

Efficient Data Sampling in Heterogeneous Peer-to-Peer Networks

Benjamin Arai, Song Lin, Dimitrios Gunopulos
Department of Computer Science & Engineering
University of California, Riverside
{barai, slin, dg}@cs.ucr.edu

Abstract

Performing data-mining tasks such as clustering, classification, and prediction on large datasets is an arduous task and, many times, it is an infeasible task given current hardware limitations. The distributed nature of peer-to-peer databases further complicates this issue by introducing an access overhead cost in addition to the cost of sending individual tuples over the network. We propose a two-level sampling approach focusing on peer-to-peer databases for maximizing sample quality given a user-defined communication budget. Given that individual peers may have varying cardinality we propose an algorithm for determining the optimal sample rate (the percentage of tuples to sample from a peer) for each peer. We do this by analyzing the variance of individual peers, ultimately minimizing the total variance of the entire sample. By performing local optimization of individual peer sample rates we maximize approximation accuracy of the samples. We also offer several techniques for sampling in peer-to-peer databases given various amounts of known and unknown information about the network and its peers.

1 Introduction

Storage requirements for data warehousing has been increasing rapidly due to the large amounts of data collected on a daily basis for tasks such as user logging, data-mining, and other data-intensive applications. With the continual increase in the amount of data stored in databases, information is becoming increasingly distributed. Specifically, data is quickly outgrowing single system databases and expanding to larger, distributed systems (e.g., Google Bigtable [9] & OceanStore [7, 24]). Traditional data sampling techniques have proven invaluable for obtaining fast approximate results [17, 4, 10, 12, 11], particularly where the exact solution cannot be obtained in a practical amount of time due to sheer size or churn within the database. In addition, individual users require fast and accurate responses to queries regardless of the size of the database; these constraints have introduced a need to produce accurate esti-

mates via random sampling.

It has become apparent that sampling is becoming a crucial component in obtaining efficient and accurate samples for large, distributed databases. Some of the most notable distributed systems include the eMule [15] and Gnutella [19] technologies. Peer-to-peer topologies can be partitioned into two distinct subgroups viz. *structured*¹ and *unstructured*² topologies.

The goal of random sampling is to obtain a subset of the database such that the distribution of the sample is an accurate representation of the distribution of the entire database. This can be difficult for distributed databases where there is varying connectivity and heterogeneity (data and distribution among individual peers may differ) among the data belonging to different peers. Traditional databases have the luxury of selecting data in blocks³ from the database at random due to the random access capabilities of disk storage. Alternately, for distributed databases the issue of obtaining random blocks (for distributed databases this is synonymous with peers) is not as straightforward. In order to obtain unbiased random samples we need to ensure that our selection of peers are unbiased but also that the selection of individual tuples within peers is unbiased as well. We do this through a two-level sampling approach by simultaneously optimizing the cost of selecting peers and minimizing the variance of samples within individual peers.

Our goal is to approximate aggregates such as the average or PDF (Probability Density Function) using histograms from a sample. For example, given a sample size X we can estimate the average for attribute Y from a random sample of size of X . Our approach is not application dependent; the average function above can be replaced with other aggregates or data-mining tasks such as summation, count, histogram, Data Cube [20], correlated aggregates [1], and so on.

¹Structured topologies are networks such that data is organized into specific peers and contains state information for efficient retrieval of data.

²Unstructured topologies are networks such that the location of specific data is unknown and there is no state information for individual peers.

³A block refers to a fixed size contiguous piece of data.

In this paper, we explore new techniques for sampling large distributed databases with high accuracy. We focus on providing high accuracy samples utilizing the fact that each peer contains a unique cardinality and distribution of tuple values. Consider the following example, suppose we have a distributed database composed of two peers. The first peer $peer_1$ contains two tuples with values $\{1, 1\}$ and the second peer $peer_2$ contains eight tuples $\{9, 9, 9, 9, 9, 9, 9, 9\}$. Our goal is to obtain a sample such that we minimize the variance of the final result. Naively, we can sample 50% of the tuples from each peer and estimate an aggregate query. For the average aggregate this would give us the following values $AVG(t_1) = 1$ and $AVG(t_2) = 9$ for a final aggregate value of 5. The actual average is 7.4. The problem with this approach is that it fails to consider the cardinality and the variance of data within individual peers. We can improve upon this naive approach if we analyze individual peers in such a way to assign each peer an independent sample rate such that it minimizes the total error of the sample. Our goal is to sample at such a rate so that the sampling quality is maximized meeting the user-defined accuracy or communication budget. We do this by analyzing the variance of $peer_1$ and $peer_2$ setting individual sample rates for each peer to maximize the accuracy of the final result.

Our Contributions: The main contributions of this paper are summarized as follows:

- We initiate research on two-level sampling in heterogeneous peer-to-peer networks demonstrating the unique challenges associated with our approach.
- We propose an efficient two-level sampling algorithm for distributed peer-to-peer databases.
- Our experimental results demonstrate the efficiency and effectiveness of our approach.

The rest of this paper is organized as follows. We describe related sampling techniques in Section 2 and formulate our problem in Section 3. We introduce sampling techniques associated in distributed networks in Section 4 and present our solution in Section 5. We provide extensive experimental evaluations of our technique in Section 6, and finally in Section 7 we conclude this paper.

2 Related Work

Traditional database sampling techniques [25] have received significant attention with the emergence of distributed databases and ever expanding data sets that need to be stored and accessed efficiently. Methods for selecting nodes at random from distributed networks (i.e., peer-to-peer) have been amply researched [6, 18, 22, 3]. These techniques impose a Markov chain random walk from the requesting node. In addition, it has been show that if certain structural elements of the network are known (second

eigenvalue) then a sample can be selected from a stationary distribution with a high probability.

