ISSN:2231-0711
www.ijcset.net

Evaluated and Suggested Range aggregate query approximation on data cubes in P2P networks

K.Seetha Devi, CH.V. Phani Krishna, P.Venkateswara Rao
CSE Department, KL University,Guntur dt., India.

Abstract- **Individual computers provide opportunities for tremendous productivity gains, but they become many times more powerful when they're connected to one another forming a peer to peer network that gives them the ability to share data and processing resources. This paper presents data exchange between peers occurs when one of them, in the role of a local peer, needs data available in other nodes, denoted the acquaintances of the local peer and also an adaptive dual-phase approach based on random walks of the P2P graph for efficiently serving range aggregate queries on data cubes in a peer to peer system. The problem of answering large-scale ad hoc analysis queries, for example, range aggregate queries on data cubes possesses unique challenges. Exact solutions can be time consuming and difficult to implement, given the distributed and dynamic nature of P2P data cubes. In this paper, we use an I/O-efficient technique based up on a multi resolution wavelet decomposition that yields excellent approximations for range aggregate queries with limited space usage and computational cost.**

Keywords- **OLAP, data cubes, aggregate queries, wavelet decomposition, P2P graph**

I. INTRODUCTION

From the past few years, there is an increase in the use of complex databases for data analysis by various scientific and business organizations. Organizations frequently expand, adding geographically distributed branches and acquiring subsidiaries. As a result, a single centralized data warehouse may be too expensive or difficult to construct. Instead, the enterprise may temporarily or permanently decide to utilize a number of smaller, remotely located data warehouses. The on-line analytical processing (OLAP) or decision support applications, analyze the data in a data warehouse to glean interesting trend information. OLAP systems typically organize the data in the form of a data cube, a hierarchy of multi-dimensional sub cubes, where each sub-cube describes the distribution of data in a set of dimensions, and materialize part or the entire data cube. For example, retailers use OLAP systems to analyze vast amounts of sales data in order to evaluate new marketing strategies. It is well known that these applications access large amounts of data and involve complex aggregate queries. At the same time, they require very quick responses. The dynamics of large-scale distributed systems are often significantly different. For example, in P2P networks, individual machines are often under the control of a large number of heterogeneous users who may join or leave the network at any time [2] Peer to peer systems have many interesting technical aspects like decentralized control, self organization, adaptation and scalability. Peer-to-peer systems can be characterized as distributed systems in which all nodes have identical capabilities and responsibilities and all communication is symmetric [3] three aspects of a P2P system: P2P network topology, query distribution and replication. By network topology, we mean the graph formed by the P2P overlay network; each P2P member has a certain number of neighbors" and the set of neighbor connections forms the P2P overlay network [7].

A. Goal of the Paper

We aim at significantly reducing the load for answering range aggregate queries on data cubes in peer-to-peer networks. Specifically, we concentrate on I/O-efficient technique based upon a multi resolution wavelet decomposition that yields a compact and accurate representation of the data cube built on the logarithms of the partial sums of the raw data values. The compact cube after normalization and thresholding to reduce storage cost provides fast and accurate answers to on-line range aggregate queries.

An approach with these characteristics can be useful in various contexts. For example, consider a worldwide company on an enterprise network or a worldwide virtual organization with users interested in geographical data. In both cases the users would not be interested in wasting time in performing data analysis related operations. Hence we introduce wavelets based approximate answering to any range aggregate query initiated by any peer in the network.

B. Our Approach

We briefly describe the framework of our approach. Our approach has two major phases. 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). At each visited peer we perform approximate aggregation operations that include sum, average, min, and max. We have used a multi resolution wavelet decomposition that yields a compact data cube and provide approximate answers to any range aggregate query from these summaries.

The aggregate measures obtained are then analyzed at the query node to determine the skewed nature of the data that is distributed across the network, such as the variance in the degrees of individual nodes in the P2P graph and so on. Once this data has been analyzed at the query node, estimation is made on how many more peers are required so that the original query can be optimally answered within the desired accuracy, with high probability.

