Fixed-Precision Approximate Continuous Aggregate Queries in Peer-to-Peer Databases

Farnoush Banaei-Kashani [1], Cyrus Shahabi [2]

Department of Computer Science, University of Southern California
Los Angeles, CA 90089
[1]banaeika@usc.edu
[2]shahabi@usc.edu

***Abstract*— In this paper, we outline our efficient sample-based approach to answer fixed-precision approximate continuous aggregate queries in peer-to-peer databases. We describe our approach in the context of *Digest*, a two-tier system we have developed for correct and efficient query answering by sampling. With Digest, at the top tier we develop a query evaluation engine that uses the samples collected from the peer-to-peer database to continually estimate the running result of the approximate continuous aggregate query with guaranteed precision. For efficient query evaluation, we propose an extrapolation algorithm that predicts the evolution of the running result and adapts the frequency of the continual sampling occasions accordingly to avoid redundant samples. We also introduce a repeated sampling algorithm that draws on the correlation between the samples at successive sampling occasions and exploits linear regression to minimize the number of the samples derived at each occasion. At the bottom tier, we introduce a distributed sampling algorithm for random sampling (uniform and nonuniform) from peer-to-peer databases with arbitrary network topology and tuple distribution. Our sampling algorithm is based on the Metropolis Markov Chain Monte Carlo method that guarantees randomness of the sample with arbitrary small variation difference with the desired distribution, while it is comparable to optimal sampling in sampling cost/time. We evaluate the efficiency of Digest via simulation using real data.**

I. Motivation

A peer-to-peer database is a fragmented database which is distributed among the nodes of a peer-to-peer network, with both the data and the network dynamically changing. In this paper, we focus on answering continuous aggregate queries in peer-to-peer databases, where the underlying peer-to-peer network of the database is inherently unstructured (as opposed to DHT-based structured peer-to-peer networks). Continuous queries [1], [2], [3] allow users to obtain new results from the database without having to issue the same query repeatedly. Continuous queries are especially useful with peer-to-peer databases which inherently comprise of large amounts of frequently changing data. For example, in a weather forecast system with thousands of interconnected stations the system administrator can issue a continuous aggregate query of the form:

> *"Over next 24 hours, notify me whenever the average temperature of the area changes more than 2 ^oF."*

Or in a peer-to-peer computing system with distributed resources, users can issue the following query to determine when there is enough memory space available to schedule their tasks:

> *"Notify me whenever the total amount of available memory is more than 4GB."*

However, considering the large size and the high rate of change in peer-to-peer databases, exact continuous aggregate queries are inevitably inefficient, if not infeasible. Exact answers are rarely necessary, and even if needed, a consistent approximation can converge to the exact result with arbitrary precision. Therefore, with our study we consider *approximate* continuous aggregate queries.

II. Problem Definition

We model an unstructured peer-to-peer network as an undirected graph $G(V, E)$ with arbitrary topology. The set of vertices $V = \{v_1, v_2, \ldots, v_r\}$ represent the set of the network nodes, and the set of edges $E = \{e_1, e_2, \ldots, e_q\}$ represent the set of the network links, where $e_i = (v_j, v_k)$ is a link between v_j and v_k. As nodes autonomously join and leave the network, the member-set of V, and accordingly, that of E vary in time. Consequently, the set sizes r and q are also variable and unknown a priori. We assume the rate of the changes in G is relatively low as compared to the sampling time (i.e., the time required to draw a sample from the peer-to-peer database), such that the network can be assumed almost static during each sampling occasion (although it may change significantly between successive sampling occasions).

For a peer-to-peer database stored in such an unstructured peer-to-peer network, without loss of generality we assume a relational model. Suppose the database consists of a single relation $R = \{u_1, u_2, \ldots, u_N\}$. R (a multiset) is horizontally partitioned and each disjoint subset of its tuples is stored at a separate node. The number of tuples stored at the node v_i is denoted by m_{v_i}. The member-set of R also varies in time; the changes are either due to the changes of V, as nodes with new content join the network (as if inserting tuples) and existing nodes leave and remove their content (as if deleting tuples), or as the existing nodes autonomously modify their local content by insertion, update, and deletion.

