SKETCH-BASED GEOMETRIC MONITORING OF DISTRIBUTED STREAM QUERIES

BY KOSTAS ATHANASOGLOU

COMMITTEE:

Professor Minos Garofalakis (Supervisor)

Assistant Professor Vasilis Samoladas

Assistant Professor Antonios Deligiannakis

Submitted to the Department of Electronic and Computer Engineering in partial fulfillment of the requirements for the degree of Master of Science (MoS) in Electronics and Computer Engineering

Technical University of Crete

March 2012

By Kostas Athanasoglou: *Sketch-based Geometric Monitoring of Distributed Stream Queries ,* Submitted to the Department of Electronic and Computer Engineering in partial fulfillment of the requirements for the degree of Master of Science (MoS) in Electronics and Computer Engineering, © March 2012

We have seen that computer programming is an art,
because it applies accumulated knowledge to the world,
because it requires skill and ingenuity, and especially
because it produces objects of beauty.

— Donald E. Knuth

ACKNOWLEDGMENTS

Above all, I would like to thank my family for their lifelong support and the useful advice not only in my years in the university, but in my life in general. Without them, all these beautiful years learning and studying the field of Electronics and Computer Engineering would not be possible.

Special thanks to my supervisor Minos Garofalakis for trusting me this work, for his valuable recommendations regarding the progress of the work and his useful guidance, whenever I had problems.

Many thanks to Mr. Vasilis Samoladas for his useful insight in solving the various geometric problems that arose in this work and suggesting efficient algorithms for finding meaningful solutions and also to Mr. Antonios Deligiannakis for his useful comments regarding the work.

ABSTRACT

Emerging large-scale monitoring applications rely on continuous tracking of complex data-analysis queries over collections of massive, physically-distributed data streams. Thus, in addition to the space- and time-efficiency requirements of conventional stream processing (at each remote monitor site), effective solutions also need to guarantee communication efficiency (over the underlying communication network). The complexity of the monitored query adds to the difficulty of the problem - this is especially true for non-linear queries (e.g., joins), where no obvious solutions exist for distributing the monitor condition across sites. The recently proposed geometric method offers a generic methodology for splitting an arbitrary (non-linear) global threshold-monitoring task into a collection of local site constraints; still, the approach relies on maintaining the complete stream(s) at each site, thus raising serious efficiency concerns for massive data streams. In this paper, we propose novel algorithms for efficiently tracking a broad class of complex aggregate queries in such distributed-streams settings. Our tracking schemes rely on a novel combination of the geometric method with compact sketch summaries of local data streams, and maintain approximate answers with provable error guarantees, while optimizing space and processing costs at each remote site and communication cost across the network. One of our key insights lies in employing the geometric method in the lower-dimensional sketching space for monitoring the sketch-based estimation query. Due to the complex, highly non-linear nature of these estimates, efficiently monitoring the local geometric constraints poses challenging algorithmic issues for which we propose novel solutions. Experimental results on real-life data streams verify the effectiveness of our approach.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

Traditional data-management systems are typically built on a *pull-based paradigm*, where users issue one-shot queries to static data sets residing on disk, and the system processes these queries and returns their results. Recent years, however, have witnessed the emergence of a new class of *large-scale event monitoring* applications, that require the ability to efficiently process continuous, high-volume *streams* of data in real time. Examples include monitoring systems for IP and sensor networks, real-time analysis tools for financial data streams, and event and operations monitoring applications for enterprise clouds and data centers. As both the scale of today's networked systems, and the volumes and rates of the associated data streams continue to increase with no bound in sight, algorithms and tools for effectively analyzing them are becoming an important research mandate.

