Bandwidth Efficient Distributed Monitoring Schemes

Yuval Alfassi
Computer Science Department
University of Haifa
Haifa, Israel
yuvalalfassi@gmail.com

Gal Yehuda
Computer Science Department
University of Haifa
Haifa, Israel
dkeren@cs.haifa.ac.il

Moshe Gabel
Computer Science Faculty
Technion
Haifa, Israel
mgabel@cs.technion.ac.ils

Assaf Shuster
Computer Science Faculty
Technion
Haifa, Israel
assaf@cs.technion.ac.il

Gal Yehuda
Computer Science Faculty
Technion
Haifa, Israel
gal2016@gmail.com

Abstract— **Distributed monitoring is a problem that arises when trying to monitor properties of dynamic data which are spread distributively. Tracking the value of a function over dynamic data in a distributed setting is a challenging problem in numerous real-world modern applications. Several monitoring schemes were proposed as an approach to coping with this problem in order to reduce the amount of communication messages between servers, as well as the communication bandwidth.**
Here, we propose several new distributed monitoring schemes using much less communication bandwidth. Existing schemes send high dimensional vectors from server to server while we propose some innovative methods for reducing the communication bandwidth.
One scheme we propose is the *Value Scheme* which exploits some traits of convex functions, and another is the *Distance Scheme* which treats the function monitoring problem as a geometric monitoring problem, thereby utilizing geometric distances for distributed monitoring.
Moreover, incorporating data-sketches into the schemes can be used to influence the bandwidth as well, allowing for some monitoring trade-offs to be made, depending on the sketch size.

I. Introduction

Monitoring a function over large amount of dynamically changed data in a distributed fashion is a common computer-science challenge. Whether its monitoring features of large distributed dynamic graphs [*] or multiple air pollution sensors [**], innovative approaches had to be developed in order to deal with the difficulties of both the data being dynamic and distributed.
The need of minimizing both the bandwidth and the processing power is expressed in [*]; good examples are internet of things objects which are operated on batteries, hence sending data via a communication channel is very costly and thus should be minimized. Furthermore, in the *Big Data* era, when data is of very high dimensionality and changes rapidly, sending the whole data is not only impractical, but also extremely time consuming; for instance, air pollution sensors which distributively have to determine the air pollution level may benefit from an economical communication approach [***].
Previous works were made on linear functions [*], where the linear properties make the distributed monitoring much easier; though in order to monitor non-linear functions a handful of difficulties arise. Works were done on distributively monitoring the entropy of distributed streams [*], monitoring the inner-product value of distributed dynamic vectors [**], and monitoring the number of triangles of dynamic graphs held in distributed servers [***].
It should be noted that the classic approach to monitor distributed dynamic data is to update at a certain frequency a central server, called *the coordinator*. This method may miss peaks of critical global data-changes, and also may be more expensive communication-wise – not only data will be sent even when little changes are made, but the whole big dimensional data will be sent which might be very expensive.

The distributed monitoring problem is defined as follows:

1) There are n servers and a coordinator.
   $server_i$ maintains its dynamic *local vector* v_i.
   The *global vector* is the average of the local vectors:
   $$v = \frac{1}{n}\sum_{i=1}^{n} v_i$$
2) A function f is monitored over the *global vector* v so it is ε-approximated with 100% accuracy.
   Let the estimation be the value μ and $f(v) \geq 0$, than:
   $$(1-\varepsilon)f(v) \leq \mu \leq (1+\varepsilon)f(v)$$

The monitoring is done by bounding the function f with a 'convex bound' function [*] – a function c which bounds the function from above (below) and used for the upper (lower) threshold monitoring. Given a fucntion f to monitor and a starting v_0 vector, it was proposed to approach the problem by find a convex function c [*] which bounds the f function e.g. for all v, $f(v) \leq c(v)$ and $f(v_0) = c(v_0)$. The approximation problem is reduced to $thresholdmonitoringproblem$ [*] where one monitors whether the function's value crosses a certain threshold. specifically, let T be the threshold's value, than the general upper-bound monitoring objective is to determine whether:

$$f(v) \leq T \tag{1}$$

and since $f(v) \leq c(v)$ for all v, we'd monitor whether:

$$c(v) \leq T \tag{2}$$

In turn, this threshold monitoring is treated as a *geometric monitoring problem*, where there's a *valid zone* of vectors $\{v \mid c(v) \leq T\}$, which since c is a convex function, every linear combination of vectors inside the *safe zone* is also inside the *safe zone*.

A. Contributions

1) Introducing multiple innovative distributed monitoring schemes which avoid sending big dimensional data unless crucially critical.
2) Proving the *Distance Lemma*, a lemma used as a basis of a distributed monitoring scheme we present. The *Distance Lemma* states that given a convex body and several points, if the sum of distances to the convex border from the points inside the convex body is greater than the sum of distances to the border from the points on the outside, than the average of the points is inside the convex body.
3) Conducting several experiments, laying out comparisons of multiple attributes of distributed monitoring schemes on real-world data, focusing on the bandwidth consumption.

