a constant t (determined at preprocessing time via experiments) such that if a peer has at most t tuples, then its database is aggregated in its entirety, whereas if the peer has more than t tuples, then t tuples are randomly selected and aggregated. Subsampling can be more efficient than scanning the entire local database, for example, by block level sampling, in which only a small number of disk blocks are retrieved. If the data in the disk blocks are highly correlated, then it will simply mean that the number of peers to be visited will increase, as determined by our cross validation approach at query time. Third, we estimate the CrossValidationError of the collected sample and use that to estimate the additional number of peers that need to be visited in the second phase. For improving robustness**,** steps 2-4 in the cross-validation procedure can be repeated a few times, as well as the average squared CroseValidationError computed.

Second Phase

Once the first phase has completed, the second phase is then straightforward. We simply initiate a second random walk based on the recommendations of the first phase and compute the final aggregate.

3.2. Hybrid Algorithm

We propose a hybrid solution for random sampling, focusing on extending our technique with a hybrid in-network decentralized approach. Upon selection of a peer p_i by the random-walk phase, p_i contains a period p_i period where further processing may be performed to improve the quality of a peer's local data. The period p_i period is defined as the number of hops remaining in the random-walk phase before the final peer p_m is selected for sampling. In order to exploit these periods, we propose an incremental decentralized sampling technique building upon the Gossip protocol over the network due to gossiping may be varied based upon the user-defined parameters r_a and r_r. Parameter r_a is the number of edges that a peer may randomly select for gossiping and r_r is the maximum number of hops from p_i that gossiping is permitted. Regardless of the value of r_a or r_r, the number of messages sent to the query node remains constant. These two parameters combined allow the user to leverage in-network computation, without affecting the number of messages sent back to the query node, avoiding possible bottlenecks.

The Gossip protocol works as follows: For each peer in the path p_i, compute the sum of all of the tuples contained in the peer. For each randomly selected adjacent peer p_{ij} of p_i, send the aggregate values s_{ij} and an associated weight w_{ij} representing the contributions of the randomly selected adjacent peers of p_i. For each selected peer p_i, local sets are created $s_i1...n$ and $w_i1...n$ from contributed s_{ij} and w_{ij} values. From the adjacent peers, these sets are used to revaluate s_i and w_i for the current peer p_i. This process allows adjacent peers to mix data and allows local data sets to take into consideration both the local and the contributed values. Each round of the Gossip algorithm involves a single pass where neighboring peers contribute local values to adjacent peers. A single iteration of the gossip algorithm (Kempe *et al*., 2003), xecutes as follows:

- Each peer p_i maintains a local sum s_i and weight w_i. All weights w_i are initialized to 1 for average and 0 for sum
- For each peer, let $s_i1...n$ represent the Contributed sums from all randomly selected adjacent peers and $w_i1...n$ represent the number of tuples contributed from each respective adjacent peer
- Each peer p_i randomly selects r_a adjacent peer's $p_i1...r_a$ for gossiping
- Assuming that the associated weights for each item in $s_i1...n$ is 1, our new s_i value is the sum of the vector $s_i1...r_a$ and similarly, w_i is the sum of the vector $w_i1...r_a$
- Send (1 = 2) s_i and (1 = 2)w_i to randomly selected adjacent peers and p_i. By sharing (1/2), peers may exchange information between peers while still obeying the mass conservation requirement.
- With each aggregate replaced with computed s_i and w_i, the new estimated aggregate is s_i/w_i

Mixing between peers increases the diffusion of values through the network. For our purpose, there is no specific constraint on the number of iterations required before exiting. Under our hybrid approach, the algorithm attempts to maximize the amount of mixing per peer p_i by exploiting the period before a peer must send a sample back to the query node. As stated, the diffusion speed of the network can be represented as T (n, €) = O (log n+log 1/€) 0 for expander-type networks. In addition, for very long walks, convergence may occur before the final peer has been selected, but we can continue to perform gossiping, without loss of benefit, since P2P networks such as Gnutella are, by nature, transient. Where peers are continually entering and leaving the network, gossiping can continue to diffuse new values as peers enter the network.