Structured topologies such as Chord [28] and Pastry [26] contain routing mechanisms that can easily be mapped to well known sampling techniques for traditional databases such as [21, 8]. Regretfully, these techniques cannot be easily extended to more complex distributed databases because they do not account for the additional overhead incurred from sampling from multiple peers or other possible constraints due to routing. Conversely, unstructured topologies such as Gnutella [19] offer no routing or state information. Unstructured topologies have no straightforward approach for obtaining random samples. This makes it increasingly difficult to obtain high quality samples from distributed databases. Sampling techniques for unstructured topologies have also been proposed by [23, 14, 2]. Finally, related to our work is the work of [5, 27], which also deals with the problem of sampling and query evaluation in p2p networks.

Stratified Sampling [13] is an efficient technique for sampling populations. The algorithm works as follows, it begins by separating the data into separate homogeneous groups (e.g., grouping by city, zip-code, etc). The algorithm takes a random sample from each group (stratum). The final aggregate is computed as the weighted combination of each of the partial aggregates collected. *Stratified Sampling* can increase the accuracy of results by employing *Neyman allocation* [13], which allows for more samples to be taken from stratum with higher variance and from stratum of larger size. However, *Stratified Sampling* needs to access and perform sampling from all the strata (this could be disk blocks, peers etc) in the database. This will result in large communication consumption especially when the accessing cost for individual stratum is much larger than the cost of retrieving a tuple within a stratum. With the same communication budget, it may be more helpful to sample many tuples from few strata than to sample much less tuples from more peers. Though applicable in traditional databases, stratified sampling cannot be easily extended into the peer-to-peer domain due to the large cost for accessing individual peers.

Bi-level Bernoulli Sampling [21] performs sampling among database pages and sampling within disk pages (disk pages can be seen as peers or sensors in distributed networks). The goal of *Bi-level Bernoulli Sampling* is to combine page-level and row-level sampling schemes to provide the user with a trade-off between speed and statistical precision through the use of various user-defined parameters. However, it assumes each node has the same number of tuples and similar distribution of data. This is unrealistic for distributed networks such as peer-to-peer, where the data and distribution may vary between peers.

Symbol	Definition
T	Set of tuples in the network
U	Set of peers in the network
P_j	Set of tuples in Peer j
n_j	Number of tuples in Peer j
v_i	Value of i th tuple in the network
α_j	Sum of all tuples v_i in P_j , $\alpha_j = \sum_{i \in P_j} v_i$.
w_i	Variable equals 1 if tuple v_i satisfied predicate and equals 0 otherwise
β_j	Sum of w_i values in P_j , $\beta_j = \sum_{i \in P_j} w_i$.
h_{ij}	Size of bucket B_i in the histogram for P_j
h_i	Size of bucket B_i in the histogram for all the data in T
B	Sampling cost budget
C_j	The cost to access peer j
c	The cost to sample a tuple within a peer
p_j	Optimal peer-level sampling probability for Peer j
r_j	Optimal tuple-level sampling rate within Peer j
$T^{(S)}$	Set of tuples in sample
$U^{(S)}$	Set of peers in sample
$P_j^{(S)}$	Set of sample tuples in P_j

Table 1. Symbol Description

3 Problem Definition

Suppose there are n peers in a network $D = \{peer_0, \dots, peer_n\}$, with $peer_j$ having n_j data tuples in its local database. The communication cost of reaching the peer $peer_j$ from the query peer is C_j , and the cost of retrieving a tuple within a peer is c (the cost of retrieving a tuple from a peer is constant for all peers). We can simply define the cost function for data sampling as the sum of the cost for reaching each peer plus the cost of selecting tuples from individual peers.

$$Cost = \sum_{i \in U^{(S)}} (C_j + m_j c)$$

where $U^{(S)}$ is defined as the selected peers and m_i is the number of tuples selected from each peer. Our optimal two-level sampling problem can be described as follows:

Optimal Two Level Sampling Problem *Given a network of n peers, the communication cost to reach individual peers, the cost of retrieving a tuple within a peer, and a sampling budget B , our goal is to sample a subset of tuples from a subset of peers such that we can approximate aggregate queries on the network with minimal approximation error while constraining the communication cost for processing the query to the pre-defined sampling budget B .*

For better understanding of our approaches, we organize the symbols used throughout the paper in Table 1.

4 Sampling Schemes

In this section we discuss the ideas behind our approach for obtaining random samples from peer-to-peer databases.

The details of our algorithm are described in section 5.

Sampling schemes such as *Row-Level Bernoulli Sampling* (RLBS) and *Page-Level Bernoulli Sampling* (PLBS) offer powerful techniques for obtaining random samples from traditional databases. RLBS samples each tuple in a database with probability p . If the database is stored on disk we can assume that the data is retrieved one page at a time. The main drawback for RLBS is the high I/O cost. Retrieving a random row from the database will incur a full page to be read from disk. Suppose that a database consist of 100,000 tuples and each disk page can store a total of 1,000 tuples. Our goal is to randomly sample 100 tuples from the database, using RLBS, if each tuple selected is located on the different disk page this process may require 100 pages to be read, which will require the entire database to be read from the disk.

PLBS offers a more efficient sampling scheme but at the cost of sample quality. Unlike RLBS, PLBS samples fewer pages from the database but includes the entire contents of the selected pages as samples. PLBS samples each disk page in the database with probability p and exclusion of a page with probability $1 - p$. Considering our previous example, suppose we want to sample 100 tuples. In this example we need to sample only one disk page since in PLBS for each page sampled the entire contents are included. The drawback of this approach is that it does not truly obtain random samples. The disk pages selected are unbiased but the tuples contained in each page are not. In fact, if tuples are clustered on disk (which is most likely the case for increased database performance), the tuples will be distributed among pages in sorted order.

Bi-Level Bernoulli Sampling (BLBS) [21] attempts to merge RLBS and PLBS to address both sample quality and performance. They do this by offering a tradeoff between RLBS and PLBS. The drawback of this approach is that it assumes each disk page has the same number of tuples and similar data distributions. This is unrealistic for distributed networks such as peer-to-peer, where individual peers may contain varying amounts of data with distribution unique only to that peer.

