Efficient Execution of Continuous Incoherency Bounded Queries over Multi-Source Streaming Data

Manish Bhide
IBM Research, India
abmanish@in.ibm.com

Krithi Ramamritham
IIT Bombay, India
krithi@iitb.ac.in

Mukund Agrawal
Symantec Corporation, India
mukund_agrawal@symantec.com

Abstract

On-line decision making often involves query processing over time-varying data which arrives in the form of data streams from distributed locations. In such environments typically, a user application is interested in the value of some function defined over the data items. For example, the traffic management system can make control decisions based on the observed traffic at major intersections; stock investors can manage their investments based on the value of their portfolios. In this paper we present a system that supports pull based data refresh and query processing techniques where such queries access data from multiple distributed sources. Key challenges in supporting such Continuous Multi-Data Incoherency Bounded Queries lie in minimizing network and source overheads, without loss of fidelity in the query responses provided to users. We address these challenges by using mathematically sound approaches based on Gradient Descent and Constraint Optimization which allow us to adapt the refresh frequencies of the dynamically changing data and adjust the quality of service provided to different users.

1. Introduction

An increasing fraction of data on the web is time-varying and is presented in the form of data streams. Examples of time-varying data include financial information such as stock prices and currency exchange rates, real-time traffic and data from process control applications. Such data is frequently used for online decision making, and in many scenarios the decision making involves *multiple* time-varying data items, from multiple distributed sources (data streams). Examples include a user tracking a portfolio of stocks and making decisions based on the net value of the portfolio. Users are generally not interested in the exact value of the portfolio and are content with knowing the value accurately within some fixed accuracy bound. Observe that computing the overall value of the portfolio at any instant requires up-to-date values of stocks in the portfolio. If the user is interested in knowing the estimated value of the stock portfolio (accurate say, within some incoherency bound/query accuracy bound), then the aggregator/proxy needs to refresh the value of each stock price intelligently so as to ensure that these user-specified query accuracy bounds are not violated. Such queries are a special form of continuous queries and are referred to as *Continuous Multi-Data Incoherency Bounded Queries (COMIQ)* [6]. In this paper we consider COMIQs which are a weighted aggregation of the value of a number of data items.

Much of the prior work on continuous queries (see Sec. 6) has assumed that queries are handled directly by the server (data source) so that the source pushes changes to the data aggregator (DA). Such an approach requires special support at the source and within the dissemination infrastructure [10], as push based techniques cannot be used with the standard pull based HTTP protocol. In this paper we focus on avoiding the need for any changes in the source or in the existing internet infrastructure. Our DA based approach raises new research challenges. The key challenge is to ensure that the results of the queries are no different from the case where the queries are handled by an idealized DA which has the current version of the data available all the time. In this paper, we develop (i) a suite of data refresh algorithms that try to minimize the messages between the sources and DAs, and show (ii) how these algorithms can be integrated for efficient processing of COMIQs. In the rest of this section, we provide a precise definition of COMIQs and then outline the research contributions of this paper.

1.1 COMIQs: An Introduction

A COMIQ $Q(\mathcal{M}, \mathcal{N}, \mathcal{T})$ operates on N data items m_1, …, $m_N \in \mathcal{M}$ with n_1, n_2, …, $n_N \in \mathcal{N}$ as the weights attached to these data items and ensures that the value of the COMIQ at the source is no different than the value of the COMIQ at the DA by more than $\mathcal{T}$ (*Incoherency bound/Query accuracy bound*). In this paper, we focus on Sum Based Queries where the user should have the correct knowledge of the total value of a COMIQ within a margin of $\mathcal{T}$. This can be stated as

$$\forall t \ \ |\sum_{i=1}^{N}(s_i(t)\cdot n_i) - \sum_{i=1}^{N}(p_i(t)\cdot n_i)| \leq \mathcal{T} \qquad (1)$$

In the above formula, $s_i(t)$ and $p_i(t)$ denote the value of the data item m_i at time t at the source and DA respectively.

1.2 Comparison Metrics

Fidelity Loss: This quantifies the accuracy of a COMIQ executed by a DA with respect to that of an idealized one i.e., a DA which has current data available all the time. Fidelity Loss is defined as the percentage of total time for which Eq. (1) is violated.

Network Overhead: We define the network overhead as the number of messages exchanged between the source and the DA for every 100 data values at the source.

1.3 Contributions of this paper

In this paper, we present efficient techniques to evaluate COMIQs at a DA so as to minimize the network overhead and fidelity loss. A user is interested only in the accuracy of a COMIQ's results and not the exact values of the data items. Hence as long as a DA is correctly informed about the value of the COMIQ within the specified query accuracy bound, even if an individual data item is changing very rapidly there may be no need to track these changes frequently. This key observation is exploited by our algorithm which allows us to reduce the network overhead.