With such a model for peer-to-peer databases, we define our basic query model for the continuous aggregate queries as follows. Consider the queries of the form:

SELECT $op(expression)$ FROM R

where *op* is one of the aggregate operations AVG, COUNT, or SUM, and $expression$ is an arithmetic expression involving the attributes of R. Suppose Q is an instance of such queries. Assuming a discrete-time model (i.e., the time is modelled as a discrete quantity with some fixed unit), the *snapshot* aggregate query Q_t is the query Q evaluated at time t. Correspondingly, the *continuous* aggregate query Q^c is the query Q evaluated continuously (i.e., repeated successively without intermission) for all $t \geq t_0$, where t_0 is the arrival time of Q^c. The result of Q^c is $X[t]$, a discrete-time function where for all $t \geq t_0$, $X[t]$ is the aggregate-value result of the snapshot query Q_t.

For instance, in a peer-to-peer computing system each node (a node consists of one or more computing units) keeps a current record of its available resources by maintaining tuples of the form $u_i = \langle \mathsf{cpu}, \mathsf{memory}, \mathsf{storage}, \mathsf{bandwidth} \rangle$, one tuple for each local computing unit. Considering $R(\mathsf{cpu}, \mathsf{memory}, \mathsf{storage}, \mathsf{bandwidth})$ as a single-relation peer-to-peer database representing the resources available in such a peer-to-peer computing system, the following continuous query returns X, the total amount of the space currently available throughout the system, as a function of time:

SELECT SUM($\mathsf{memory} + \mathsf{storage}$) FROM R

With a *fixed-precision approximate* version of the exact continuous query Q^c defined above, the exact result $X[t]$ is approximated by an estimate $\widehat{X}[t]$ with guaranteed precision.

III. Contributions

Previous approaches for approximate query answering are not applicable to peer-to-peer databases. The model based approaches [4] are parameterized, where with peer-to-peer databases parameters are unknown and variable. The histogram based [5] and the precomputed-sample based [6], [7] data reduction approaches are not appropriate either. Although dynamically updated, with the high rate of change in peer-to-peer databases maintaining histograms and precomputed samples is intolerably costly. The large set of techniques proposed for approximate continuous aggregate query over data streams [8], [9] naturally assume the data are collected centrally and are being received in sequence, where none of these assumptions hold for the data in peer-to-peer databases. Finally, the current on-the-fly sampling approaches, mostly developed for query size estimation [10], [11], [12], [13], are limited to snapshot (or one-time) aggregate queries.

We study an approach for answering fixed-precision approximate continuous aggregate queries based on on-the-fly sampling from peer-to-peer databases. Our query answering system, called *Digest*, evaluates approximate continuous aggregate queries by continual execution of the approximate snapshot aggregate queries, where each snapshot query is evaluated by sampling the database. The snapshot queries probe the database and accordingly the running[1] result of the continuous query is updated. As we elaborate below, with continuous queries the main issue transcends how to execute each snapshot query, but how to execute snapshot queries continually such that while the fixed precision requirements of the continuous query are guaranteed, the query is answered efficiently by deriving minimum number of samples.

[1]In this paper, we use the term "running" to refer to a quantity which is a function of time, in a sense different from a progressive/online (aggregate) quantity as used in [14].

With fixed-precision approximate continuous aggregate queries, the required (or fixed) precision of the approximate result is defined by the user in terms of 1) the *resolution* of the result in capturing the changes of the actual running aggregate value (e.g., the result reflects the changes of the average temperature iff a change is more than 2 ^{o}F), and 2) the *confidence* (or accuracy) of the result *at each time* as compared to the exact aggregate value at that time. Using continual snapshot queries to answer such queries, the resolution of the result is determined by the frequency of the snapshots, and the confidence of the result depends on the number of the samples derived to approximate the result of each snapshot query. Therefore, for efficient evaluation of the continuous queries while guaranteeing the fixed precision, both the frequency of the snapshot queries and the number of the samples derived at each snapshot query must be minimized while the resolution requirement and the confidence requirement of the query are still satisfied, respectively. Digest provides solutions for both of these optimization problems.