Simply, since we know how many peers remain to be selected by the random-walk phase, the lower bound for the period pi period is the remaining number of hops required to obtain the required sample, given the specified jump size and sample size. For example, suppose a query is executed with the following parameters: jump size of 10, tuples per peer 100 and a sample size of 400. After selection of the first peer, at least 30 hops are required by the random walk before completion. For the first peer selected p_1, the period p_1 period is equal to 30 hops. This determines that for the next 30 hops, further processing can be utilized to improve the sample quality for the selected peer p_1. Thus, for each consecutive peer selected, the period p_i period is bounded as the (jump size)- the number of remaining peers to be selected).

As shown in **Fig. 2** the earlier that a peer is selected for sampling; the larger is the period available for gossiping. As additional hops are taken to reach the next peer for sampling, already selected peers can continue to gossip (this is represented by the rings around peers for each period). By combining our knowledge of Gossip and the p_i period for selected peers, we can maximize the quality of the sample obtained from individual peers. Our hybrid sampling algorithm executes as follows:

- Given a random start-location peer p_0, the local group is $\{p_0\}$
- Initialize a group for each selected peer p_i: group$_i$ $\epsilon\{p_i\}$
- For each peer in group$_i$, randomly select r_a adjacent peers
- Extend the local group to include adjacent peers if and only if (path from $p_i \le r_r$) group$_i$ u group$_i$ {For each peer in group$_i$ add $p_i1 . . . r_a$}
- Perform Gossip on current group$_i$
- Continue steps 2-5 for each peer in group$_i$ until p_i period has been reached
- All peers selected by the random sampling phase, excluding peers selected by the local groups, send their current mixed values back to the query node
- Compute remaining algorithm normally

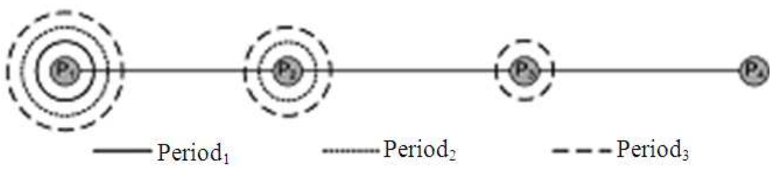


Fig. 2. Each ring represents the increase in gossiping per period for each peer.

Peers near the beginning of the random walk have a longer period to gossip, whereas peers closer to the end of the walk contain an incrementally smaller period for gossiping. This creates an uneven level of mixing among the local groups of peers, but since all peers obey mass conservation as previously defined, the number of rounds performed by each group doesn't affect the overall results between the different Gossiping groups.

4. RESULTS

4.1. ACTIVE NODES DETECTION

The **Fig. 3** shows the Active node detection. This Active node detection deals with cases:

4.1.1. Listing All Nodes

This case deals with finding all the nodes which are connected to the network.

4.1.2. Active Nodes Detection

This case takes the input from the previous case and detects the active nodes by checking for sql server connection. If the node is having sql server connectivity, then it is listed as active node.

4.2. AGGREGATE QUERIES

4.2.1. AVERAGE

The average function is done for two cases:

4.2.1.1. EXACT AVERAGE

This deals with calculating the exact average value by randomly selecting any one of the active nodes. By considering the selected active node, calculate the average with all the tuples present in the table.

4.2.1.2. APPROXIMATE AVERAGE

This deals with calculating the approximate value by randomly selecting any one of the nodes from the active nodes and from the randomly selected nodes, randomly select tuples. Then the calculated value is checked against the exact value. Choose the acceptable error rate (say 25% or less).Then using the exact and approximate values, determine the error rate for comparison. If the error rate is lesser than or equal to the acceptable error rate, return the average result. Else perform second random walk and check for error rate continue until the error rate is accepted as shown in **Fig. 4** and also given exact result, approximate result of Avg (**Fig. 5**) and comparison chart for exact and approximate processing time (**Fig. 6**).

4.2.2. SUM

Like average function, sum is also computed for two cases. It follows the similar steps of average.

4.2.2.1. EXACT SUM

This deals with calculating the exact sum value by randomly selecting any one of the active nodes. By considering the selected active node, calculate the sum with all the tuples present in the table.