The *Constraint Optimization Based COMIQs Execution Approach ($CoCEA$)*, outlined in Sec. 2, models the problem as a Constraint Optimization problem. This approach results in some fidelity loss but has low network overhead. The problem of fidelity loss is avoided by our *Gradient Descent Based COMIQs Execution Approach (GdCEA)*, outlined in Sec. 3. The $GdCEA$ makes use of the Gradient Descent algorithm which results in lower fidelity loss at the expense of higher network overhead. We also present a high performance *Hybrid COMIQs Execution Approach ($HyCEA$)* in Sec. 4, that intelligently uses both the Gradient Descent and the Constraint Optimization based approaches. We demonstrate that refreshing frequently when there is a violation in data accuracy requirement leads to an increase in the network overhead, but does not always lead to a reduction in fidelity loss. This surprising result is used by the $HyCEA$ algorithm, so as to improve its performance vis-a-vis $CoCEA$ and $GdCEA$.

The algorithms presented in this paper are superior alternatives to the techniques proposed in [6]. The approach presented in [6] is heuristics based and has drawbacks which are avoided by our solution (further details in Section 4.2). We provide details of our approaches and experimentally evaluate their performance using COMIQs defined over real-world streams of dynamically changing data (specifically, stock prices) in Sec. 2 through 5. These streams were constructed through repeated polling of http://finance.yahoo.com. All algorithms were evaluated using a prototype source/DA that employed trace replay. Sec. 5 presents a technique to extend $HyCEA$ to handle multiple COMIQs. Related work is summarized in Sec. 6 and Sec. 7 concludes the paper.

2 The Constraint Optimization Based COMIQs Execution Approach ($CoCEA$)

$CoCEA$ attempts to minimize the network overhead without loss in fidelity by constructing and solving an inequality constrained minimization problem. The algorithm dynamically computes the data accuracy requirement for each data item, taking into account the changes in the value of the data item vis-a-vis the changes in other data items constituting the COMIQ. The data accuracy requirement (c) of a data item denotes the maximum permissible deviation in the data item value cached at the DA from its value at the source. The data accuracy requirement needs to satisfy the following equation:

$$c_1 \times n_1 + c_2 \times n_2 + \ldots + c_N \times n_N = \mathcal{T} \qquad (2)$$

The parameter c_i is the data accuracy requirement of the i^{th} data item. Intuitively, if each data item (m_i) changes by an amount c_i, the value of the COMIQ changes by $\mathcal{T}$. The challenge then is to determine an appropriate c_i for each data item such that Eq. (2) is satisfied. Once the values of c have been calculated, the next challenge is to determine the refresh interval or Time to refresh (TTR) for each of the data items such that the network overhead is minimized without loss in fidelity. We first present a solution to the TTR problem and then address the question of finding the data accuracy requirement for each data item.

2.1 Determining TTRs

If we are able to estimate the expected *rate of change* (d_i) in the value of each data item in the future, then the TTR can be computed as follows:

$$TTR_i = \frac{c_i}{|d_i|} \qquad (3)$$

The challenge now is to determine the value of d_i for each data item. To do so, we have extended the core idea of Asset Pricing Model for Stocks [12]. It tries to model the behavior of a dynamically changing data item by decomposing its changes into drift component, which is the expected change in its value (based on the history of its behavior) and a diffusion component (changes caused by processes outside the system). Our technique is based on the Black Scholes Differential Equation [12] and is given by:

$$d \ = \ \mu \times dT \ + \ \sigma \times dX \quad where \qquad (4)$$

d is the estimated rate of change in the value of the data item during time dT, μ is the mean of change in the value of the data item per unit time, σ is the volatility in the value of the data item and dX is the measure of external factors. We fix dT as 1 so as to calculate the estimated change in data value over the next time unit. Every time a data item is refreshed, we compute its rate of change and use an exponentially smoothed value to find μ. The value of σ is calculated using the standard formula for volatility [11]. The next task is to estimate the value of dX:

$$dX = \frac{L \times (d' - \mu_{lt})}{\sigma} + (1 - L) \times dX_{lt} \tag{5}$$

In this equation, d' is the actual rate of change in the data item value during the last refresh interval. Thus, we find the value of $\frac{d'-\mu_{lt}}{\sigma}$ (μ_{lt} is the value of μ during last refresh), which (as per Eq. 4) is the actual value of dX operational during the last refresh interval. Hence this gives us a measure of the actual unexpected changes that occurred during the last refresh interval. We use an exponential smoothed value of this parameter (using smoothing constant L) to come up with an estimate of the external forces likely to be active during the next refresh interval. If we now look back at Eq. (4), the first term $\mu \times dT$ helps us to estimate the drift and the second term $\sigma \times dX$ helps us in estimating the diffusion.