Digest is implemented as a two-tier system, with a sampling operator at the bottom tier and a query evaluation engine at the top tier. The sampling operator implements a distributed sampling algorithm to derive arbitrary random samples from the peer-to-peer databases (with arbitrary topology and tuple distribution). The query engine uses the samples collected from the database to evaluate the snapshot queries. To minimize the frequency (or equivalently, the number) of the snapshot queries, the query engine exploits an *extrapolation algorithm* that predicts the evolution of the running aggregate value based on its previous behavior and adapts the frequency of the continual snapshot queries accordingly. With this approach, the more varying the aggregate value, the more becomes the frequency of the snapshot queries in order to maintain the resolution of the result, and when the aggregate value is steady the frequency of the snapshot queries decreases accordingly to avoid redundant sampling.

On the other hand, to minimize the number of the samples derived at each snapshot query, the query engine employs a *repeated sampling algorithm*. Repeat sampling draws on the observation that across successive snapshot queries the values of the database tuples are expected to be autocorrelated and, therefore, exploiting the regression of the value of a sampled tuple at the current query on that of the previous query can improve the accuracy of the current estimate. Repeated sampling uses regression estimation to achieve the required confidence using fewer samples as compared with the straightforward independent sampling which ignores the correlation between the snapshots.

We study the extrapolation algorithm and the repeated sampling algorithm both analytically and empirically (via simulation using real data).

IV. Related Work

Previous approaches for answering continuous queries are not applicable with peer-to-peer databases. The approaches proposed for centralized databases, whether for continuous query answering [1] or materialized view maintenance [15], naturally assume all changes of the data are *given* locally, whereas with peer-to-peer databases the data is distributed and changes occur autonomously at the nodes. With continuous query systems for distributed databases and data sources, such as OpenCQ [2] and NiagaraCQ [3], an active database model is assumed with which data changes are detected at the source as events and *all* events are pushed to the querying node to update the result of the continuous query. Considering the rate of the data changes in peer-to-peer databases, pushing all events to the querying node(s) fails to scale. To address this scalability problem, for distributed data stream applications Olston et al. [16] proposed adaptive filtering of the events at the sources of the data to reduce the cost of the data communication. However, considering the sheer number of nodes in peer-to-peer databases (which is usually three to six orders of magnitude greater than the number of data sources assumed in typical distributed data stream applications), even such push-based techniques that filter out the redundant data communication fail to scale well and, therefore, are not appropriate for continuous query answering in peer-to-peer databases. With Digest, we adopt sampling, a pull-based approach that scales by avoiding data collection from all nodes but a relatively small sampled subset of the nodes.

In contrast with Digest and other aggregate query answering approaches with which the query evaluation process occurs out of the network at the querying node, there are a number of techniques recently developed for *in-network* aggregate query processing. The randomized distributed algorithms [17], [18], [19], [20] are communication-intensive and their communication overhead is only justified when all nodes of the network issue the same aggregate query simultaneously. TAG [21] incurs less overhead but with its tree-based aggregation scheme, it is prone to severe miscalculations due to frequent fragmentation of the poorly connected topology of the tree, specially in the dynamic peer-to-peer databases. Also, DHT based aggregation techniques [22] are limited to the peer-to-peer databases with structured topologies.

The most relevant related work is the work by Arai et al. [23] on sample-based approximate aggregation queries in peer-to-peer networks, which is limited to snapshot queries, whereas we focus on continuous queries.