In peer-to-peer databases these assumptions can produce less desirable results. Specifically, in a peer-to-peer databases such as Gnutella the variance between peers may vary greatly depending upon individual user preferences. For peer-to-peer it is desirable to sample in such a way that each peer may be sampled at an independent rate defined on a per peer basis.

5 Optimal Two-Level Sampling

In this section, we present a two-level optimal sampling algorithm for a variety of aggregate queries. The general idea of our approach is to first query a few peers in the network (peer-level sampling) and then sample a few tu-

ples from the peers we have selected (tuple-level sampling). We demonstrate how our algorithm performs sampling from both the peer pool and from each peer's data set in order to maximize the approximation accuracy while constraining the query communication cost within a user-defined budget.

5.1 SUM Queries

In our two-level sampling approach, the tuples at individual peers will be sampled with distinct probabilities. If we sample each tuple v_i with probability $\pi_i = p_j \cdot r_j$, according to Horvitz-Thompson estimator, we can obtain an estimate of the total sum α by

$$\hat{\alpha} = \sum_{j \in U^{(S)}} \sum_{i \in T^{(S)}} \frac{v_i}{p_j \cdot r_j}. \quad (1)$$

Theorem 1 $\hat{\alpha}$ is an unbiased estimator of the summation query α .

Proof: We can rewrite the exact summation of all the tuples as $\alpha = \sum_{j \in U} \alpha_j$, where $\alpha_j = \sum_{i \in P_j} v_i$. Set $I_j = 1$ if peer j is in the sample and $I_j = 0$ otherwise, we observe that

$$\begin{aligned} E[\hat{\alpha}] &= \sum_{j \in U^{(S)}} \sum_{i \in T^{(S)}} \frac{v_i}{p_j \cdot r_j} = E\left[\sum_{j \in U^{(S)}} \frac{\hat{\alpha}_j}{p_j}\right] \\ &= \sum_{j \in U} \frac{E[\hat{\alpha}_j I_j]}{p_j} = \sum_{j \in U} \frac{E[\hat{\alpha}_j] E[I_j]}{p_j} \\ &= \sum_{j \in U} \alpha_j = \alpha \end{aligned}$$

To derive the variance of $\hat{\alpha}$, we use the variance decomposition formula as follows,

$$\text{Var}[X] = \text{Var}[E[X|Y]] + E[\text{Var}[X|Y]]$$

where X and Y are random variables. If we take $X = \hat{\alpha}$ and $Y = U^{(S)}$, we get

$$\text{Var}[\hat{\alpha}] = \text{Var}[E[\hat{\alpha}|U^{(S)}]] + E[\text{Var}[\hat{\alpha}|U^{(S)}]] \quad (2)$$

In the above formula,

$$\begin{aligned} \text{Var}[E[\hat{\alpha}|U^{(S)}]] &= \text{Var}\left[E\left[\sum_{j \in U^{(S)}} \frac{\hat{\alpha}_j}{p_j}\right]\right] \\ &= \text{Var}\left[E\left[\sum_{j \in U^{(S)}} \frac{\hat{\alpha}_j}{p_j}\right]\right] \\ &= \text{Var}\left[\sum_{j \in U^{(S)}} \frac{\alpha_j}{p_j}\right] \\ &= \sum_{j \in U} \left(\frac{1}{p_j} - 1\right) \alpha_j^2. \end{aligned}$$

and

$$\begin{aligned} E[\text{Var}[\hat{\alpha}|U^{(S)}]] &= E\left[\text{Var}\left[\sum_{j \in U^{(S)}} \frac{\hat{\alpha}_j}{p_j}\right]\right] \\ &= E\left[\sum_{j \in U^{(S)}} \text{Var}\left[\frac{\hat{\alpha}_j}{p_j}\right]\right] \\ &= \sum_{j \in U} p_j \text{Var}\left[\frac{\hat{\alpha}_j}{p_j}\right] \\ &= \sum_{j \in U} \frac{1}{p_j} \text{Var}[\hat{\alpha}_j] \\ &= \sum_{j \in U} \frac{1}{p_j} \cdot \left(\frac{1}{r_j} - 1\right) \sum_{i \in P_j} v_i^2. \end{aligned}$$

Therefore,

$$\text{Var}[\hat{\alpha}] = \sum_{j \in U} \left(\frac{1}{p_j} - 1\right) \alpha_j^2 + \sum_{j \in U} \frac{1}{p_j} \cdot \left(\frac{1}{r_j} - 1\right) \sum_{i \in P_j} v_i^2 \quad (3)$$

Since the sampled estimate $\hat{\alpha}$ is an unbiased estimator of the real summation query α , the variance of $\hat{\alpha}$ is then an indicator of how accurate the sample can become. On average, the smaller the variance of $\hat{\alpha}$ the more accurate the estimation while; the larger a variance the less accurate the estimation. The next problem is then to minimize the variance of the approximation $\hat{\alpha}$ which is described as follows,

Problem 1 Given all the tuple values v_i from the peers, and the expected peer sample cost B , the objective is to find the best sampling probability p_j for each peer, in such a way that the variance of the sampling result is minimized.