The second phase is then straightforward: A random walk is reinitiated, and peers are selected 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.

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. For each round, participating peers select adjacent peers uniformly at random sharing information.

C. Related work

P2P systems are very popular because they provide an efficient mechanism for building large scalable systems. Many P2P systems have been proposed for data management, including P2P databases such as PIER [20]. Exact solutions to OLAP queries have been considered in [15]. Methods to sample random peers in P2P networks have been proposed in [16], [19]. These techniques use Markov-chain random walks to select random peers from the network. . Data warehouses can be extremely large, yet obtaining quick answers to queries is important. In many situations, obtaining the exact answer to an OLAP query is prohibitively expensive in terms of time and/or storage space. It can be advantageous to have fast, approximate answers to queries.[6]

Peer OLAP acts as a large distributed cache, which amplifies the benefits of traditional client-side caching.[4] Peer OLAP which is a distributed caching system for OLAP queries based on a Peer-to-Peer1 (P2P) network. The contributions of this work include: (i) the proposal of the Peer OLAP architecture, (ii) the employment of three cache control policies that impose different levels of cooperation among the peers, and (iii) the development of adaptive techniques that dynamically recognize the network structure in order to minimize the query cost.[4] an adaptive, bandwidth-efficient and easy to deploy search algorithm for unstructured Peer-to-Peer networks, the Adaptive Probabilistic Search method (APS)[5]

Previous work on data cube computation has concentrated on how to compute the exact data cube. But in reality, a data warehouse usually consists of many tables and hence very expensive. Computing all the extended data cubes and storing and retrieving them on disk becomes infeasible when the number of underlying relations/cubes is large since no enough disk storage is available. Ho et al. [2] present an efficient algorithm to speed up range-sum queries on a single data cube. The main idea is to preprocess the (raw) data cube A and pre-compute all the multidimensional partial sums, which can be represented in what we call the partial sum data cube P. Any range-sum query can be answered by accessing and computing 2^d entries from the partial sum data cube, where d is the number of dimensions for which ranges have been specified in the query. The storage required for this partial sum data cube can be proportional to the size of the raw data cube, which is very large. Hence we employ a wavelet decomposition based technique that has less storage cost and provides fast and accurate answers.

There are known techniques for computing approximate aggregates in distributed settings (most notably, the Gossip protocol [13], [14], [18]). 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.

D. Contributions and Overview

The contributions of this paper are summarized as follows:

- We address the important problem of AQP on data cubes in P2P systems, which is likely to be of increasing significance in the future.
- We introduce I/O efficient algorithm to compute the partial sum cube and multidimensional wavelet decomposition of large data cubes in peer to peer networks.
- We implement an efficient thresholding method [10] based on the logarithm transform that dramatically reduces the relative and absolute error in the approximations.
- By using the thresholding method in building wavelet-based histograms as proposed in [10], better accuracy can be achieved even for the low dimensional data.
- Hybrid sampling technique maximizes per-peer in-network computation building upon the Gossip protocol.
- Adaptive two-phase approaches are used based on well-founded theoretical principles.

II. FOUNDATIONS OF OUR APPROACH

In this section, we discuss the principles behind our approach for AQP on P2P databases.

A. The Peer-to-Peer Model

We assume an unstructured P2P network represented as a graph $G = \{ P,E \}$ with a vertex set $P = \{ p_1,p_2,\ldots, p_M \}$ and an edge set E. The vertices in P represent the peers in the network, and the edges in E represent the connections between the vertices in P. Each peer p is identified by the processor's IP address and a port number (IP_p and $port_p$). In unstructured P2P networks, a node becomes a member of the network by establishing a connection with at least one peer currently in the network. Each node maintains a small number of connections with its peers. The number of connections is typically limited by the resources at the peer. We denote the number of connections that a peer is maintaining by p_{conn}. The peers in the network use the Gnutella P2P protocol to communicate. Our approach, on the other hand, uses a probabilistic search algorithm based on random walks. The key idea is that each node forwards a query message, called walker, randomly to one of its adjacent peers. This technique is shown to improve the search efficiency and reduce unnecessary traffic in the P2P network.