Acknowledgments

This research has been funded in part by NSF grants IIS-0238560 (PECASE), IIS-0324955 (ITR), IIS-0534761 and IIS-0742811 (SGER), and in part under JPL SURP program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] D. Terry, D. Goldberg, D. Nichols, and B. Oki, "Continuous queries over append-only databases," in *Proceedings of SIGMOD*, June 1992.

[2] L. Liu, C. Pu, and W. Tang, "Continual queries for internet scale event-driven information delivery," *IEEE Transcations Knowledge and Data Engineering (TKDE)*, vol. 11, no. 4, pp. 610–628, 1999.

[3] J. Chen, D. DeWitt, F. Tian, and Y. Wang, "NiagaraCQ: A scalable continuous query system for internet databases," in *Proceedings of SIGMOD*, May 2000.

[4] C. Chen and N. Roussopoulos, "Adaptive selectivity estimation using query feedback," in *Proceedings of SIGMOD*, May 1994.

[5] D. Donjerkovic, Y. Ioannidis, and R. Ramakrishnan, "Dynamic histograms: Capturing evolving data sets," in *Proceedings of ICDE*, February 2000.

[6] P. Gibbons and Y. Matias, "New sampling-based summary statistics for improving approximate query answers," in *Proceedings of SIGMOD*, June 1998.

[7] V. Ganti, M. Lee, and R. Ramakrishnan, "Icicles: Self-tuning samples for approximate query answering," in *Proceedings of SIGMOD*, May 2000.

[8] S. Babu and J. Widom, "Continuous queries over data streams," *SIGMOD Record*, September 2001.

[9] A. Dobra, M. Garofalakis, J. Gehrke, and R. Rastogi, "Processing complex aggregate queries over data streams," in *Proceedings of SIGMOD*, June 2002.

[10] W. Hou and G. Ozsoyoglu, "Statistical estimators for aggregate relational algebra queries," *ACM Transactions on Database Systems (TODS)*, vol. 16, no. 4, pp. 600–654, December 1991.

[11] R. Lipton, J. Naughton, and D. Schneider, "Practical selectivity estimation through adaptive sampling," in *Proceedings of SIGMOD*, May 1990.

[12] P. Haas and A. Swami, "Sequential sampling procedures for query size estimation," in *Proceedings of SIGMOD*, June 1992.

[13] S. Acharya, P. Gibbons, and V. Poosala, "Congressional samples for approximate answering of group-by queries," in *Proceedings of SIGMOD*, May 2000.

[14] J. Hellerstein, P. Haas, and H. Wang, "Online aggregation," in *Proceedings of SIGMOD*, May 1997.

[15] A. Gupta and I. S. Mumick, "Maintenance of materialized views: Problems, techniques, and applications," *IEEE Data Engineering Bulletin*, vol. 18, no. 2, pp. 3–18, 1995.

[16] C. Olston, J. Jiang, and J. Widom, "Adaptive filters for continuous queries over distributed data streams," in *Proceedings of SIGMOD*, June 2003.

[17] M. Bawa, H. Garcia-Molina, A. Gionis, and R. Motwani, "Estimating aggregates on a peer-to-peer network," submitted for publication.

[18] E. Cohen and H. Kaplan, "Spatially-decaying aggregation over a network: model and algorithms," in *Proceedings of SIGMOD*, June 2004.

[19] J. Considine, F. Li, G. Kollios, and J. Byers, "Approximate aggregation techniques for sensor databases," in *Proc. of ICDE*, March 2004.

[20] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proceedings of FOCS*, October 2003.

[21] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "TAG: a Tiny AGgregation service for ad-hoc sensor networks," in *Proceedings of OSDI*, December 2002.

[22] L. Galanis and D. DeWitt, "Scalable distributed aggregate computations through collaboration in peer-to-peer systems," submitted for publication.

[23] B. Arai, G. Das, D. Gunopulos, and V. Kalogeraki, "Approximating aggregation queries in peer-to-peer networks," in *Proceedings of ICDE*, April 2006.