Minimize

$$\text{Var}[\hat{\alpha}] = \sum_{j \in U} \left(\frac{1}{p_j} - 1\right) \alpha_j^2 + \sum_{j \in U} \frac{1}{p_j} \cdot \left(\frac{1}{r_j} - 1\right) \sum_{i \in P_j} v_i^2$$

subject to

$$\sum_{j \in U} (C_j p_j + c n_j r_j p_j) \leq B. \quad (4)$$

We can solve this constrained optimization problem by utilizing the Lagrange Multipliers method. The solution for this problem has a closed form as follows,

$$p_j = \frac{B \sqrt{\frac{\tau_j}{C_j}}}{\sum_{j \in U} \left(\sqrt{\tau_j C_j} + \sqrt{c n_j \pi_j}\right)}, \quad r_j = \sqrt{\frac{C_j \pi_j}{c \tau_j n_j}} \quad (5)$$

where $\tau_j = \alpha_j^2 - \sum_{i \in P_j} v_i^2$ and $\pi_j = \sum_{i \in P_j} v_i^2$. The parameter v_i is the value of the i th tuple and α_j is the summation of all tuples in the j th peer.⁴ It can be concluded from

⁴In our settings, we assume all the tuple values v_i are positive ($\forall i \in T, v_i > 0$).

equation (5) that in the optimal two-level sampling solution, the tuple level sampling rate r_j within peer j only depends on the local information of peer j (i.e. peer access cost C_j , tuple values v_i , number of tuples n_j in peer j), while the peer level sampling probability p_j not only depends on local information at peer j (i.e. τ_j, C_j) but also the information from all other peers (i.e. $\sum_{j \in U} (\sqrt{\tau_j C_j} + \sqrt{cn_j \pi_j})$).

5.2 COUNT Queries

Similar to SUM queries, we can estimate the COUNT β of all the tuples by

$$\hat{\beta} = \sum_{j \in U(s)} \sum_{i \in T(s)} \frac{w_i}{p_j \cdot r_j}. \quad (6)$$

and the variance of our estimation can be represented as

$$\text{Var}[\hat{\beta}] = \sum_{j \in U} \left(\frac{1}{p_j} - 1 \right) \beta_j^2 + \sum_{j \in U} \frac{1}{p_j} \cdot \left(\frac{1}{r_j} - 1 \right) \beta_j$$

We can then formulate a similar optimal sampling problem as **Problem 1**, and solve the problem using the Lagrange Multipliers method. The optimal solution of the optimal sampling for count queries has a close form as follows,

$$p_j = \frac{B \sqrt{\frac{\tau_j}{C_j}}}{\sum_{j \in U} (\sqrt{\tau_j C_j} + \sqrt{cn_j \beta_j})}, \quad r_j = \sqrt{\frac{C_j \beta_j}{c \tau_j n_j}} \quad (7)$$

where $\tau_j = \beta_j^2 - \beta_j$. As described in Table 1, $\beta_j = \sum_{i \in P_j} w_i$, is the summation of all the predicate values of the data at peer P_j .

5.3 Histogram Queries

To construct a histogram of all the data in the peer-to-peer network, we first find the domain D of the data tuples ($D = [LB, UB]$, where LB and UB are the lower bound and upper bound of the data separately) and then compute a sequence of separators $s_0, s_1, \dots, s_b \in D$ ($s_i = LB + (UB - LB) * i/b$). These separators partition D into b even-width and non-overlapping buckets B_1, B_2, \dots, B_b where $B_j = \{v \in D | s_{j-1} \leq v \leq s_j\}$. Let \hat{h}_i be the estimated size of (i.e. the number of records in) bucket B_i by our approximation, and h_i be the actual size of B_i . According to [16], we can use the variance-error metric (the mean square error across all buckets, normalized with respect to the mean bucket size) to measure approximation error of the histogram,

$$\Delta^2 = E \left[\frac{b}{n^2} \sum_{i=1}^b (\hat{h}_i - h_i)^2 \right] \quad (8)$$

We can represent these histogram queries as b different count queries (i.e. $count_1, \dots, count_b$) in which each count query computes the number of tuples within a bucket of the histogram (i.e. $count_i$ computes the bucket size of B_i). Similar to sum queries, the two-level sampling approximation \hat{h}_i of count query $count_i$ is an unbiased estimation for bucket size h_i ,

$$h_i = E(\hat{h}_i) \quad (9)$$

Then we can rewrite the approximation error for histogram (i.e. equation (8)) as,

$$\begin{aligned} \Delta^2 &= E \left[\frac{b}{n^2} \sum_{i=1}^b (\hat{h}_i - h_i)^2 \right] \\ &= \frac{b}{n^2} \sum_{i=1}^b E \left[(\hat{h}_i - h_i)^2 \right] \\ &= \frac{b}{n^2} \sum_{i=1}^b \text{Var}[\hat{h}_i] \end{aligned}$$

where $\text{Var}[\hat{h}_i] = \sum_{j \in U} (\frac{1}{p_j} - 1) h_{ij}^2 + \sum_{j \in U} \frac{1}{p_j} \cdot (\frac{1}{r_j} - 1) h_{ij}$.

The optimal sampling problem for histograms can be formulated as

Minimize $\Delta^2 = \frac{b}{n^2} \sum_{i=1}^b \text{Var}[\hat{h}_i]$, subject to the equation $\sum_{j \in U} (C_j p_j + cn_j r_j p_j) - B = 0$.

We can also solve this constrained optimization problem by Lagrange Multipliers, which has a close form solution as follows,

$$p_j = \frac{B \sqrt{\frac{\tau_j}{C_j}}}{\sum_{j \in U} (\sqrt{\tau_j C_j} + \sqrt{cn_j \pi_j})}, \quad r_j = \sqrt{\frac{C_j \pi_j}{c \tau_j n_j}} \quad (10)$$

where $\tau_j = \sum_{i=1}^b h_{ij}^2 - \sum_{i=1}^b h_{ij}$ and $\pi_j = \sum_{i=1}^b h_{ij}$. As described in Table 1, h_{ij} is the size of bucket B_i in the histogram for peer P_j .