2.2 Determining Data Accuracy Requirements

In this section, we first formulate the problem of determining the data accuracy requirements as an inequality constrained minimization problem and then show how standard techniques need to be modified to find the optimal solution. Assuming that the value of d_i is accurate, if the value of data accuracy requirement is computed such that it satisfies Eq. (2), then there won't be any fidelity loss. Thus we need to compute the value of data accuracy requirement for each data item such that Eq. (2) is not violated. Using Eq. (3), the number of messages per second can be calculated as:

$$Messages\ Per\ Second = \sum_{i=1}^{N} \left(\frac{|d_i|}{c_i} \right) \tag{6}$$

Our aim of minimizing the fidelity loss and network overhead can be achieved by finding c_i values such that (a) Eq. (6) is minimized and (b) Eq. (2) is not violated. This can be done using convex optimization techniques (notice that Eq. (6) can be trivially proved to be convex for $c_i > 0$). However, when the changes in data items in the COMIQ are in opposite directions, the use of absolute value of d_i can lead to a lot of unnecessary network overhead even if, overall there is no change in the value of the COMIQ. Due to this problem, standard minimization techniques cannot be used directly for minimizing Eq. 6. We tackle this problem in the later part of this section. Let us for the time being use Eq. (6) as is. Thus, in this problem we have to minimize Eq. (6) under the following constraints:

$$\sum_{i=1}^{N} (c_i \times n_i) - \mathcal{T} \leq 0 \tag{7}$$

$$\forall i : \ c_i > 0 \tag{8}$$

Notice that for achieving low fidelity loss we need not satisfy Eq. (2). Eq. (2) represents the boundary condition and hence the equality sign can be replaced by an inequality as is done in Eq. (7). Thus this turns out to be an inequality constrained minimization problem which can be solved by using Interior Point methods such as Barrier method [13]. The idea is to approximate the inequality constrained minimization problem using an equality constrained minimization problem to which methods such as Newton's method [3] can be applied. This is done by making the inequality constraints implicit in the objective function using a logarithmic Indicator function (Logarithmic Barrier). Using this our new objective function is:

$$\mathcal{O} = \sum_{i=1}^{N} \left(\frac{|d_i|}{c_i} \right) - \frac{1}{t} \times \left(log(\mathcal{T} - \sum_{i=1}^{N} (c_i \times n_i)) + \sum_{i=1}^{N} log(c_i) \right) \tag{9}$$

The parameter t in the above equation governs the slope of the logarithmic barrier (more on this later). Note that if any of the constraints given by Eq. (7) and (8) are violated, the value of the objective function becomes ∞. Thus the solution (which consists of c_i values) given by standard techniques like the Damped Newton method ensures that the constraints are not violated and the objective function is minimized. As the constraints are not violated, the fidelity loss is minimized.

Coming back to the parameter t, it can be seen that with an increase in the value of t, the indicator function approximates the ideal function in a better manner. However, if we increase the value of t arbitrarily, it is difficult to minimize the objective function. This problem can be circumvented as follows: We start with a small value of t and optimize Eq. (9) using the Damped Newton method. We then increase the value of t and optimize the modified objective function, starting with the optimal value of c_i calculated during the previous iteration. This continues till a stopping criterion such as number of iterations is met.

2.2.1 Modifying Barrier Method

As noted earlier, Eq. (6) might be incorrect if the data items in the COMIQ change in opposite directions. We tackle this inadequacy by increasing the value of the query accuracy bound $\mathcal{T}$, so that all the data items can be assumed to move in one direction.

During each refresh, we compute the sum of the weighted rate of change ($ROC = |d_i| \times n_i$) of those data items (minority data items) which are changing in the direction opposite to the direction of change in the COMIQ value. In order to invert the direction of change of the minority data items, the quantity $2 \times ROC \times TTR$ is added to the query accuracy bound $\mathcal{T}$. We have to add twice the ROC value to $\mathcal{T}$ in order to ensure that the total time required by the COMIQ to exceed the query accuracy bound with the modified $\mathcal{T}$ and absolute d_i values (i.e., $\frac{\mathcal{T}+2\times ROC\times TTR}{\sum(|d_i|\times n_i)}$) is no different than the time required with the original $\mathcal{T}$ and the original d_i values (i.e., $\frac{\mathcal{T}}{\sum(d_i\times n_i)}$). With the modified query accuracy bound, all data items can be assumed to change in the positive direction, thereby ensuring correctness of Eq. (6). The TTR that is used above is the one that we are about to calculate. Thus in order to find the exact value of $\mathcal{T}$ to be used in the objective function, we need to know the future TTR's. But we need the objective function to compute the future TTR's. Hence this is a chicken and egg problem. The solution lies in the fact that our technique for minimizing Eq. (9) involves solution to a sequence of problems (one for each value of t). The algorithm (modified Barrier method) that we use is given below:

- **Given**: Objective function $\mathcal{O}$ (Eq. 9), $\nu > 1$, ROC, c' (initial guess for c), t
- **Repeat**
  - *Centering Step*: Use Damped Newton method to find the optimal value of $\mathcal{O}$ starting at c'.
  - *Update*: $c' = c^*$ (optimal value computed in centering step)
  - *Stopping criterion*: Quit if solution is sufficiently accurate.
  - *Increase* $\mathcal{T}$: Using c' in Eq. (3), find minimum TTR_{min}. Set $\mathcal{T} = \mathcal{T}' + 2 \times TTR_{min} \times ROC$ in the objective function ($\mathcal{T}'$ = original $\mathcal{T}$).
  - *Increase* t: Set $t = \nu \times t$

In this algorithm during each iteration, we slowly increase the value of t using the parameter ν thereby increasing the accuracy of the logarithmic indicator function. During our experimental evaluations, we noticed that this algorithm converges very rapidly and it takes a few milliseconds to find the optimal value of the data accuracy requirement. In summary, the $CoCEA$ algorithm involves the following steps: (i) It uses the formula given in Eq. (3) to compute the TTR for each data item. (ii) The expected rate of change (d_i) in the data item value is computed using the modified Black Scholes differential equation (Eq. 4) and (iii) The data accuracy requirement value is computed using the modified Barrier method. This formulation ensures that the c values do not violate Eq. (7) and Eq. (8). At the same time it also ensures that Eq. (6) is minimized. Hence the network overhead is optimized with minimal fidelity loss.

2.2.2 Relation of c and d

The CoCEA algorithm does not assign the largest c value to the fastest changing data item. This contradicts the result presented in [9] which suggests that the largest data accuracy requirement should be given to the fastest changing data item. This contradiction is due to the fact that the technique presented in [9] is push based whereas we use a pull based technique. A pull based technique has to be inherently pessimistic as there is no way to predict the change in a dynamic data value with 100% accuracy. Hence, an important conclusion of our work is that in a pull based technique, in order to achieved the desired fidelity loss, a fast changing data item should not have the largest data accuracy requirement.

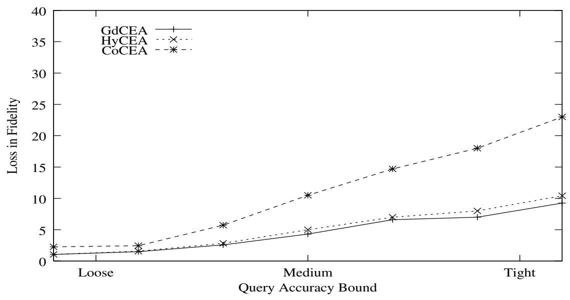
2.3 Performance of CoCEA

As mentioned earlier, we used stock traces as our data streams. Trace for each stock consisted of 10,000 data values. The COMIQs were formed by selecting data items uniformly at random from the 150 stock traces and assigning random weights to each of the data items. In our experimentation, the TTR value was not allowed to go beyond $TTR_{max} = 60$ so as to limit the staleness of the data at the DA. The value of ν was set to 2 and the initial value of t was taken to be 1.

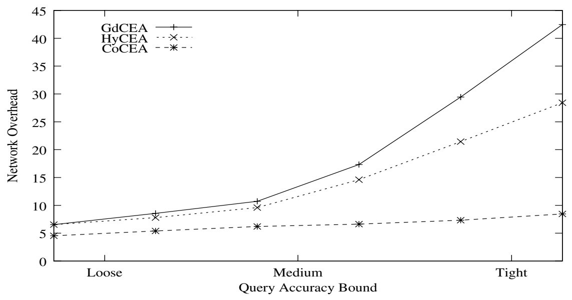
Figure 1 compares the network overhead and the loss in fidelity of $CoCEA$ with two other algorithms which are introduced in Sec. 3 and 4. We discuss their relative performance in the respective sections. The figure shows that the network overhead offered by $COCEA$ is very low, which shows that the algorithm is successful in minimizing the objective function. The graphs show that as the value of the query accuracy bound decreases (i.e., the query accuracy bound becomes tighter) there is a gradual increase in the fidelity loss and network overhead. The CoCEA increases the refresh frequency with the tightening of the query accuracy bound which manages to salvage the loss in fidelity to some extent, but it might not still be acceptable for applications with high fidelity requirements. The reason for this loss in fidelity is not hard to fathom. Eq. (7) ensures that there is no loss in fidelity provided the value of d_i is correct. Due to the unpredictable nature of stock quotes, as the query accuracy bound becomes very tight, the tolerable margin of error in the d_i values becomes very small. Hence, the value of c_i might be too large to detect any unpredictable changes in the value of the data item. This suggests that we need to be a bit more conservative in calculating the value of the data accuracy requirement so that we are able to handle unexpected changes in the value of the data items. Such a technique, which uses the Gradient Descent algorithm, is proposed in the next section.