B. Random Walk in Graphs

In order to select random peers in a network a Markov-chain random walk is used. It is a procedure that is initiated at the query node, and for each visited peer, the next peer to visit is selected with equal probability from among its neighbors. 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 ={ $p_1,p_2,\ldots,p_M$ } be the entire set of peers, let E be the entire set of edges, and let the degree of a peer p be deg(p). Then, the probability of any peer p in the stationary distribution is: prob(p) = $\frac{deg(p)}{2|E|}$.

It is important to note that the above distribution is not uniform. The probability of each peer is proportional to its degree. We assume that we are allowed a certain amount of preprocessing to determine various properties of the P2P graph that will be useful at query time. The speed of convergence of a random walk in this graph is determined in this preprocessing step, in addition to other useful properties such as the number of nodes M, the number of edges |E|, and so on. With respect to speed of convergence, we essentially determine a *jump* parameter j that determines how many peers can be skipped between selections of peers for the sample. As the jump increases, the correlation between successive peers that are selected for the sample decreases rapidly.

C. Our Compact Data cube Construction Algorithm

Let D={$D_1,D_2,\ldots D_d$ } denote the set of dimensions where each dimension corresponds to a functional attribute .We represent the d-dimensional (raw) data cube A by a d-dimensional array of size $|D_1|x|D_2|x \ldots\ldots |D_d|$, where $|D_i|$ is the size of dimension D_i. The problem of computing a range-sum query in a d-dimensional data cube can be formulated as follows:

Sum ($l_1:h_1,..,l_d:h_d$) = $\sum_{i_1=l_1:h_1} \cdots \sum_{i_d=l_d:h_d} A[i_1,\ldots,i_d]$.

The partial sum data cube P is a d-dimensional array of size $|D_1|x|D_2|x \ldots\ldots x|D_d|$ (which is the same as the size of A). Its cells are defined as: P[$x_1,\ldots\ldots,x_n$] = Sum ($0:x_1,\ldots..,0:x_d$) =

$\sum_{0=i_1:x_1} \cdots \cdots \sum_{0=i_d:x_d} A[i_1,\ldots,i_d]$.

At a high level, our approximate data cube construction algorithm works as follows:

1. In a preprocessing step, we form the partial sum data cube P from the (raw) data cube A. (In our method, we process P by replacing each cell value by its natural logarithm.)
2. We compute the wavelet decomposition of P, obtaining a set of N coefficients, where N is the size of array A.
3. We keep only the C most significant wavelets coefficients, for some C. The choice of which C coefficients to keep demands upon the particular thresholding method we use.

In the online phase, a query is answered by using the C wavelet coefficients to reconstruct approximation of the necessary values in P.

D. Thresholding and Error Measures

Our motivation in this paper is a compact, yet accurate representation of the partial sum data cube. Given the storage limitation for the compact data cube, we can only "keep" a certain number of the N wavelet coefficients. Let C denote the number of wavelet coefficients that we have room to keep; the remaining wavelet coefficients are implicitly set to 0. Typically we have C«N. The goal of thresholding is to determine which are the "best" C coefficients to keep, so as to minimize the error of approximation.

We can measure the error of approximation in several ways. Let v_i be the actual answer of a query q_i and let $\hat{v}_i$ be the approximate answer. We use the following four different error measures for the error e_i of approximating query q_i :

Type	Notation	Definition
Absolute error	e_i^{abs}	$\lvert v_i - \hat{v}_i \rvert$
Relative error	e_i^{rel}	$\frac{\lvert v_i - \hat{v}_i \rvert}{Max\{1, V_i\}}$
Modified relative error	$e_i^{m\_rel}$	$\frac{\lvert v_i - \hat{v}_i \rvert}{max\{1, min\{v_i - \hat{v}_i\}\}}$
Combined error	e_i^{comb}	$min\{\alpha \times e_i^{abs}, \beta \times e_i^{rel}\}$

The parameters α ,β are positive constants.