4.2.2.2. APPROXIMATE SUM

This deals with calculating the approximate value by randomly selecting any one of the nodes from the active nodes and from the randomly selected nodes, randomly select tuples. Then the calculated value is checked against the exact value. Choose the acceptable error rate (say 25% or less). Then using the exact and approximate values, determine the error rate for comparison. If the error rate is lesser than or equal to the acceptable error rate, return the calculated result. Else perform second random walk and check for error rate continue until the error rate is accepted as shown in **Fig. 4** and also given exact result, approximate result of Sum (**Fig. 7**) and comparison chart for Exact and Approximate processing time (**Fig. 8**).

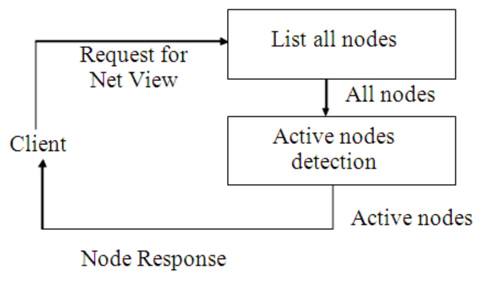


Fig. 3. Active node detection

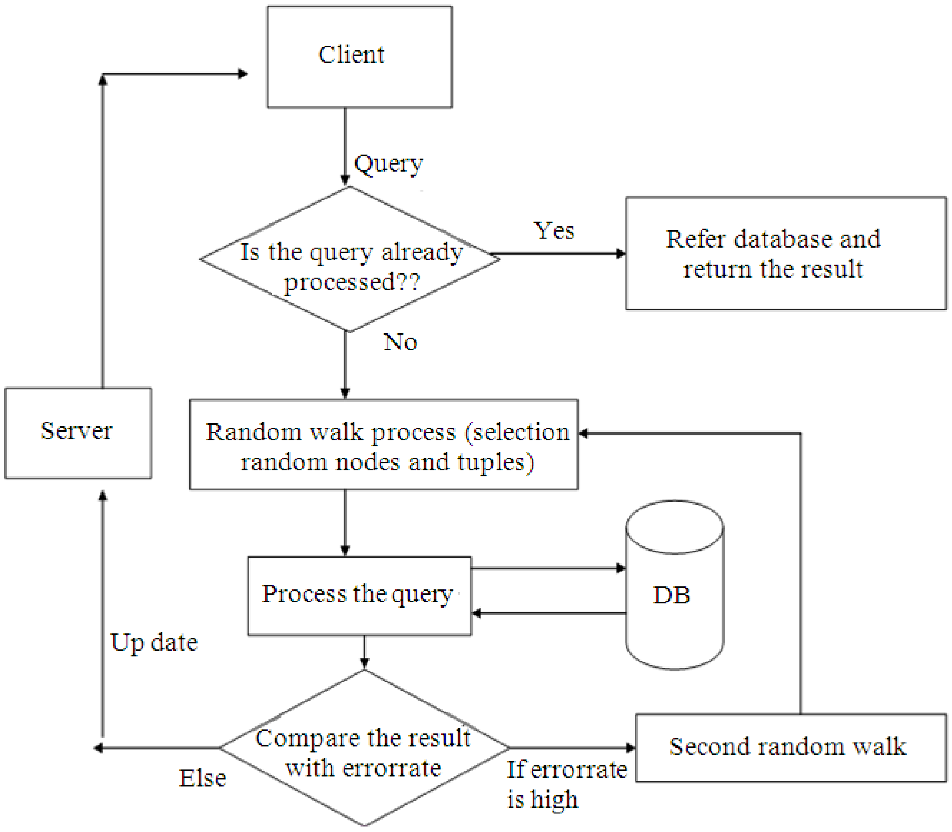
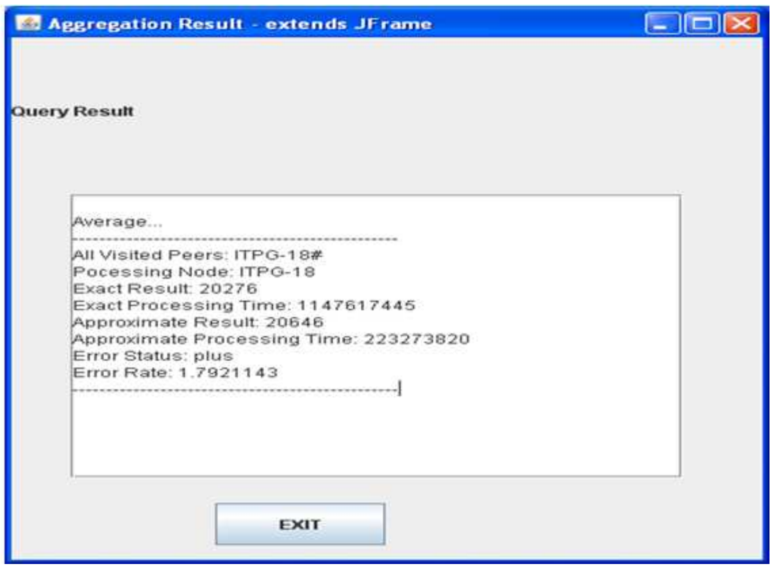