3 Gradient Descent Based COMIQs Execution Approach ($GdCEA$)

Gradient Descent is a convex optimization technique in which the design variable is changed along the direction of the steepest descent in the objective function. In this section we explore the use of Gradient Descent to minimize the network overhead as well as fidelity loss. As our goals are the same as in the previous algorithm, the basic optimization problem is unchanged and is given by Eq. (6). In order to ensure that the fidelity requirements are met, the constraints given by Eq. (7) and (8) need to be incorporated in the optimization function. The constraints can be handled in the Gradient Descent technique by adding a "*performance penalty*" to the optimization function. The performance penalties related to Eq. (7), E_1^C, and Eq. (8), E_2^C, for a COMIQ C are given as:



Variation of fidelity loss with query accuracy bound



Variation of network overhead with query accuracy bound

Figure 1. Performance of $CoCEA$, $GdCEA$ and $HyCEA$

$$E_1^C = \begin{cases} \frac{1}{2}(\sum_{i=1}^{N}(c_i n_i) - \mathcal{T})^2 & \text{if } \sum_{i=1}^{N}(c_i n_i) > \mathcal{T} \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

$$E_2^C = \begin{cases} \frac{1}{2}c_i^2 & \text{if } c_i < 0 \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

In this equation, the c_i value calculated during the previous run of the Gradient Descent algorithm is used. In order to ensure the correctness of Eq. (6) (due to the use of the absolute value of d_i), we use the same strategy as used in the $CoCEA$ algorithm. The Gradient Descent algorithm minimizes an objective function which is the weighted sum of the objective function given by Eq. (6) and the performance penalty.

$$\mathcal{O} = (WT_P \times Messages\ Per\ Second) + (WT_E \times E) \quad (12)$$

where E is the total performance penalty obtained by adding Eq. (10) and (11). WT_P is the weight given to the number of messages and WT_E is the weight given to the error term.

We have to find the c_i values such that the above objective function is minimized. The Gradient Descent technique changes the c vector in the direction that produces the steepest descent along the surface of the objective function. The direction of the steepest descent is given by the negative of the gradient (derivative) of the objective function $\mathcal{O}$ with respect to each of the c's. Hence after each refresh, c_i is recomputed as:

$$\vec{c} = \vec{c} - \eta \times \bigtriangledown\mathcal{O}(\vec{c}) \quad (13)$$

The parameter η ($\eta < 1$) is called as the learning rate. Without loss of generality, it can be assumed that this parameter is incorporated in the weights WT_P and WT_E of the objective function. After calculating the partial derivatives of the objective function, the final formula for c is given by:

$$\begin{aligned} c_i &= c_i - WT_P \times PD_1 - WT_E \times PD_2 \ \ where \\ PD_1 &= -1 \times \frac{|d_i|}{c_i^2} \ and \\ PD_2 &= n_i \times (\sum_{j=1}^{N}(c_j n_j) - \mathcal{T}) + c_i \end{aligned} \quad (14)$$

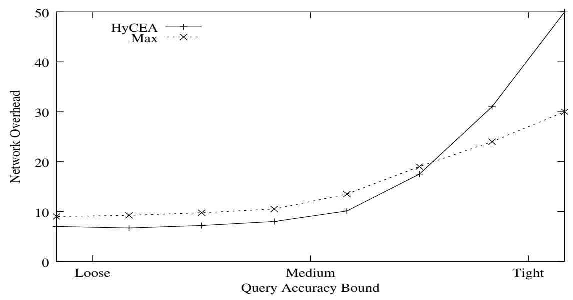
When the constraints given by Eq. (10) and (11) are satisfied, the factor PD_2 is set to zero and the c is changed only due to the factor PD_1. Thus, in such a case the c is changed based on the relative weight and the rate of change of the data item. The data accuracy requirement generally goes on changing till the time that the constraint gets violated. At that point, the performance penalty causes a decrease in the value of c until we are again in the space where the constraints are satisfied.

Thus the Gradient Descent algorithm can be summarized as: (i) At each refresh, find the d_i value using the modified Black Scholes differential equation. (ii) Find the ROC value of the COMIQ during the previous refresh. We use the previous TTR as an estimate of the TTR in the future. Hence using the previous TTR, change the value of $\mathcal{T}$ by adding an amount $TTR_{previous} \times ROC \times 2$ and (iii) Use the previous c values, the modified $\mathcal{T}$ value and the d values to find the new c value using Eq. (13). Use these new c values to find the TTR of the data item using Eq. (3). The Gradient Descent algorithm can be used to find the final optimal value of c by running multiple iterations of Eq. 13. However, our aim is to detect the unexpected changes and hence we use only one iteration after every refresh.