II. Previous Work

**** Consult with Dani on this part ****

(1) Linear Functions
(2) Covering Spheres
(3) Convex Decomposition

III. Vector Scheme

The *Vector Scheme*'s idea is to balance the data vectors of the servers whenever a local vector gets out of the covex function's bound. The *Vector Scheme*would try to balance the *violated nodes* with other node's data vectors. It is done by incorporating *slack vectors*, namely, $server_i$ would maintain a slack $\overrightarrow{s_i}$. It's important to note that the *Vector Scheme* makes sure that at all times the sum of the slacks is zero: $\sum \overrightarrow{s_i} = \overrightarrow{0}$. In order to take into consideration these *slacks*, a server raises a violation and initiates a communication channel with the coordinator if $c(v_i + s_i)$ exceeds the threshold; specifically, for a upper bound threshold, when $c(v_i + s_i) \leq T$. This ensures that whenever all the local constraint hold, the global constraint (2) holds. proof due to convexity of c and sum of slacks is zero:

$$\begin{aligned} c(v) &= c\left(\frac{1}{n}\sum_{i=0}^{n} v_i\right) = \frac{1}{n}c\left(\sum_{i=0}^{n}(v_i + s_i)\right) \\ &\leq \frac{1}{n}\sum_{i=0}^{n} c(v_i + s_i) \leq \frac{1}{n}(n \cdot T) = T \end{aligned} \tag{3}$$

When a violation occurs, i.e. $c(v_i + s_i) > T$ at a certain server, (3) cannot longer be proven so a *violation resolution protocol* has to occur. In the *violation resolution* phase, the slack vectors are balanced so $c(v_i + s_i)$ would get inside the convex zone. When a server detects a local violation, it sends its local vector $(v_i + s_i)$ to the coordinator, which polls other servers for their local vector as well. When the average of those vectors is inside the convex zone, i.e. $c(E(v_i + s_i)) <= T$ a resolution can be done. Let $(k - 1)$ be the number of polled nodes, then coordinator sends the average vector – $\frac{1}{k}\sum(v_i + s_i)$ to the polled nodes as well as the violated node, which update their slack to be $s_i \leftarrow -v_i + \frac{1}{k}\sum(v_i + s_i)$. Note that the sum of the slack vectors is still zero.
When all the nodes are polled and the average vector still isn't inside the convex zone, a *full sync* has to be done; Since the coordinator knows each node's $(v_i + s_i)$, the coordinator can calculate the global vector v, and hence $f(v)$. Then, the upper bound and lower bound are reset to $(1 \pm \varepsilon)f(v)$ and the monitoring continues with the coordinator notifying the nodes of the new bounds and their local vector – v and their new slack – the zero vector.
Considering the data vectors are of very high dimension, this scheme sends the whole vectors whenever even the slightest violation occurs. Therefore, even though the *Vector Scheme* is better than the naive monitoring scheme, its still wasteful in communication bandwidth.

IV. Value Scheme

The *Value Scheme* is a distributed monitoring scheme done by reducing the bandwidth from sending a whole vector to sending just one scalar. Though, due to the dimensional reduction, we'd expect false alarms to occur than at the *Vector Scheme*, and thus require more *full syncs*. The scalar that will be passed will represent the *value* of the convex bound function.
The *Value Scheme* maintains local scalar slack values; $server_i$ maintains the scalar λ_i. Also, at all times we'd enforce that the sum of the slacks is zero: $\sum \lambda_i = 0$. Here, the local server's constraint is whether $c(v_i) + \lambda_i \leq T$; hence, if all the local constraints are being held, the global constraint (2) will be

held as well:

$$\begin{aligned} c(v) &= c\left(\frac{1}{n}\sum_{i=0}^{n} v_i\right) \leq \frac{1}{n}\sum_{i=0}^{n} c(v_i) \\ &= \frac{1}{n}\sum_{i=0}^{n}(c(v_i)+\lambda_i) \leq \frac{1}{n}(n\cdot T) = T \end{aligned} \tag{4}$$

When a violation occurs, i.e. $c(v_i)+\lambda_i > T$, proof (4) cannot longer hold, thus a violation resolution protocol has to be initiated.

A. Violation Resolution

The violation resolution protocol goes as follows: the violated server $server_i$ sends its local value $c(v_i)+\lambda_i$ which exceeds the threshold. The coordinators tries to 'balance' this scalar value by gradually polling other servers for their $c(v_j)+\lambda_j$ value. Let $(k-1)$ be the number of polled servers plus the violated server, then, when $\frac{1}{k}\sum(c(v_i)+\lambda_i) <= T$ the violation will be resolved by sending $\frac{1}{k}\sum(c(v_i)+\lambda_i)$ to the polled servers and the violated server, which in turn, will set their local scalar slack to $\lambda_i \leftarrow -c(v_i)+\frac{1}{k}\sum(c(v_i)+\lambda_i)$. Its important to note that the sum of the slacks is zero after the resolution, and $c(v_i)+\lambda_i \leq T$ as well.
I fall the nodes are being polled without being able to 'balance' the slack, a *full sync* has to be done, which is very expensive regrading the communication bandwidth.