Client
Query
Is the query already processed??
Yes
Refer database and return the result
No
Server
Random walk process (selection random nodes and tuples)
Process the query
DB
Up date
Compare the result with errorrate
Else
If errorrate is high
Second random walk

Fig. 4. Average



```
Average...
--------------------------------------------------
All Visited Peers: ITPG-18#
Pocessing Node: ITPG-18
Exact Result: 20276
Exact Processing Time: 1147617445
Approximate Result: 20646
Approximate Processing Time: 223273820
Error Status: plus
Error Rate: 1.7921143
--------------------------------------------------
```

Fig. 5. Average

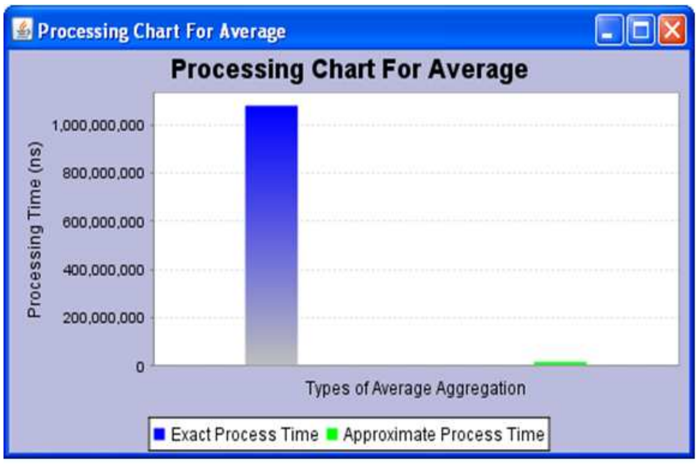
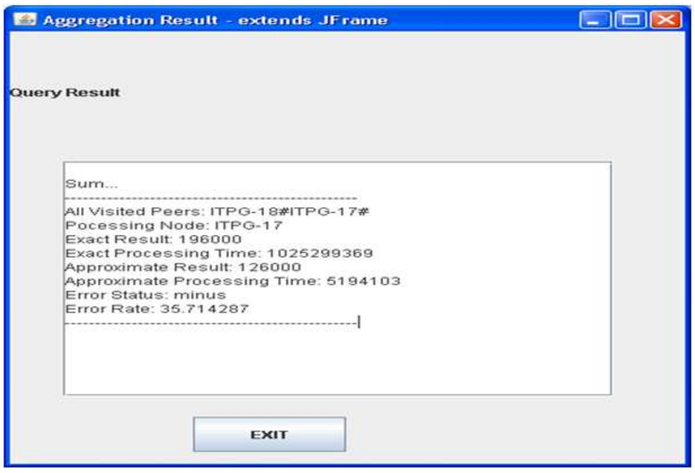


Fig. 6. Processing chart-average



```
Sum...
--------------------------------------------------
All Visited Peers: ITPG-18#ITPG-17#
Pocessing Node: ITPG-17
Exact Result: 196000
Exact Processing Time: 1025299369
Approximate Result: 126000
Approximate Processing Time: 5194103
Error Status: minus
Error Rate: 35.714287
--------------------------------------------------|
```