5.4 Approximating Statistics at Different Peers

In most cases, the information about the tuple value v_i distributed at different peers is unknown *a priori*. A possible way to estimate this information is to use a pilot sampling technique. The general idea is to let each peer perform a small pilot sampling S' using a small sampling rate r' ($r' < r$) first, and using these pilot samples to estimate the tuple v_i at all the different peers.

$$\hat{\alpha}'_j = \frac{1}{r'} \sum_{i \in P_j(S')} v_i. \quad (11)$$

$$\sum_{i \in P_j} \hat{v}_i^2 = \frac{1}{r'} \sum_{i \in P_j^{(s')}} v_i^2. \quad (12)$$

With the estimator $\hat{\alpha}'_j$ and $\sum_{i \in P_j} \hat{v}_i^2$ for the data at different peers, we can compute the optimal two-level sampling plan for the peer-to-peer network.

5.5 Random Walks in Peer-to-Peer

So far we have presented our optimal two-level sampling algorithms to compute the optimal sampling plan for peer-to-peer networks. The optimal sampling plan includes the peer-level sampling probability for each individual peer and the tuple-level sampling rate within each peer. Next we discuss how to implement the optimal sampling plan in a peer-to-peer network.

Efficiently obtaining a random sample of peers from a peer-to-peer network is a difficult task considering the degree⁵ of individual peers may vary greatly. Obviously, if we perform a random walk in the network, individual peers with higher degree have a higher probability of being visited than peers with lower degree. This probability can be computed for each peer as

$$Prob(peer_j) = \frac{degree(peer_j)}{2|E|}$$

where $Prob(peer_j)$ is a function of the degree of the peer $peer_j$ and $|E|$ the total number of edges in the network.

The length of the random walk can greatly affect the latency of a given query. For example, if the network contains two clusters of peers connected by only one edge then the random walk must continue for an extended number of hops to ensure each peer has an equal probability of selection. Conversely, if the network is well connected (expander) then the random walk may be shorter since only a few hops are required to reach any peer within the network.

The solution to this problem is *Markov chain random walks* [18]. Briefly, the *Markov chain random walks* start a walk from the query peer. For each step of the random walk the goal is to select each peer landed upon with the same probability. From graph theory we know that if a random walk is continued for a sufficient length, the probability of selecting any peer reaches a stationary distribution. An adequate length for the random walk is determined as a pre-processing step and the results may be used for several queries.

In order to draw a random sample of peers from the network each peer must have the same probability of being included in the sample, we can satisfy this constraint by performing a random walk in the peer-to-peer network and including each visited peer into the sample with probability

$$Prob(peer_j) = \frac{1}{degree(peer_j)}$$

where $Prob(peer_j)$ is a function of the degree of the peer $peer_j$. As a result, for each step of the random walk, a peer is sampled with the same probability.

Due to the heterogeneity of the individual peers (i.e. difference in the size and distribution of data in individual peers), the sampling probability for individual peers tend to differ. We therefore revise the *Markov chain random walks* to address the per peer sampling probability by introducing a weighting factor. With the assigned probability p_j to sample each individual peer $peer_j$, we perform a random walk in the peer-to-peer network and include the visited peer $peer_j$ into the sample with probability

$$Prob(peer_j) = \frac{1}{degree(peer_j)} * \frac{p_j}{\sum_{i \in U} p_i}$$

Once the optimal two-level sampling plan is computed by the query peer, we can use the revised *Markov chain random walks* definition described above to sample individual peers with the assigned probability and draw tuple samples from the selected peers accordingly.

6 Experimental Evaluation

In this section, we present an extensive performance evaluation of our **Two-level Optimal Sampling** algorithm and other sampling techniques in peer-to-peer networks using two real world data sets. Our goal is to demonstrate that our **Two-level Optimal Sampling** algorithm can efficiently approximate aggregate queries in peer-to-peer networks with high accuracy. We have implemented a trace driven simulator in GNU C++. The simulation was performed on a PC with an Intel Pentium 4 (2.8 GHz) CPU and 1 GB of memory.

6.1 Data sets

We use synthetic data sets generated by a Zipf generator. Each data set includes 100,000 data records each having a random value ranging from 0 to 1000. The parameters that determine the characteristics of each data set are summarized as below,

- *NetworkSize* - the number of peers in the network. In our experiments, we vary the network size from 100 to 1000 peers.
- *ClusterRate* - the data cluster rate among peers. The *ClusterRate* controls the uniformity of data distributed among peers. When *ClusterRate* = 0, all the data is distributed randomly among peers. On the other hand, if *ClusterRate* = 1, the data is distributed

⁵The number of edges connecting a peer to other peers within the network.

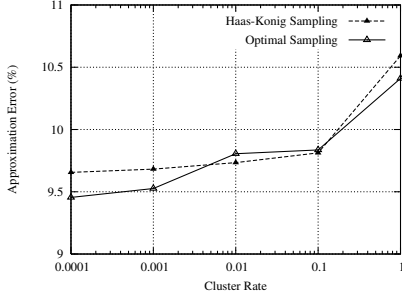


Figure 1. Approximation error for SUM query with varied cluster rate.

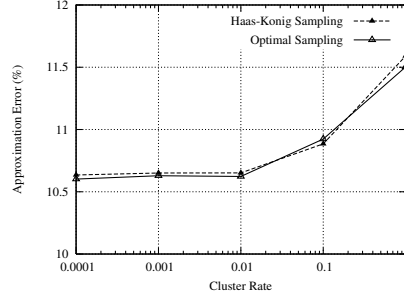


Figure 2. Approximation error for COUNT query with varied cluster rate.

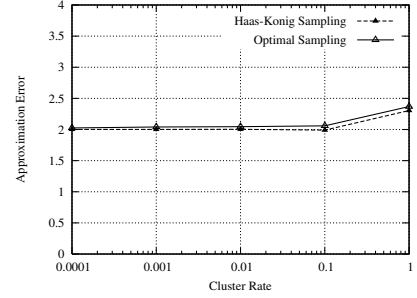


Figure 3. Approximation error for HISTOGRAM query with varied cluster rate.

in sorted order (i.e., any tuple in $peer_i$ will always be smaller (or larger) than any tuple in $peer_j$).