These error measures are special cases of the p-norm average error, for p>0 :

$$\|e\|_p = \left(\frac{1}{Q} \sum_{1 \le i \le Q} e_i^p\right)^{\frac{1}{p}}.$$

The first step in thresholding is normalizing the coefficients in a certain way (which corresponds to using a particular basis, such as an orthonormal basis). To minimize the relative error: We take the natural Logarithm of each element in P before we do the wavelet decomposition, and we apply the inverse (i.e., exponentiation) after reconstruction in the on-line phase. This logarithm transformation not only does it dramatically lower the relative error of the approximation in our experiments, it also lowers the absolute error, no matter which norm we use to measure the error. Such a phenomenon does not occur when the logarithm transform is used with histogram methods, such as MaxDiff histogram [17], for example; the MaxDiff relative error shows some improvement (compared with when the logarithm transform is not used), but its absolute error increases substantially. The C wavelet coefficients together with their C indices(in the one-dimensional order of cells), form the compact data cube.

1) Answering Range Queries in the On-Line Phase

Each range-sum query can be expressed as sums and differences of a certain set of cell values from the multidimensional partial sum data cube P [2]. The set of cells are the ones on the corners of the query hyper plane:

Theorem 1 [12] : The answer for the d-dimensional range sum query

$l_1 \le D_1 \le h_1$ AND …… AND $l_d \le D_d \le h_d$ is v = $\sum_{i_k \in \{l_k - 1, h_k\} \text{ for each } 1 \le k \le d} \left(\prod_{1 \le k \le d} s(l_k)\right) \times P[l_1, l_2, \ldots\ldots, l_d]$,

where $s(k) = \begin{cases} 1 & \text{if } l_k = h_k, \\ -1 & \text{if } l_k = l_k - 1. \end{cases}$

By convention, we define $P[l_1, l_2, \ldots\ldots, l_d]$=0 if i_j = -1 for any $1 \le j \le d$.

We make use of Theorem to compute our approximation $\hat{v}_i$ of the query value v by computing an approximate reconstruction of each needed cell value $P[i_1,i_2,..... i_d]$ in (4). Each reconstruction is based on the inverse wavelet transform of the C wavelet coefficients; the other N-C coefficients are implicitly set to 0. The time for reconstruction is crucial for the query performance.

2) Sampling Theorems

In this section, we shall develop the formal sampling theorems that drive our algorithm. It implies how the "query plan" should be executed, for answering the query approximately so that a desired error is achieved. Let $P = \{p_1,p_2, . . . , p_M\}$ is the set of peers. Let the aggregate for a peer p be y(p)and y be the exact answer for the query, that is, $y=\sum_{p\in P} y(p)$. The query also comes with a desired error threshold Δ_{req}. The implication of this requirement is that if $y^1$is the estimated count by our algorithm, then $|y-y^1| \leq \Delta_{req}$.

Now, consider a fixed-size sample of peers $S = \{s_1; s_2 . . . s_m\}$; where each s_i 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 s_i is picked from P, with probability prob(s_i).

Consider the quantity y^{11} defined as follows:

$$Y^{11} = \frac{\sum_{s \in S} \frac{y(s)}{prob(s)}}{m}$$

Intuitively, each sampled peer s tries to estimate y as y(s) / prob(s), that is, by scaling its own aggregate by the Inverse of its probability of getting picked. The final estimate y^{11} is simply the average of the m individual estimates. Hence, $E[y^{11}] = y$, that is, y^{11}, is an unbiased estimator of y. Standard Error Theorem proves that the variance varies inversely as the sample size.

$Var[y^{11}] = C/m$.

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

The quantity C also represents the "badness" of the clustering of the data in the peers: The larger the C, the more the correlation among the tuples within peers and, consequently, the more peers need to be sampled to keep the variance of the estimator y^{11} small for a given desired error threshold Δ_{req}, and the task is to determine the appropriate number of peers to sample that will satisfy this threshold. A simple cross-validation procedure is described below to estimate C

Consider two random samples of peers of size m, each drawn from the stationary distribution. Let y_1^{11} and y_2^{11} be the two estimates of y by these samples, cross-validation error (CVError) is defined as : CV Error = $|y_1^{11} - y_2^{11}|$.