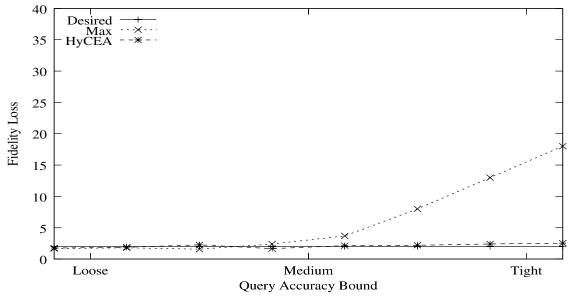
3.1 Performance of GdCEA

Figure 1 compares the network overhead and the loss in fidelity of $GdCEA$ with that of $CoCEA$. In these experiments, the value of WT_P was set to 0.01 and that of WT_E was set to 0.1. From the figure it can be observed that, unlike the $CoCEA$ algorithm, there is much less increase in fidelity loss of $GdCEA$ with the tightening of the query accuracy bound. However, the network overhead of $GdCEA$ is higher than that of $CoCEA$. The $GdCEA$ is a conservative algorithm and due to the small increments in the value of c it can handle sudden unexpected changes in the value of the data item much more efficiently as compared to $CoCEA$. Another important reason for the increase in the network overhead of the $GdCEA$ algorithm is the effect of performance penalty. When any of the c values violates the constraints, the penalty factor comes into play and reduces the value of c. It was observed that the reduction in the value of c was very large, due to which c was set to a very small value.

Thus the bottom line is that the $GdCEA$ and $CoCEA$



Network Overhead



Fidelity Loss

Figure 2. Comparison of $HyCEA$ with Max algorithm

algorithm have complementary properties in terms of loss of fidelity and network overhead. This immediately begs the following question: *Can we combine these two techniques such that we get the low loss in fidelity offered by the $GdCEA$ algorithm while saving on network overhead?* We explore the answer to this question in the next section.

4 A Hybrid COMIQs Execution Approach ($HyCEA$)

In this section, we explore a technique to combine the $GdCEA$ and $CoCEA$ algorithm in an intelligent manner. As noted in the previous section whenever there is a violation, the performance penalty of the GdCEA algorithm reduces the data accuracy requirement by a value far greater than the ideal value. This prompted us to make use of the optimal value of the data accuracy requirement calculated using the $CoCEA$ algorithm. This is done in the $HyCEA$ algorithm which predominantly uses the $GdCEA$ algorithm and kicks in the $CoCEA$ algorithm only when any of the constraints is violated. Thus this algorithm relaxes the data accuracy requirement when the performance penalty comes into play. This might seem a bit counter-intuitive as ideally one would expect to refresh very frequently when the data accuracy requirement is violated. The reason for this is that when the current data item is refreshed, the DA gets an accurate value of the data item from the source. Hence the DA can now relax till the time that the COMIQ is expected to change by a value more than $\mathcal{T}$ as compared to the current value at the DA. Hence if we use the $CoCEA$ algorithm in such a scenario, the chances of error are minimal and we save on network overhead.

4.1 Evaluation of $HyCEA$

Figure 1 shows that the network overhead due to the $HyCEA$ algorithm is in between that of $GdCEA$ and $CoCEA$. This combined with the fact that the algorithm provides almost the same fidelity as that of $GdCEA$ proves that $HyCEA$ succeeds in saving the extra network overhead due to the performance penalty.

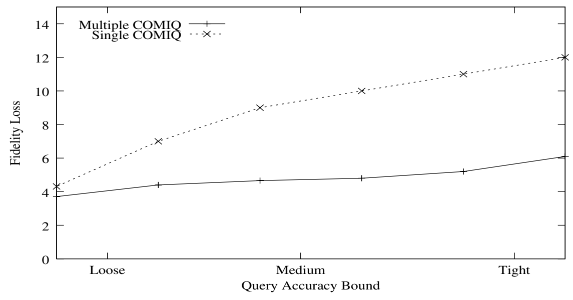
4.2 Towards Achieving Specified Fidelity

The results presented so far suggest that the algorithms do have some amount of fidelity loss. However in practice, users can only tolerate a specific amount of fidelity loss. In this section we show how HyCEA can be altered so as to limit the fidelity loss offered to the user depending on the user's requirements. We use a feedback based technique [6], which adaptively changes the query accuracy bound based on the fidelity delivered till that point of time. In this technique, the fidelity loss is periodically monitored at the DA. If the fidelity loss is less than that desired by the user, the query accuracy bound is reduced so that TTR's are reduced and the algorithm gets a chance to improve its performance. On the other hand, if the fidelity loss is less than that desired by the user then the query accuracy bound is increased.