- α - the skew parameter for Zipf generator. The Zipf generator is used to generate data conforming to the Zipf distribution. In the Zipf distribution, the probability for the occurrence of the i th largest value is inversely proportional to i^α (i.e. $Prob(v_i) \propto i^{-\alpha}$).
- *DistributionMode*. The *DistributionMode* determines how the generated data set is distributed among peers. **Uniform Distribution** represents a uniform division of the data set across the peers. In other words, in **Uniform Distribution**, each peer has the same number of tuples. On the other hand, in **Biased Distribution**, the cardinality of individual peers are different, which is more representative of the real world scenarios.

6.2 Comparison Systems

We evaluate and compare the performance of our Optimal Two Level Sampling algorithm and Haas-Konig Sampling algorithm [21] for aggregate query processing in peer-to-peer networks:

Optimal Sampling: This is the optimal two-level sampling algorithm we described in Section 5. Optimal Sampling assigns peer-level sampling probabilities and tuple-level sampling rates to individual peers, adjusting to the heterogeneity of the data distribution in the network and greatly increase the sampling accuracy.

Haas-Konig Sampling: This sampling approach also performs sampling among peers and sampling within individual peers. Haas-Konig Sampling assumes each peer has the same number of tuples and similar distribution of data and therefore cannot adjust itself efficiently to heterogeneous peer-to-peer networks.

6.3 Query Types

We report our experimental results using three types of aggregate queries (i.e., SUM, COUNT, HISTOGRAM). As

stated previously our approach can be easily be extended to more complex aggregates such as Data Cubes, correlated aggregates, and so on.

SUM Queries: The SUM aggregate is used to determine the summation of the all the tuple values in the network, which can easily be extended to answer AVERAGE and COUNT queries. For SUM queries, we measure the approximation accuracy in terms of relative error.⁶

COUNT Queries: The COUNT query computes the number of tuples that satisfy some user defined predicate. For example, how many data records have temperature value larger than $90^\circ F$. For COUNT queries, we also measure the approximation accuracy in terms of relative error.

HISTOGRAM Queries: The HISTOGRAM query constructs a histogram utilizing the sensor data sampled from the network. The approximation error for histogram queries is defined by Equation (8).

6.4 Clustering Data Among Peers

In the first set of experiments, we want to evaluate the impact of data clustering on the performance of the Haas-Konig Sampling technique and our Optimal Sampling algorithm. In real world peer-to-peer networks, we usually observe homogeneity among the data distributed within each peer and heterogeneity among the data belongs to different peers. We simulate this scenario by introducing data clustering into the synthetic data set. As mentioned in Section 6.1, we use the parameter *ClusterRate* to represent the heterogeneity of the data at different peers. We generated 100,000 random tuples and 1,000 peers (each with 100 tuples) and compared the performance of Haas-Konig Sampling and our Optimal Sampling algorithm by varying the clustering parameter *ClusterRate*. As shown in figures 1, 2 and 3, as the cluster rate increases from 0.0001 to 1, the approximation error for both Haas-Konig Sampling and Optimal Sampling increase. This is because as the value of

⁶Given a value x and its estimator x' , the relative error of the estimation is defined as $|\frac{x-x'}{x}| \times 100\%$.

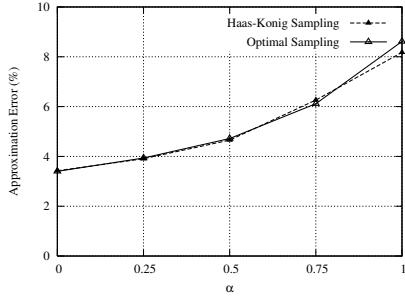


Figure 4. Approximation error for SUM query with Zipf generator.

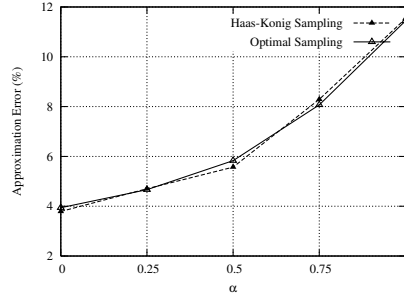


Figure 5. Approximation error for COUNT query with Zipf generator.

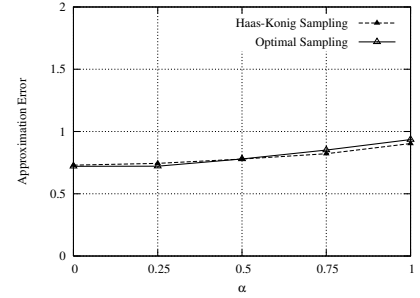


Figure 6. Approximation error for HISTOGRAM query with Zipf generator.

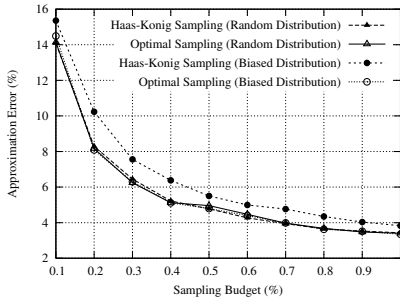


Figure 7. Approximation error for SUM query with varied sampling budget.

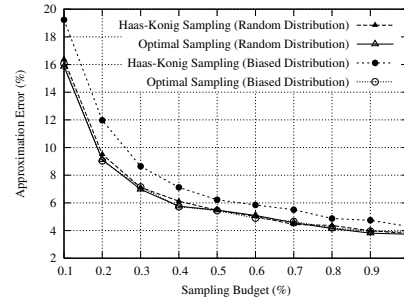


Figure 8. Approximation error for COUNT query with varied sampling budget.

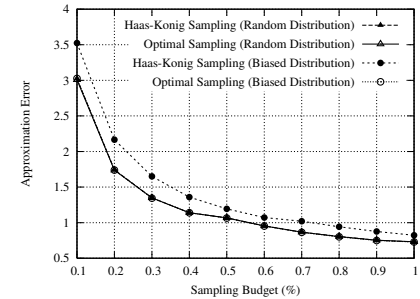


Figure 9. Approximation error for HISTOGRAM query with varied sampling budget.