Fig. 7. Sum

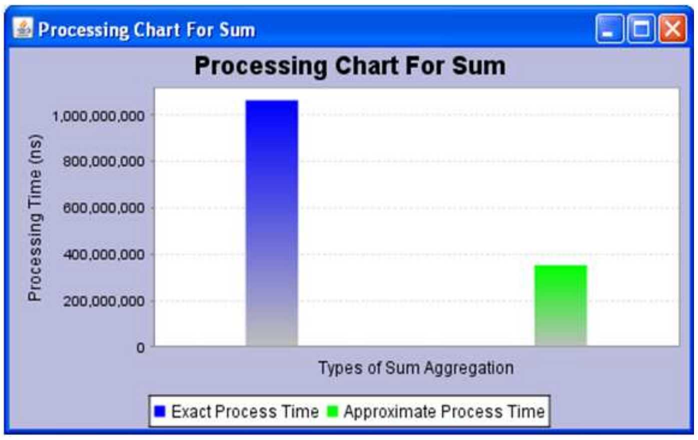


Fig. 8. Processing chart-sum

4.2.3. COUNT

This function follows the similar steps of average.

4.2.3.1. EXACT COUNT

This deals with calculating the exact count value by randomly selecting any one of the active nodes. By considering the selected active node, calculate the count with all the tuples present in the table.

4.2.3.2. APPROXIMATE COUNT

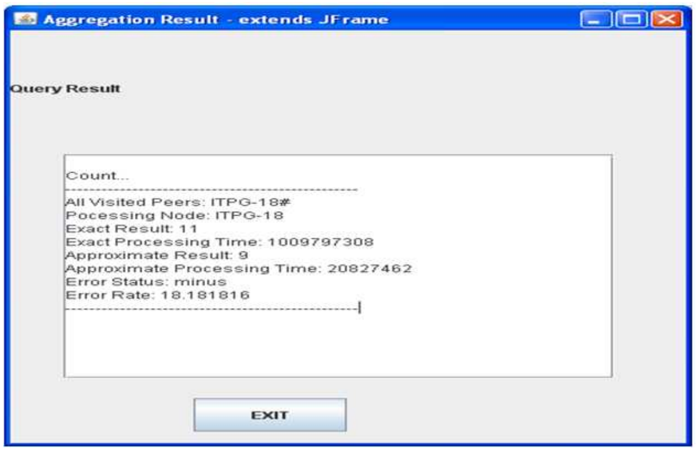
This deals with calculating the approximate value by randomly selecting any one of the nodes from the active nodes and from the randomly selected nodes, randomly select tuples. Then the calculated value is checked against the exact value. Choose the acceptable error rate (say 25% or less). Then using the exact and approximate values, determine the error rate for comparison. If the error rate is lesser than or equal to the acceptable error rate, return the calculated result. Else perform second random walk and check for error rate continue until the error rate is accepted as shown in **Fig. 4** and also given exact result, approximate result of Count (**Fig. 9**) and comparison chart for Exact and Approximate processing Time (**Fig. 10**).

4.2.4. MINIMUM

This function detects and displays the minimum value in the table selected by the user. Processing steps is similar to the above queries.

4.2.5. MAXIMUM

This function detects and displays the maximum value in the table selected by the user. Processing steps is similar to the above queries.



```
Count...
------------------------------------------------
All Visited Peers: ITPG-18#
Pocessing Node: ITPG-18
Exact Result: 11
Exact Processing Time: 1009797308
Approximate Result: 9
Approximate Processing Time: 20827462
Error Status: minus
Error Rate: 18.181816
------------------------------------------------|
```