Theorem 1: $E[CV\ Error^2] = 2E[(y^{11} - y)^2]$

Proof: $E[CV\ Error^2] = E[(y_1^{11} - y_2^{11})^2]$

$= E[(y_1^{11} - y)^2] + E[(y_2^{11} - y)^2]$

$= 2E[(y^{11} - y)^2]$

This theorem says that the expected value of the square of the CVError is two times the expected value of the square of the actual error. This CVError can be estimated in the first phase by the following procedure. Randomly divide the m samples into halves and compute the CVError (for sample size m=2). We can then determine C by fitting this computed error and the sample size m=2 in $Var[y^{11}]$. We also note that since the CVError is larger than the true error, the value of C is conservatively overestimated. Once C is determined (that is, the "badness" of the clustering of data in the peers), we can determine the right number of peers to sample in the second phase m^1 to achieve the desired accuracy.

3) Hybrid Approach

In order to further improve the quality of our random sampling process, we have employed a hybrid sampling technique by allowing individually selected peers 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.

We use 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 pm is selected for sampling. In order to exploit these periods, we propose an incremental decentralized sampling technique building upon the Gossip protocol [18].

The number of messages sent 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. 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.

Our hybrid algorithm executes as follows:

1. Given a random start-location peer p_0, the local group is $\{ p_0 \}$.
2. Initialize a group for each selected peer p_i : $group_i \in \{ p_i \}$.
3. For each peer in $group_i$, randomly select r_a adjacent peers.
4. Extend the local group to include adjacent peers if and only if (path from $p_i \leq r_r$) $group_i \in group_i \cup \{$for each peer in $group_i$ add $p_{i1} . . . r_a\}$.
5. Perform Gossip on current $group_i$.
6. Continue steps 2-5 for each peer in $group_i$ until pi period has been reached.
7. 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.
8. Compute remaining algorithm normally.

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.

III. OUR ALGORITHM

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

A. Count

Our approach in the first phase is broken up into the following main components. 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. 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, 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.

B. 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 in the algorithm. For the SUM, no changes are required, and for the AVERAGE, we calculate SUM/COUNT.

Algorithm: COUNT queries

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

Inputs
Q : COUNT query with selection condition
Sink : peer where query is initiated
Δ_{req} : desired max error

Phase 1
// Perform random walk
1. Curr = Sink; Hops = 1;
2. while (Hops < j * m) {
3. if (Hops % j)
4. Visit(Curr);
5. Hops++;
6. Curr = random adjacent peer
7. }
//Visit peer
1.Visit(Curr) {
2. if (thresholded Cube *P* is present){
3. Thresholding the Data Cube *P* and minimizing the error measures (2.4.3)
4. Execute query in On-Line Phase (2.4.4)
5. else
6. Compute partial Sum Data Cube *P* (2.4.1)
7. Wavelet Decomposition of the Partial Sum Data Cube *P* (2.4.2)
8. Thresholding the Data Cube *P* and minimizing the error measures (2.4.3)
9. Execute the query in On-Line Phase (2.4.4)
10. }
11. y(Curr)=result of Q
12. Return(y(Curr), deg(Curr)) to Sink
13.}

// Cross validate at sink
1. Let S = { $s_1, s_2, \ldots, s_m$ } be the visited peers
2. Partition S randomly into halves: S1 and S2
3. Compute $Y_1^{11} = \frac{\sum_{s \in S1} \frac{y(s)}{prob(s)}}{m/2}$;

$Y_2^{11} = \frac{\sum_{s \in S2} \frac{y(s)}{prob(s)}}{m/2}$ Where $prob(s) = \frac{deg(s)}{2|E|}$;

4. Compute CV Error = $|y_1^{11} - y_2^{11}|$.
5. Return $m^1 = (m/2) * \left(\frac{CV\ Error^2}{\Delta^2_{req}} \right)$