ClusterRate increases the variance among tuples at different peers increase as well causing peer-level samples to be more skewed.

6.5 Data Generator

We also evaluate and compare the performance of Haas-Konig Sampling and Optimal Sampling on both uniform and skewed data sets. Zipf distribution is a well-known data distribution used to represent real world scenarios. The parameter α represents the skew of the generated data. $\alpha = 0$ represents uniformly generated data with each value having the same probability of occurring in the data set. We generated 100,000 random tuples and 1,000 peers (each with 100 tuples) and compared the performance of Haas-Konig Sampling and our Optimal Sampling algorithm using Zipf generated data sets varying the parameter α . As shown in figures 4, 5 and 6, as the value α increases, sampling few peers will introduce higher biased estimation, increasing the approximation error for both sampling algorithms.

6.6 Varying Sampling Budget

We first want to compare the approximation accuracies of the Haas-Konig Sampling technique and our Optimal Sampling algorithm with different sampling budgets. As shown in figures 7, 8, and 9, with increasing budget for

data sampling, the approximation accuracies increase for both Haas-Konig Sampling and Optimal Sampling. This is because we can draw a higher number of samples from the network to approximate the aggregate query more accurately. For **Uniform Distribution** where each peer has the same number of tuples, Haas-Konig Sampling and Optimal Sampling are similar in approximation accuracy as the data tuples are distributed uniformly across the peers. However, for **Biased Distribution**, which we believe is more representative of real world heterogeneous peer-to-peer networks, Optimal Sampling outperforms Haas-Konig Sampling by decreasing approximation error by 20%. This is due to the fact that Haas-Konig Sampling samples individual peers with the same probability and samples tuples within the peer with the same sampling rate. Therefore, Haas-Konig Sampling cannot adapt efficiently to the heterogeneous data distribution in the network.

6.7 Varying Network Size

With a total data size of 100,000 tuples, we fixed the energy budget to 1% of the total energy required to retrieve all the tuple in the network. As shown in figures 10, 11, and 12, Haas-Konig Sampling and Optimal Sampling show similar approximation accuracy for **Uniform Distribution**, but both have higher approximation error for **Biased Distri-**

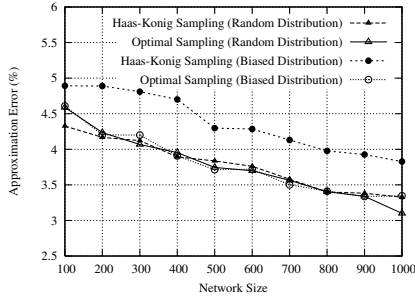


Figure 10. Approximation error for SUM query with varied network size.

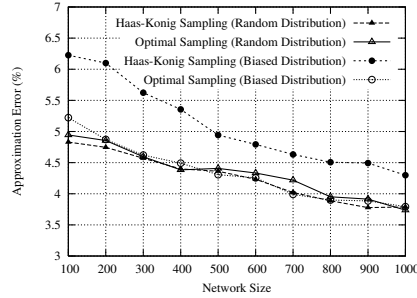


Figure 11. Approximation error for COUNT query with varied network size.

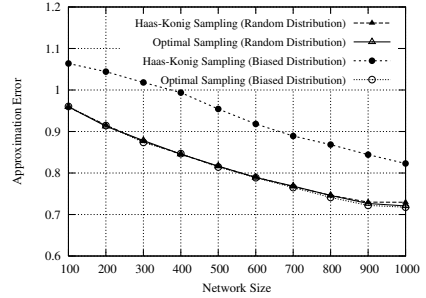


Figure 12. Approximation error for HISTOGRAM query with varied network size.

bution since the tuples distributed across individual peers show much higher variance. In addition, Optimal Sampling consistently outperforms Haas-Konig Sampling in **Biased Distribution**, as it addresses the heterogeneity of the data distribution in the network. We can also see that as the network size grows, the sampling accuracy increases as well. This is because with larger network size, the total cost to retrieving all tuples is minimized (the cost to accessing a peer is much larger than to retrieve a tuple from a peer). Therefore, we may sample more tuples from the selected peers and approximate the query more accurately.

6.8 Varying Peer Cardinality

The previous two sections show that when the cardinality of individual peers are similar, Optimal Sampling doesn't have an advantage over Haas-Konig Sampling. In real world peer-to-peer networks, the data size at individual peers tends to show large variance. Our Optimal Sampling addresses this property by sampling individual peers with different probabilities and sampling tuples within individual peers with different sampling rates in order to maximize the approximation accuracy. In the last experimental series, we evaluate how Optimal Sampling outperforms Haas-Konig Sampling when the data size of individual peers are skewed. Without loss of generality, we generate skewed data size of individual peers with the Zipf generator. As shown in figures 13, 14 and 15, as the value of α increases for the Zipf generator, the approximation error increases for both Haas-Konig Sampling and Optimal Sampling. This is because larger values of α introduce higher skew into the distribution of the data among peers and therefore making sampling less accurate. In addition, Optimal Sampling consistently outperforms Haas-Konig Sampling and the performance difference between them increases when α is larger. This is due to the fact that Haas-Konig Sampling cannot efficiently adjust the sampling plan to the heterogeneity of data distribution among different peers.

7 Conclusions

We have proposed a two-level sampling approach focusing on peer-to-peer databases for maximizing sample quality given a user-defined communication budget. The idea of our approach is to balance the tradeoff between peer-level and tuple-level sampling, which addresses the varying cardinality and the data distribution of individual peers. We do this by analyzing the variance of individual peers, ultimately minimizing the total variance of the approximation. By performing local optimization on the tuple-level sampling rates for individual peers we reduce the total number of tuples sent over the network. Our approach can be applied to a variety of statistical and data-mining tasks where the computation over the entire dataset is infeasible or impractical. We also offer several techniques for sampling in peer-to-peer databases given various levels of known and unknown information about the network and its peers. Our experiments showed that our technique is efficient and optimal for various data scenarios.