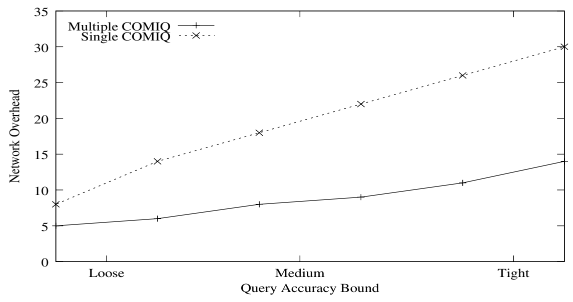
[6] proposes heuristic based techniques (such as Max algorithm) to handle continuous queries at data aggregators. Figure 2 shows the performance of $HyCEA$ algorithm with feedback vis-a-vis the heuristic based Max algorithm presented in [6]. In these experiments, user desired fidelity loss was set to 2%. It is evident from the figure that unlike the Max algorithm, the $HyCEA$ algorithm is capable of delivering the desired fidelity to users. It pays the price for this in terms of increased network messages for tight query accuracy bounds. The reason for the increase in fidelity loss in the Max algorithm is due to the use of heuristics wherein it does not refresh all the data items constituting the COMIQ. For tight query accuracy bounds, this might not be a wise thing to do as any sudden jump in the data item value that is not refreshed can violate the query accuracy bound and can go unnoticed. Such sudden spurts in data values can turn out to be the most interesting activities for an end user.

5 Extension of $HyCEA$ to Multiple COMIQs

In this section we present a technique to extend the $HyCEA$ algorithm to cater to multiple COMIQs, where a large number of COMIQs are being executed at the DA.



Variation in Fidelity Loss



Variation of Network Overhead

Figure 3. Performance of $HyCEA$ for single and multiple COMIQs

5.1 Enhanced Objective Function

In this setup, our basic objective is to minimize the network overhead, which is same as that in the previous algorithms. Hence the basic objective function is similar as that given by Eq. (6).

$$\mathcal{O} = \sum_{i=1}^{N} \left(\frac{|d_i|}{c_i} \right) \tag{15}$$

The only difference is that in this case, N is the total number of data items being handled by the DA. Similarly, the performance penalty term for the constraint given by Eq. (7), for a single COMIQ j is given by:

$$E_1^j = \begin{cases} \frac{1}{2}(\sum_{i \in \mathcal{M}(j)}(c_i \times n_i) - \mathcal{T}^j)^2 & \text{if } \sum_{i \in \mathcal{M}(j)}(c_i \times n_i) > \mathcal{T}^j \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

In the above equation, $\mathcal{M}(j)$ gives the constituent data items of the COMIQ j and T^j is the modified query accuracy bound of the COMIQ j. As there can be multiple COMIQs executing at the DA the total error term due to all the COMIQs is given by:

$$E_1 = \sum_{j=1}^{Q} E_1^j \tag{17}$$

where Q is the total number of COMIQs executing simultaneously at the DA. There won't be any change in the performance penalty term given by Eq. (11). Hence the total error due to this term is:

$$E_2 = \sum_{i=1}^{N} \sum_{j \in X_i} E_2^i \tag{18}$$

In this equation, X_i is the set of all those COMIQs in which the i^{th} data item participates. The values given by Eq. (17) and (18) can be substituted in Eq. (12) to get the objective function to be minimized for the multiple COMIQ scenario. The partial derivative of this new objective function with respect to the individual data accuracy requirement values is given by:

$$\begin{aligned} c_i &= c_i - WT_P \times PD_1 - WT_E \times PD_2 \;\; where \\ PD_1 &= -1 \times \frac{|d_i|}{c_i^2} \;\; and \\ PD_2 &= \sum_{k \in V} n_i \times (\sum_{i,j \in \mathcal{M}(k)}(c_j n_j) - \mathcal{T}^k) + |R| \times c_i \end{aligned} \tag{19}$$

V in the above equation is the set of all those COMIQs in which data item i participates and for which the constraint given by Eq. (7) is violated and the set R is the set of all those COMIQs for which the value of c_i is less than zero. Using the above formula, we can now calculate the data accuracy requirement for each data item handled by the DA. As in the $HyCEA$ algorithm, if any of the constraints are violated, we calculate the optimal value of the data accuracy requirement using the $CoCEA$ algorithm, optimized with respect to any one randomly selected COMIQ for which the constraint got violated.

5.2 Performance for Multiple COMIQs

We compared the fidelity loss and network overhead characteristic of $HyCEA$ applied to multiple COMIQs with that of the $HyCEA$ algorithm applied to a single COMIQ. In this setup, if the DA was handling 20 COMIQs, then in order to compare the results of this setup, we ran the 20 COMIQs individually, and averaged the results across the 20 runs to come up with the results for $HyCEA$ applied to a single COMIQ. The results presented in Figure 3 show a surprising result. The network overhead and fidelity loss of the algorithm is significantly less as compared to the single COMIQ case. The reason for this is that the algorithm by virtue of considering all the queries together, is able to use the knowledge of data item distribution across queries. For example, consider a data item that occurs in a very large number of queries. HyCEA will give more importance to this data item and hence will set a tighter data accuracy requirement for this data item. Although, this results in higher network overhead for this single data item, the fact of this data item being more coherent can be used to relax the data accuracy of several other data items that co-occur with this data item. Thus, there is an overall decrease in network overhead and fidelity loss. In a nutshell, the $HyCEA$ algorithm helps us to reduce the network overhead by almost 17% and it offers better fidelity loss characteristics to the tune of 5% as compared to the single COMIQ case. This also shows the scalable nature of our techniques.