Phase 2

1. Visit m^1 peers by using random walk
2. Let $S^1 = \{s1, s2, \ldots, s_m^1\}$ be the visited peers
3. Return $y^1 = \frac{\sum_{s \in S} \frac{y(s)}{prob(s)}}{m^1}$

IV. EXPERIMENTAL EVALUATIONS

A. Implementation

Our algorithms and P2P topologies are implemented in Java 6.0 with the graph generation tool Jung [17] version 2.0. Our implementation includes both sampled and real-world Gnutella topology samples. All of our experiments were run on Intel Core2Duo 2.0-GHz processors with 2 Gbytes of RAM.

B. 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 (Δ_{req}). This parameter defines the maximum allowed error for the estimated answer. Provided by the user for each query.
2. Jump Size (j). This parameter defines the number of peers to be passed over before selecting the next peer for sampling.

C. Evaluation Metrics

Accuracy, Performance and Efficiency

Our algorithms are evaluated based up on their relative performance with other techniques, accuracy and how close they get to the desired accuracy. Efficiency of our hybrid algorithm is compared with the original algorithm. Here we measure the accuracy as the percentage of error obtained.

D. Experiments

All of our results were generated from three independent experiments and averaged for each individual parameter configurations. Errors are normalized between 0 and 1.

E. Accuracy

Figs. 2 and 3 shows representative accuracy results on applying our hybrid algorithm for COUNT query using synthetic and real world topologies. In this case, we have a COUNT query whose absolute error percentage is noted as we vary the required accuracy. The figure shows that the algorithm's result is always within the required
accuracy.

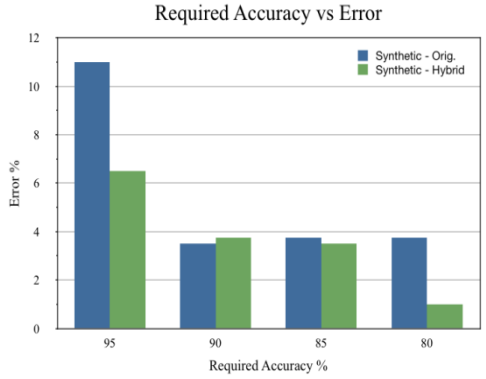

Fig 2: Req Accuracy vs Error in synthetic topologies

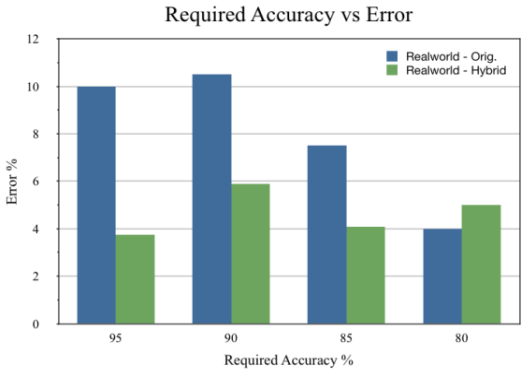

Fig 3: Req Accuracy vs Error in Gnutella topologies

F. Relative performance of wavelets

Figs. 4 and 5 show the effect of storage space on absolute and relative error respectively by using wavelets and its related methods. We can observe that wavelet-based histograms use less storage space at the same time give more accurate results than random sampling and traditional histogram methods.