Fig. 9. Count

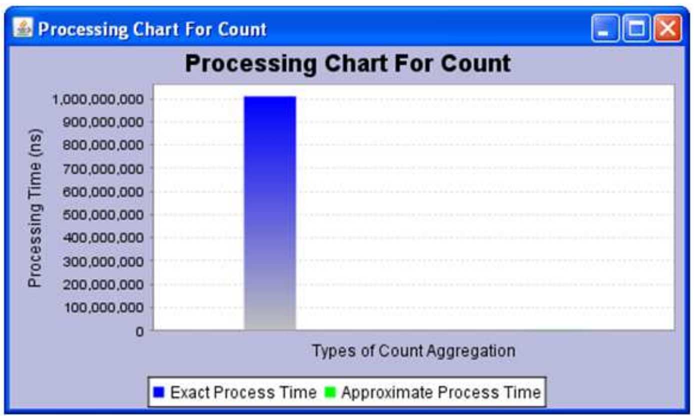


Fig. 10. Processing chart-count

4.3. Processing Chart

A chart for each aggregate query. The chart compares the processing time involved for computing exact and approximate values. The processing time is expressed in nanoseconds. The chart clearly says that the time involved for computing approximate values using Hybrid algorithm will be less compared to exact processing.

5. CONCLUSION

In this project, Hybrid algorithm used for the approximate answering of ad hoc aggregation queries in P2P databases. This approach requires a minimal number of messages sent over the network and provides tunable parameters to maximize performance for various network topologies. This approach provides a powerful technique for approximating aggregates of various topologies and data clustering but comes with limitations based upon a given topologies structure and connectivity. For topologies with very distinct clusters of peers, it becomes increasingly difficult to accurately obtain random samples due to the inability of random-walk process to quickly reach all clusters. This can be resolved by increasing the jump size, allowing a larger number of peers to be considered and increasing the allowed mixing by our hybrid algorithm. By varying a few parameters, Hybrid algorithm successfully computes aggregates within a given required accuracy.

5.1. FUTURE ENHANCEMENTS

As with the development of the computer field and its related technology this project is a very dynamic and user-friendly which blends with the needs and the great deal of requirements of the individual user and also the needs of industries. This project has been developed with ideas of flexibility and scalability. This project does not stay resident to the system but grows to the level of great assistance in the field of data warehousing and data mining. Though this project at its initial level is a pre-cursor to data warehousing and mining operations it is truly sure that it is of good help in the financial and share-market oriented fields of data handling. To make things feasible there are sure certain improvements to be made in this project like the determination of the error rate with the view of user inputs and the random selection of the records in a more efficient manner. The queries can be made more dynamic. To enhance the security operations we can add Asymmetric Encryption algorithm.

6. REFERENCES

Das, G., D. Gunopulos B. Arai and V. Kalogeraki, 2007. Efficient approximate query processing in peer-to-peer networks. IEEE Trans. Knowl. Data Eng., 19: 919-933.DOI: 10.1109/TKDE.2007.1064

King V. and J. Saia, 2004. Choosing a random peer. Proceedings of the 23rd Annual ACM Symposium on Principles of Distributed Computing, Jul. 25-28, ACM Press, St John, Newfoundland, Canada, pp: 125-130. DOI:10.1145/1011767.1011786

Babcock, B., S. Chaudhuri and G. Das, 2003. Dynamic sample selection for approximate query processing, Proceedings of the 22nd International Conference Management of Data, Jun. 09-12, ACM New York, NY, USA., pp: 539-550. DOI: 10.1145/872757.872822

Kempe, D., A. Dobra and J. Gehrke, 2003. Gossip-based computation of aggregate information. Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, Oct. 11-14, IEEE Xplore Press, pp: 482-491. DOI: 10.1109/SFCS.2003.1238221

Zhao, J., R. Govindan and D. Estrin, 2003. Computing aggregates for monitoring wireless sensor networks. Proceeding of the 1st IEEE International Workshop Sensor Network Protocols and Applications, May 11-11, IEEE Xplore Press, pp: 139-148. DOI: 10.1109/SNPA.2003.1203364

Gribble, S., A. Halevy, Z. Ives, M. Rodrig and D. Suciu, 2001. What can peer-to-peer do for databases and vice versa? Proceedings of the 4th International Workshop on the Web and Databases (WDB’ 01), Santa Barbara, California, USA.

Gupta, A., D. Agrawal and A.E. Abbadi, 2002. Approximate range selection queries in peer-to-peer systems. University of California at Santa Barbara.