6 Related Work

Caching of *dynamic* content has been studied in [7] wherein a scheme based on push-based invalidation and dependence graphs is proposed. This does not explicitly address data accuracy maintenance for efficiently executing queries at a DA. The concept of queries over data streams was presented in [1]. However, this work assumed that the streams were available in their entirety and effectively ran the queries at the source. This is a major drawback and might not be practical for real life scenarios. The concept of filters is used in [9] to deal with continuous aggregate queries. They make use of filters at the source and assume that the source can push data values to the proxy. A similar approach is used in [4] which tries to address the problem of tracking approximate summaries of the complete data distribution over distributed data streams. The approach also requires the source to push data values to the proxy. The concept of thresholded counts is introduced in [5]. The solution consists of multiple remote sites and a coordinator site. The technique requires the remote sites to keep track of a local threshold and push the data value to the coordinator site when the threshold is exceeded. Due to the push element, this approach cannot be used with the existing internet infrastructure. Our approach assumes no special support at the source and hence can be used with the normal HTTP protocol. A pull based approach for achieving stochastic of single data items is proposed in [2, 14]. However, our work is for queries over multiple data items and hence orthogonal to them.

The problem of stream querying in finance was addressed by [8]. They focus on storing the historical stream data in an intelligent manner using which they answer queries. Our work handles queries that do not need historical information and hence our approach is orthogonal to their work. In summary, none of these research efforts have focused on the pull based infrastructure/algorithms necessary for efficient execution of COMIQs at a DA, which is the focus of this paper.

7 Conclusions

On-line decision making often involves processing significant amounts of time-varying data. In such environments, decisions are typically made using an estimated value (with bounded inaccuracy) of a continuous query over a set of data items. In this paper, we investigated adaptive data refresh techniques where such *Continuous Multi-Data Incoherency Bounded Queries* access data from multiple distributed sources. Key challenges in supporting such queries lie in meeting users' consistency requirements while minimizing network overheads, without the loss of fidelity in the query responses provided to users. Meeting these required us to solve two subproblems: (1) deriving the data accuracy requirement of each of the data items used by the DA, and (2) ensuring that the (derived) data accuracy requirement associated with each data item is satisfied. To address the former problem, we formulated it as a convex optimization problem and solved it using Inequality Constrained Minimization, Gradient Descent and a combination of the two techniques. We addressed the latter problem by modifying the Black Scholes differential equation. Our work shows that contrary to the results of [9], for pull based techniques having a large data accuracy requirement for fast changing data items does not lead to good performance results. Finally we showed the scalability of our algorithms by extending the hybrid approach to cater to multiple COMIQs.

References

[1] S. Babu and J. Widom. Continuous queries over data streams. In *ACM Sigmod Record*, 2001.

[2] M. Bhide, P. Deolasee, A. Katkar, A. Panchbudhe, K. Ramamritham, and P. Shenoy. Adaptive push-pull: Disseminating dynamic web data. In *IEEE ToC*, 2002.

[3] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, USA, 2004.

[4] G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi. Holistic aggregates in a networked world: Distributed tracking of approximate quantiles. In *ACM Sigmod*, 2005.

[5] G. Cormode, K. Keralapura, and J. Ramamirtham. Communication-efficient distributed monitoring of thresholded counts. In *ACM Sigmod*, 2006.

[6] R. Gupta, A. Puri, and K. Ramamritham. Executing incoherency bounded queries at web data aggregators. In *WWW Conference*, 2005.

[7] A. Iyengar and J. Challenger. Improving web server performance by caching dynamic data. In *USENIX Symposium on Internet Technologies and Systems*, 1997.

[8] A. Lerner and D. Shasha. The virtues and challenges of ad hoc + streams querying in finance. In *IEEE Data Engineering Bulletin*, 2003.

[9] C. Olston, J. Jiang, and J. Widom. Adaptive filters for continuous queries over distributed data streams. In *ACM Sigmod*, 2003.

[10] S. Shah, K. Ramamritham, and C. Ravishankar. Client assignment in cdn. In *VLDB*, 2005.

[11] S. Taylor. *Asset Price Dynamics, Volatility, and Prediction*. Princeton University Press, USA, 2005.

[12] P. Wilmott, S. Howinson, and J. Dewyne. *The Mathematics of Financial Derivatives*. Cambridge University Press, New York, USA, 1995.

[13] M. Wright. *Interior methods for constrained optimization*. Cambridge University Press, New York, USA, 1992.

[14] S. Zhu and C. Ravishankar. Stochastic consistency and scalable pull-based caching for erratic data sources. In *VLDB*, 2004.