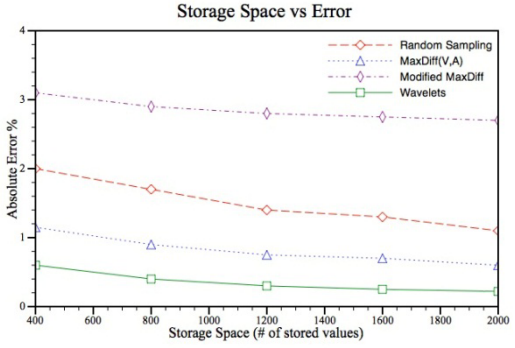

Fig 4: Storage space vs Absolute Error on applying different methods

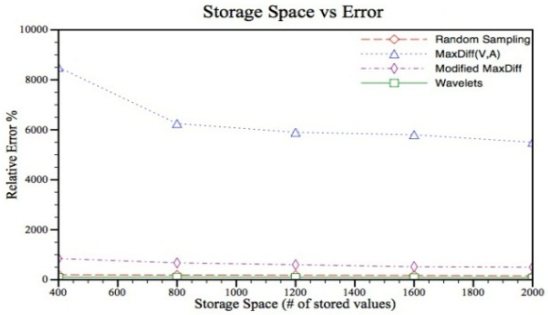

Fig 5: Storage space vs Relative Error on applying different methods

G. Comparison with Naive Techniques

Fig. 6 compares our approach for selecting peers with the traditional techniques like depth-first search (DFS), breadth-first search (BFS) where we execute our query on peers selected using a random walk in the neighborhood of the querying peer. Note that our method always meets the required accuracy. Our technique clearly outperforms both techniques.

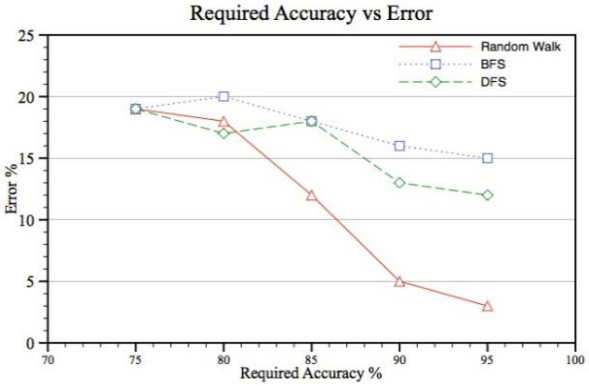

Fig. 6. Random walks perform better than BFS and DFS

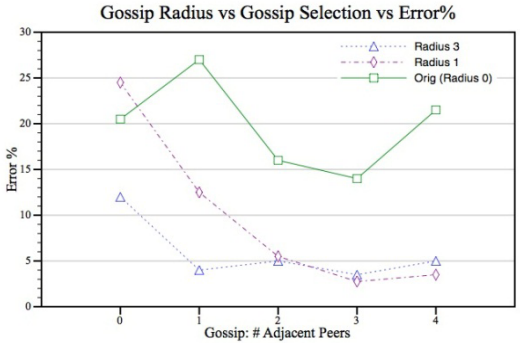

Fig. 7. Effect of Gossip radius with no. of adjacent rs on error percentage for the COUNT query.

H. Efficiency of Hybrid algorithm

Fig 7 includes the original algorithm by setting $\mathbf{r_r} = 0$, which represents the original algorithm without gossiping. Fig. 7 shows that as the number of edges that a peer randomly selects for gossiping $\mathbf{r_r}$ and the maximum number of hops from $\mathbf{p_i}$ that gossiping is permitted $\mathbf{r_a}$ are increased, the accuracy of the results increases steadily and that $\mathbf{r_r}$ and $\mathbf{r_a}$ are increased, the accuracy increases for the hybrid algorithm, obtaining lower error percentage as compared to the original algorithm.

V. CONCLUSION

In this paper, we present adaptive sampling-based techniques for the approximate answering of ad hoc aggregation queries in P2P databases. Our approach requires a minimal number of messages sent over the network and provides tunable parameters to maximize performance for various network topologies.

We present an I/O-efficient technique based upon a multi resolution wavelet decomposition that yields an approximate and space-efficient representation of the data cube. We get excellent approximations for on-line range-sum queries with limited space usage and computational cost. One drawback of our current approach is that the construction of the wavelet decomposition is performed on the dense data cube, which may be very large. Thus to reduce the I/O cost in constructing the compact data cube, the partial sum data cube can be implicitly represented in a more compressed form, and some sparse techniques may be used to reduce the I/O in computing the wavelet decomposition.

Our approach provides a powerful technique for approximating range 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 (small cut size), it becomes increasingly difficult 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 approach.