Large-scale stream processing applications rely on *continuous*, event-driven monitoring, that is, real-time tracking of measurements and events, rather than one-shot answers to sporadic queries. Furthermore, the vast majority of these applications are inherently *distributed*, with several remote monitor sites observing their local, high-speed data streams and exchanging information through a communication network. This distribution of the data naturally implies critical communication constraints that typically prohibit centralizing all the streaming data, due to either the huge volume of the data (e.g., in IP-network monitoring, where the massive amounts of collected utilization and traffic information can overwhelm the production IP network), or power and bandwidth restrictions (e.g., in wireless sensornets, where communication is the key determinant of sensor battery life [15]). Finally, an important requirement of large-scale event monitoring is the effective support for tracking complex, *holistic queries* that provide a global view of the data by combining and correlating information across the collection of remote monitor sites. For instance, tracking aggregates over the result of a distributed *join* (the "workhorse" operator for combining ta-

bles in relational databases) can provide unique, real-time insights into the workings of a large-scale distributed system, including system-wide correlations and potential anomalies [10]. Monitoring the precise value of such holistic queries without continuously centralizing all the data seems hopeless; luckily, when tracking statistical behavior and patterns in large scale systems, *approximate answers* (with reasonable approximation error guarantees) are often sufficient. This often allows algorithms to effectively tradeoff efficiency with approximation quality [10].

Given the prohibitive cost of data centralization, it is clear that realizing sophisticated, large-scale distributed data-stream analysis tools must rely on novel algorithmic paradigms for processing local streams of data *in situ* (i.e., locally at the sites where the data is observed). This, of course, implies the need for intelligently decomposing a (possibly complex) global data-analysis and monitoring query into a collection of *safe, local queries* that can be tracked independently at each site (without communication), while guaranteeing correctness for the global monitoring operation. This decomposition process can enable truly distributed, event-driven processing of real-time streaming data, using a *push-based paradigm*, where sites monitor their local queries and communicate only when some local query constraints are violated [10, 34]. Nevertheless, effectively decomposing a complex, holistic query over the global collections of streams into such local constraints is far from straightforward, especially in the case of *non-linear* queries (e.g., joins) [34].

The need of many recent applications is the ability to process in real time huge amount of data that are stored distributively. A common set-up is: many distributed nodes that are geographically spread, communicate with a central coordinator that is responsible for answering user-provided queries. Each client receives huge amount of data which are called data streams and must communicate with the coordinator as little as possible. Each client does not know the others' data and there are inherent communication constraints. These systems are called data stream systems [5]. Examples of these systems are: sensor nets [29], real-time analysis of financial data [36] and intrusion detection. A useful class of queries are the so called monitoring queries, where the user is interested in monitoring an aggregate value over the collection of the data and receive an alert when something interesting

about the data comes up. This constitutes a shift from the traditional pull paradigm of DBMS where users issue different queries, the system receives these queries and responds with the appropriate answer. Monitoring queries are based on a push paradigm where the coordinator is aware of one or maybe several queries, monitors the data and when any of the queries has a significant change, informs the user about that change. This is usually called threshold crossing when we are interested about the time where the function value has crossed from the one side of the threshold to the other or function value monitoring queries when we are interested on the actual value of the function within specified error-guarantees [13]. Typically in these systems, approximate query answers are sufficient. There is no need to know exactly the query answer. Often a good approximation is enough. The solutions provided in these monitoring queries must be efficient in both space and time requirements. That is, local nodes cannot store the whole data as they are streaming by (considering the huge amount of data and usually the high dimensionality of them), but only a recent history or a summary of them. Solutions must also be communication efficient, that is the local nodes must communicate with the coordinator when something important has happened. That is a threshold crossing for threshold crossing queries or the function value exceeded the required error-guarantees in function value monitoring queries.

Our problem contains technical bottlenecks: even efficient streaming solutions can lead to constant updates and become highly communication-inefficient. To deal with the huge size of data, we are using sketches, that is randomized linear projections of the data on random variables, as presented in [1], [2] and [8]. The use of sketches helps us deal with the limited storage space that is typically available on the local nodes, but also on the rapid-rate which the streams are changing because are very easy to update. We specifically focus on the Fast-AGMS sketches which were presented in [10], which is the latest work and provide the best error guarantees for join operations of the underlying data they represent.

