Bandwidth Efficient Distributed Monitoring Schemes

Abstract

The distributed monitoring problem which tracks changes 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 this problem.

Here, we propose new distributed monitoring schemes which monitor the distributed function using much less communication bandwidth. Existing schemes send high dimensional vectors between the servers while we propose two methods for reducing the communication into one scalar and also two sketch-based methods which are a tradeoff.

1 Introduction

Monitoring the a function over an aggregation of large amount of data which changes

Given convex f

The monitoring objective is to determine whether:

$$f(v) <= T \tag{1}$$

2 Previous Work

3 Vector Scheme

The Vector Scheme's idea is to balance the data vectors of the servers. when a server's local data vector gets out of the function's bound, this scheme would like to balance it with other data vector. It would be 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 (3) $\sum \overrightarrow{s_i} = \overrightarrow{0}$ In order to take into consideration these slacks, a server raises a violation and initiates a communication channel with the coordinator if $f(v_i+s_i)$ exceeds the threshold; specifically, for a lower bound threshold, when $f(v_i+s_i) <= T$. This ensures that whenever all the local constraint hold, the global constraint mentioned in section [1]. proof due to (1), (3):

$$\begin{aligned} f(v) &= f\left(\frac{1}{n}\sum_{i=0}^{n} v_i\right) = \frac{1}{n} f\left(\sum_{i=0}^{n}(v_i+s_i)\right) \\ &<= \frac{1}{n}\sum_{i=0}^{n} f(v_i+s_i) <= \frac{1}{n}(n\cdot T) = T \end{aligned} \tag{2}$$

When a violation occurs, i.e. $f(v_i+s_i) > T$ at a certain server, (2) cannot longer be proven so a violation resolution has to occur. In the violation resolution phase, the slack vectors are balanced so $f(v_i+s_i)$ 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. $f(E(v_i+s_i)) <= T$. after that, the coordinator sends the average vector (k - number of polled nodes plus violated node) $-\frac{1}{k}\sum(v_i+s_i)$ 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 condition (3) still holds.

When all the nodes are polled and the average vector still isn't inside the convex zone, a full sync has to be done, the real value of $f(v)$ is known, so the upper bound and lower bound reset to $(1\pm\varepsilon)f(v)$ and the monitoring continues.

The Vector Schemeis extremely wasteful in communication bandwidth. Considaring that the data vectors are of very high dimension, this scheme sends the whole vectors.

4 Value Scheme

The Value Scheme is distributed monitoring scheme done by reducing the bandwidth from sending a whole vector to one scalar. The whole monitoring scheme is being reduced to one scalar passed. Though, it might have more false alarms than the Vector Scheme, and thus require more full syncs.
The Value Scheme maintains local scalar slack values; $server_i$ maintains the scalar λ_i. like Vector Scheme's (3), $\sum \lambda_i = 0$. Here, the local server's constraint is whether $f(v_i) + \lambda_i <= T$; hence, if all the local constraints are being held, the global constraint (1) will be held as well:

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

When a violation occurs, i.e. $f(v_i) + \lambda_i > T$, proof 3 cannot longer hold, thus a violation resolution protocol has to be initiated.

4.1 Violation Resolution

The violation resolution protocol goes as follows: the violated server $server_i$ sends its local value $f(v_i) + \lambda_i$ which exceeds the threshold. The coordinators tries to 'balance' this scalar value by gradually polling other servers for their $f(v_j) + \lambda_j$ value. Let k be the number of polled servers plus the violated server, then, when $\frac{1}{k}\sum(f(v_i) + \lambda)_i <= T$ the violation will be resolved by sending $\frac{1}{k}\sum(f(v_i) + \lambda_i)$ to the polled servers and the violated server, which in turn, will set their local scalar slack to $\lambda_i \leftarrow -v_i + \frac{1}{k}\sum(f(v_i) + \lambda_i)$. Its important to note that (*) and (**) are still holding and (***) holds as well.

5 Distance Scheme

5.1 Distance Lemma

6 Sketched Data Resolution

7 Sketched Change Resolution

8 Experimental Results

References