8 Acknowledgements

The work of Dimitrios Gunopulos was supported by NSF (IIS 0330481, IIS 0534781).

References

- [1] R. Ananthakrishna, A. Das, J. Gehrke, F. Korn, S. Muthukrishnan, and D. Srivastava. Efficient approximation of correlated sums on data streams. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):569–572, 2003.
- [2] B. Arai, G. Das, D. Gunopulos, and V. Kalogeraki. Approximating aggregation queries in peer-to-peer networks. *ICDE*, 2006.
- [3] A. Awan, R. A. Ferreira, S. Jagannathan, and A. Grama. Distributed uniform sampling in unstructured peer-to-peer networks. *HICSS*, 2006.
- [4] B. Babcock, S. Chaudhuri, and G. Das. Dynamic sample selection for approximate query processing. In *SIGMOD '03*:

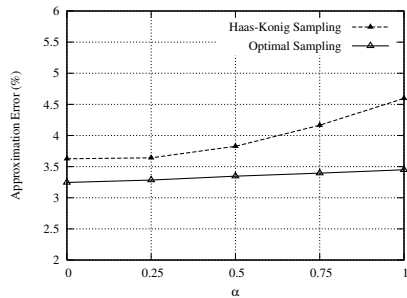


Figure 13. Approximation error for SUM query in the heterogeneous P2P network.

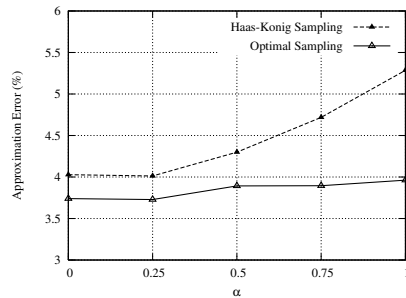


Figure 14. Approximation error for COUNT query in the heterogeneous P2P network.

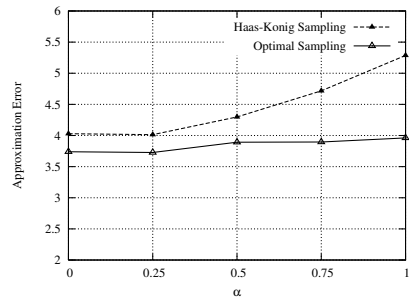


Figure 15. Approximation error for HISTOGRAM query in the heterogeneous P2P network.

Proceedings of the 2003 ACM SIGMOD international conference on Management of data, pages 539–550, New York, NY, USA, 2003. ACM Press.

- [5] F. Banaei-Kashani and C. Shahabi. Partial selection query in peer-to-peer databases. *ICDE*, 2006.
- [6] A. R. Bharambe, M. Agrawal, and S. Seshan. Mercury: supporting scalable multi-attribute range queries. In *SIGCOMM*, pages 353–366, 2004.
- [7] D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, B. Zhao, and J. Kubiatowicz. Oceanstore: An extremely wide-area storage system. In *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems*, November 2000.
- [8] P. G. Brown and P. J. Haas. Techniques for warehousing of sample data. In *Proceedings of the 22nd International Conference on Data Engineering*, page 6, 2006.
- [9] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. In *OSDI*, pages 205–218, 2006.
- [10] M. Charikar, S. Chaudhuri, R. Motwani, and V. Narasayya. Towards estimation error guarantees for distinct values. In *PODS '00: Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 268–279, New York, NY, USA, 2000.
- [11] S. Chaudhuri, G. Das, M. Datar, R. Motwani, and V. R. Narasayya. Overcoming limitations of sampling for aggregation queries. In *ICDE*, pages 534–542, 2001.
- [12] S. Chaudhuri, R. Motwani, and V. Narasayya. Random sampling for histogram construction: how much is enough? pages 436–447, 1998.
- [13] W. G. Cochran. *Sampling Techniques*. John Wiley & Sons, New York, NY, 1977.
- [14] S. Datta and H. Kargupta. Uniform data sampling from a peer-to-peer network. *ICDCS*, page 50, 2007.
- [15] eMule. <http://www.emule-project.net>.
- [16] P. B. Gibbons, Y. Matias, and V. Poosala. Fast incremental maintenance of approximate histograms. *ACM Trans. Database Syst.*, 27(3):261–298, 2002.
- [17] P. B. Gibbons, V. Poosala, S. Acharya, Y. Bartal, Y. Matias, S. Muthukrishnan, S. Ramaswamy, and T. Suel. AQUA: System and techniques for approximate query answering. Technical report, Murray Hill, New Jersey, U.S.A., 1998.
- [18] C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks. In *Proceedings of IEEE INFOCOM*, 2004.
- [19] Gnutella. <http://www.gnutella.com>.
- [20] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-total. In S. Y. W. Su, editor, *Proceedings of the Twelfth International Conference on Data Engineering, February 26 - March 1, 1996, New Orleans, Louisiana*, pages 152–159. IEEE Computer Society, 1996.
- [21] P. J. Haas and C. Konig. A bi-level bernoulli scheme for database sampling. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 275–286, 2004.
- [22] V. King and J. Saia. Choosing a random peer. In *Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pages 125–130, 2004.
- [23] W. Kowalczyk and N. Vlassis. Newscast em. *NIPS*, pages 713–720, 2004.
- [24] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of ACM ASPLOS*. ACM, November 2000.
- [25] F. Olken. Random sampling from databases. In *PhD thesis, University of California, Berkeley*, 1993.
- [26] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, Nov 2001.
- [27] C. Shahabi and F. Banaei-Kashani. Modelling peer-to-peer data networks under complex system theory. *IJCSSE*, 2006.
- [28] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *In Proceedings of ACM SIGCOMM*, August 2001.