We focus on the join operator which is the most important correlation operator in the relation world. We are often interested in correlating different data, so we focus on the specific operator.

2

RELATED WORK

There is plenty of related work in the field, but several of them differ on important aspects from our work. Some of them, focus on a single data stream and don't consider a distributed environment like [3], [6] and [31]. Others are centralized and consider one-shot computations on the data streams.

These works do not consider communication efficiency issues. The set of queries they consider is broad: quantile-summary computation [22], distinct values [20], set-expression cardinalities [17], counting frequent elements (heavy hitters) [11], [31], approximating large Haar wavelet coefficients [21], estimating join sizes and stream norms [1], [2], [16].

More recent work focuses on communication cost for approximating different queries in a distributed setting, but assume computation is triggered either periodically or in response to a one-shot request, hence it is not applicable for continuous monitoring. This work includes quantiles [22] and heavy hitters [30].

The technique of morphing one-shot solutions to continuous problems entails propagating each change and recomputing the solutions. This quickly becomes communication inefficient or involves periodic updates and other heuristics that can no longer provide real-time estimation guarantees.

Prior research has looked at the monitoring of single values (e.g. SUM of distributed values) [32] proposed a scheme based on adaptive filters, [26] propose building a Kalman filter. BBQ system [15] builds a dynamic multidimensional probabilistic model. This was extended to the continuous case in the Ken system [7].

Closest to our work are [4] (tracking approximate top-k values) and [14] (set-expression cardinalities), as well as distributed quantile tracking [12] (approximately tracking one-dimensional quantile summaries). All these consider tradeoff between accuracy and communicatoin for monitoring a limited class of continuous queries over distributed streams.

Other recent work on distributed trigger monitoring includes [28], [25] and [24].

3

BACKGROUND MATERIAL

3.1 SKETCHES

Through research in the last decade, sketching techniques evolved as the premier approximation technique for aggregate queries over data streams. All sketching techniques share one common feature: they are based on randomized algorithms that combine random seeds with data to produce random variables that have distributions connected to the true value of the aggregate being estimated. By measuring certain characteristics of the distribution, correct estimates of the aggregate are obtained. The interesting thing about all sketching techniques that have been proposed is that the combination of randomization and data is a linear operation with the result that, as observed in [9], [27], sketching techniques can be used to perform distributed computation of aggregates without the need to send the actual data values. The tight connection with both data-streaming and distributed computation makes sketching techniques important from both the theoretical and practical point of view.

Sketches can be used as the actual approximation technique - like in this work, in which case they require a single pass over the data or as the basic technique in multi-pass techniques such as skimmed sketches [18] and red-sketches [19]. For either application, it is important to understand as well as possible its approximation behavior depending on the characteristics of the problem and to be able to predict as accurately as possible the estimation error. As opposed to most approximation techniques - one of the few exceptions are sampling techniques [23] - theoretical approximation guarantees in the form of confidence bounds are provided for all types of sketches from the beginning. All the theoretical guarantees that we know of are expressed as memory and update time requirements in terms of big-$\mathcal{O}$ notation, and are parameterized by ϵ, the target relative error, δ, the target confidence (the relative error

is at most ϵ with probability at least $1-\delta$), and the characteristics of the data - usually the first and the second frequency moments.