REFERENCES

[1] Mauricio Minuto Espil, Alejandro A. Vaisman, "Aggregate Queries in Peer to Peer OLAP**"** *DOLAP'04,* November 12–13, 2004, Washington, DC, USA. Copyright 2004 ACM 1581139772/04/0011

[2] David Kempe_, Alin Dobra, and Johannes Gehrke, "Gossip-Based Computation of Aggregate Information**"** Supported by NSF Grants IIS-0133481 and CCR-0205452, and by gifts from Microsoft and Intel

[3] Antony Rowstron1 and Peter Druschel "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems**"** Appears in Proc. of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001). Heidelberg, Germany, November 2001

[4] Panos Kalnis, Wee Siong Ng, Beng Chin Ooi, Dimitris Papadias, Kian-Lee Tan "An Adaptive Peer-to-Peer Network for Distributed Caching of OLAP Results**"** *ACM SIGMOD* '2002 June 4-6, Madison, Wisconsin, USA Copyright 2002 ACM 1-58113-497-5/02/06

[5] Dimitrios Tsoumakos, Nick Roussopoulos, "Adaptive Probabilistic Search (APS) for Peer-to-Peer Networks" CS-TR-4451, UMIACS-TR-2003-21 February 25, 2003

[6] Jeffrey Scott Vitter, Min Wang, Bala Iyer, "Data Cube Approximation and Histograms via Wavelets"

[7] Qin Lv, Pei Cao, Edith Cohen, "Search and Replication in Unstructured Peer to Peer Networks" *ICS'02,* June 2226,2002, New York, New York, USA Copyright 2002 ACM 1581134835/02/0006

[8]Valerie King, Jared Saia "Choosing a Random Peer**"** *PODC'04,* July 25–28, 2004, St. Johns, Newfoundland, Canada.Copyright 2004 ACM 1581138024/04/0007

[9] M.M. Espil and A.A. Vaisman, "Aggregate Queries in Peer-to-Peer OLAP," Proc. 7th ACM Int'l Workshop Data Warehousing and On-Line Analytical Processing (DOLAP '04), 2004.

[10] Y.Matias, J.S. Vitter, and M. Wang, Wavelet based histograms for selectivity estimation. In Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, Seattle, Washington, June 1998.

[11] V. Poosala and Y. E. Ioannidis. Selectivity estimation without the attribute value independence assumption. In Proceedings of the 1997 International Conference on Very Large Databases, Athens, Greece, August 1997.

[12] S. Sarawagi and M. Stonebraker. Efficient organization of large multidimensional arrays. In Proceedings of the 11th Annual IEEE Conference on Data Engineering (ICDE '94),Houston, Texas, 1994.

[13] 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.

[14] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip and Mixing Times of Random Walks on Random Graphs," Proc. IEEE INFOCOM '05, 2005.

[15] Y. Chu, S. Rao, and H. Zhang, "A Case for End System Multicast," Proc. ACM Int'l Conf. Measurement and Modeling of Computer Systems (SIGMETRICS '00), 2000.

[16] C. Gkantsidis, M. Mihail, and A. Saberi, "Random Walks in Peer to Peer Networks," Proc. IEEE INFOCOM '04, 2004.

[17] JUNG Web Site, http://jung.sourceforge.net, 2010.

[18] 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.

[19] V. King and J. Saia, "Choosing a Random Peer," Proc. 23rd Ann. ACM Symp. Principles of Distributed Computing (PODC '04), 2004.

[20] R. Heubsch, J. Hellerstein, N. Lanhan, B.T. Loo, S. Shenker, and I.Stoica, "Querying the Internet with PIER," Proc. 29th Int'l Conf.Very Large Data Bases (VLDB '03), 2003.

[21] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," Proc. IFIP/ACM Int'l Conf. Distributed Systems Platforms (Middleware '01),2001.

[22] D. Zeinalipour-Yazti, V. Kalogeraki, and D. Gunopulos, "Exploiting Locality for Scalable Information Retrieval in Peer-to-Peer Networks," Information System, vol. 30, no. 4, pp. 277-298, 2005.