V. Distance Scheme

**** Note that the convex body is {v | c(v) <= T} ****
The *Distance Scheme* is one more *distributed monitoring scheme* that relies on passing a single scalar when communicating. The scalar that will be passed represents the *distance* of the local vector at the server to the boundary of the convex bound.
This scheme is based on the *Distance Lemma* which is proved below.

A. The Distance Lemma

The *Distance Lemma* states the average of points is inside a convex body, if the sum of distances to the surface of the points inside the convex body is greater than the sum of distances to the boundary of the points which are outside of the convex body. This *Distance Lemma* will be the basis of the *Distance Scheme*.
Let $v_1...v_n$ be vectors and let C be a convex body. If the sum of the distances of the vectors inside C is greater than the sum of the distances to the boundary of the vectors from outside, than the average vector $\frac{1}{n}\sum v_i$ is also inside the convex set.
proof:

1) Let $p_1,...,p_k$ be points inside the convex body C and $l_1,...,l_k$ their distances to the boundary.
2) Let $q_1,...,q_m$ be points outside the convex body C and $d_1,...,d_m$ the distance vectors from the boundary to the points. Let $c_1,...,c_m$ be the points on the boundary which are the source of the distance vectors d_i.
3) Let $d=\sum d_i$ and $l=\sum l_i$
4) Assume $d \geq l$
5) We need to show that $\frac{1}{k+m}(\sum p_i + \sum q_i) \in C$

$$\begin{aligned} &(p_1+...+p_k)+(q_1+...+q_m)) = \\ &(p_1+...+p_k)+(c_1+d_1)+...+(c_m+d_m) = \\ &(p_1+...+p_k)+d+(c_1+...+c_m) = \\ &\left(p_1+\frac{l_1 d}{l}\right)+...+\left(p_k+\frac{l_k d}{l}\right)+(c_1+...+c_m) \end{aligned} \tag{5}$$

$c_i \in C$ by definition - c_i is on the boundary. also, $p_i+\frac{l_i d}{l} \in C$ because $d \geq l$ and l_i is the distance of p_i to the boundary.

B. Monitoring

The *Distance Scheme* is based on the distance value of the local vectors at the servers. The *Distance Scheme* would monitor distributively whether the sum of the distances from inside to the boundary is greater than the sum of the distances to the boundary from the outside. For convenience, denote the distance to the boundary from inside the convex body as negative, and the distance from outside as positive.
Like the *Value Scheme*, the *Distance Scheme*maintains slack scalars that would help 'balance' the distances from the boundary between servers as data changes. Let $server_i$ maintain the scalar λ_i. As in *Value Scheme*, at all times the sum of the slacks would be zero: $\sum \lambda_i = 0$.
Denote d_i the distance of $server_i$'s vector to the boundary, then the local server's monitoring constraint is whether $d_i + \lambda_i \leq 0$. Therefore, if all the local constraints hold, the global vector's distance is negative and by the *Distance Lemma*, the global constraint (2) holds:

$$\sum d_i = \sum(d_i+\lambda_i) \leq n\cdot 0 = 0 \tag{6}$$

The sum of the distances is negative, thus the average vector is inside the convex set.
Whenever $d_i+\lambda_i > 0$ at a certain server, a violation occurs and a *violation resolution protocol* has to occur.

C. Violation Resolution

When the local vector at a server changes so $d_i+\lambda_i > 0$, a violation occurs. In order to resolve the violation, the λ_i slacks has to be balanced so $d_i+\lambda_i \leq 0$ at the violated server as well as other nodes where some slack would be 'borrowed'.
Firstly, in order to resolve the violation, the violated server sends its $d_i+\lambda_i$ value to the coordinator; afterwards, the coordinator gradually polls others servers for their $d_j+\lambda_j$ value. Let $(k-1)$ be the number of polled nodes, than when $\frac{1}{k}(d_i+\lambda_i) \leq 0$ the violation can be resolved, and the coordinator sends the single value $\frac{1}{k}(d_i+\lambda_i)$ to the polled nodes and the violated node. Then, the slacks are updated so $\lambda_i \leftarrow -d_i+\frac{1}{k}(d_i+\lambda_i)$. Note that the sum of slacks is still zero after the update, and at each server the local constraint holds, $d_i+\lambda_i \leq 0$.

VI. Sketched Data Resolution

VII. Sketched Change Resolution

VIII. Experimental Results

IX. Conclusions

X. Summary

References

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.