3.1.1 *AGMS Sketches*

Techniques based on small-space pseudo-random sketch summaries of the data have proved to be very effective tools for dealing with massive, rapid-rate data streams in a centralized setting [1], [2], [8], [16]. The key idea in such sketching techniques is to represent a streaming frequency vector $\mathbf{f}$ using a much smaller sketch vector (denoted by sk($\mathbf{f}$)) that can be easily maintained as the updates incrementally rendering $\mathbf{f}$ are streaming by. Typically, the entries of the sketch vector sk($\mathbf{f}$) are appropriately defined random variables with some desirable properties that can provide probabilistic guarantees for the quality of the data approximation. More specifically, consider the AGMS (or "tug-of-war") sketches proposed by Alon, Gibbons, Matias, and Szegedy in their seminal papers [1], [2]: The ith entry in an AGMS sketch sk($\mathbf{f}$) is defined as the random variable $\sum_{v=0}^{U-1} \mathbf{f}[v]\xi_i[v]$, where $\{\xi_i[v] : v \in [U]\}$ is a family of four-wise independent binary random variables uniformly distributed in $\{-1,+1\}$ (with mutually-independent families used across different entries of the sketch). The key here is that, using appropriate pseudo-random hash functions, each such family can be efficiently constructed on-line in small (i.e., $\mathcal{O}(\log U)$) space [1]. Note that, by construction, each entry of sk($\mathbf{f}$) is essentially a *randomized linear projection* (i.e., an inner product) of the $\mathbf{f}$ vector (using the corresponding ξ family), that can be easily maintained over the input update stream: Start with each counter sk($\mathbf{f}$)[i] = 0 and, for each i, simply set

$$sk(\mathbf{f})[i] = sk(\mathbf{f})[i] + \xi[v] \quad (respectively, sk(\mathbf{f})[i] = sk(\mathbf{f})[i] - \xi_i[v])$$

whenever an insertion (resp., deletion) of v is observed in the stream. Another critical property is the *linearity* of such sketch structures: Given two "parallel" sketches (built using the same ξ families) sk($\mathbf{f_1}$) and sk($\mathbf{f_2}$) and scalars α, β then

$$sk(\alpha\mathbf{f_1} + \beta\mathbf{f_2}) = \alpha sk(\mathbf{f_1}) + \beta sk(\mathbf{f_2})$$

(i.e., the sketch of a linear combination of streams is simply the linear combination of their individual sketches). The following theorem summarizes some of the basic estimation properties of AGMS sketches (for centralized streams) that we employ in our study (Throughout, the notation $x \in (y \pm z)$ is equivalent to $|x - y| \leqslant z$). For these sketches, we use the standard "inner product" operator over sketch vectors as shorthand for a slightly more complex operator, involving both averaging and median-selection operations over the sketch-vector components [1], [2] – formally, each sketch vector can be viewed as a two-dimensional $n \times m$ array, where $n = \mathcal{O}\left(\frac{1}{\epsilon^2}\right), m = O\left(\log(1/\delta)\right)$ and the "inner product" in the sketch-vector space for both the join and self-join case is defined as

$$sk(\mathbf{f_1})sk(\mathbf{f_2}) = \underset{j=1,\ldots,m}{\text{median}} \left\{ \frac{1}{n} \sum_{i=1}^{n} sk(\mathbf{f_1})[i,j]sk(\mathbf{f_2})[i,j] \right\}$$

Theorem 3.1.1. *([1], [2]) Let sk($\mathbf{f_1}$) and sk($\mathbf{f_2}$) denote two parallel sketches comprising $\mathcal{O}\left(\frac{1}{\epsilon^2}\log(1/\delta)\right)$ counters, built over the streams $\mathbf{f_1}$ and $\mathbf{f_2}$, where $\epsilon, 1-\delta$ denote the desired bounds on error and probabilistic confidence, respectively. Then, with probability at least $1-\delta$, $\|sk(\mathbf{f_1}) - sk(\mathbf{f_2})\|^2 \in (1 \pm \epsilon)\|\mathbf{f_1} - \mathbf{f_2}\|^2$ and $sk(\mathbf{f_1})sk(\mathbf{f_2}) \in (\mathbf{f_1}\mathbf{f_2} \pm \epsilon\|\mathbf{f_1}\|\|\mathbf{f_2}\|)$. The processing time required to maintain each sketch is $\mathcal{O}\left(\frac{1}{\epsilon^2}\log(1/\delta)\right)$ per update.*

Thus, the self-join of the difference of the sketch vectors gives a high-probability, ϵ relative-error estimate of the self-join of the difference of the actual streams (so, naturally, $\|sk(\mathbf{f_1})\|^2 \in (1 \pm \epsilon)\|\mathbf{f_1}\|^2$); similarly, the inner product of the sketch vectors gives a high-probability estimate of the join of the two streams to within an additive error of $\epsilon\|\mathbf{f_1}\|\|\mathbf{f_2}\|$. To provide ϵ relative-error guarantees for the binary join query $\mathbf{f_1}\mathbf{f_2}$. Theorem 3.1.1 can be applied with error bound $\epsilon' = \epsilon(\mathbf{f_1}\mathbf{f_2})/(\|\mathbf{f_1}\|\|\mathbf{f_2}\|)$, giving a total sketching space requirement of $\mathcal{O}\left(\frac{\|\mathbf{f_1}\|^2\|\mathbf{f_2}\|^2}{\epsilon^2(\mathbf{f_1}\mathbf{f_2})^2}\log(1/\delta)\right)$ counters [2].

3.1.2 *Fast-AGMS Sketches*

A drawback of AGMS randomized sketches is that every streaming update must "touch" every component of the sketch vector (to update the corresponding randomized linear projection). This requirement, however, could pose significant practical problems when dealing with

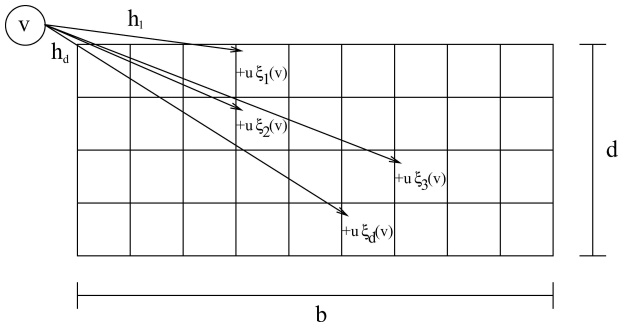


Figure 1. Structure of the Fast-AGMS sketch summary

massive, rapid-rate data streams. Since sketch-summary sizes can vary from tens to hundreds of Kilobytes, especially when tight error guarantees are required, for example, for join or multi-join aggregates [2], [16], touching every counter in such sketches is simply infeasible when dealing with large data rates (e.g., monitoring a high-capacity network link). The proposed Fast-AGMS sketch structure in [10] solves this problem by guaranteeing *logarithmic-time* (i.e., $\mathcal{O}(\log(1/\delta))$) sketch update and tracking costs, while offering essentially the same (in fact, slightly improved) space/accuracy tradeoff as basic AGMS sketches. That is, there is an improvement of the update time from $\mathcal{O}\left(\frac{1}{\epsilon^2}\log(1/\delta)\right)$ to $\mathcal{O}(\log(1/\delta))$.

A Fast-AGMS sketch for a stream $\mathbf{f}$ over $[U]$ (also denoted by $sk(\mathbf{f})$) comprises $b \times d$ counters (i.e., linear projections) arranged in d hash tables, each with b hash buckets. Each hash table $l = 1, \ldots, d$ is associated with: (1) a *pairwise-independent hash function* $h_l()$ that maps incoming stream elements uniformly over the b hash buckets (i.e., $h_l : [U] \to [b]$); and (2) a family $\{\xi_l[v] : v \in [U]\}$ of four-wise independent $\{-1, +1\}$ random variables (as in basic AGMS). To update $sk(\mathbf{f})$ in response to an addition of u to element v, we use the $h_l()$ hash functions to determine the appropriate buckets in the sketch, setting

$$sk(\mathbf{f})[h_l(v), l] = sk(\mathbf{f})[h_l(v), l] + u\xi_l(v),$$

for each $l = 1, \ldots, d$. Note that the required time per update is only $\mathcal{O}(d)$, since each update touches *only one bucket* per hash table. The structure of the sketch is illustrated in Figure 1.

Now, given two parallel Fast-AGMS sketches $sk(\mathbf{f_1})$ and $sk(\mathbf{f_2})$ (using the same hash functions and ξ families), we estimate the inner product $\mathbf{f_1}\mathbf{f_2}$ by the sketch "inner product":

$$sk(\mathbf{f_1})sk(\mathbf{f_2}) = \underset{l=1,\ldots,d}{\text{median}} \left\{ \sum_{i=1}^{b} sk(\mathbf{f_1})[i,l]sk(\mathbf{f_2})[i,l] \right\}$$

In other words, rather than averaging over independent linear projections built over the entire $[U]$ domain, our Fast-AGMS sketch averages over *partitions* of $[U]$ generated randomly (through the $h_l()$ hash functions). As the following theorem shows, this results in essentially identical space/accuracy tradeoffs as basic AGMS sketching, while requiring only $\mathcal{O}(d) = \mathcal{O}(\log(1/\delta))$ processing time per update (since an element only touches a single partition, i.e., bucket, per hash table).

Theorem 3.1.2. *Let sk($\mathbf{f_1}$) and sk($\mathbf{f_2}$) denote two parallel Fast-AGMS sketches of streams $\mathbf{f_1}$ and $\mathbf{f_2}$, with parameters $b = \frac{8}{\epsilon^2}$ and $d = 4\log(1/\delta)$, where $\epsilon, 1-\delta$ denote the desired bounds on error and probabilistic confidence, respectively. Then, with probability at least $1-\delta$, $\|sk(\mathbf{f_1}) - sk(\mathbf{f_2})\|^2 \in (1 \pm \epsilon)\|\mathbf{f_1} - \mathbf{f_2}\|^2$ and $sk(\mathbf{f_1})sk(\mathbf{f_2}) \in (\mathbf{f_1}\mathbf{f_2} \pm \epsilon\mathbf{f_1}\mathbf{f_2})$. The processing time required to maintain each sketch is $\mathcal{O}(\log(1/\delta))$ per update.*

The proof of the theorem is contained in [10].

3.2 GEOMETRIC METHOD AND DISTRIBUTED MONITORING

3.2.1 *Problem Formulation*

Monitoring data streams in a distributed system is the focus of much research in recent years. Most of the proposed schemes, however, deal with monitoring simple aggregated values, such as the frequency of appearance of items in the streams. More involved challenges, such as the important task of feature selection (e.g., by monitoring the information gain of various features), the computation of inner-join and range-queries still require very high communication overhead using naive, centralized algorithms.

Let $S = \{s_1, s_2, \ldots, s_n\}$, be a set of n data streams, collected at nodes $P = \{p_1, p_2, \ldots, p_n\}$. Let $\boldsymbol{\nu_1}(t), \boldsymbol{\nu_2}(t), \ldots, \boldsymbol{\nu_n}(t)$ be d-dimensional real vectors derived from the streams (the value of these vectors varies over

time). These vectors are called *local statistics vectors*. Let $w_1, w_2, \ldots, w_n$ be positive weights assigned to the streams.

The weight w_i assigned to the node p_i usually corresponds to the number of data items its local statistics vector is derived from. Assume, for example, that we would like to determine whether the frequency of occurrence of a certain data item in a set of streams is above a certain threshold value. In this case, the weight we assign to each node at time t is the number of data items received on the stream at time t (and $\boldsymbol{v_i}(t)$ is a scalar holding the frequency of occurence of the item in the stream s_i). In this setup weights change over time. A variant of the problem stated above is for each node to maintain the frequency of occurrence of the item in the recent N_i data items received on the stream (this is known as working with a sliding window of size N_i). In this work, the proposed solution uses sketch summaries instead of the original vectors in each local node and hence, the size of each local vector is fixed and equal to the pre-defined sketch summary size.

Let $\boldsymbol{v}(t) = \frac{\sum_{i=1}^{n} w_i \boldsymbol{v_i}(t)}{\sum_{i=1}^{n} w_i}$. $\boldsymbol{v}(t)$ is called the *global statistics vector*. Let $f : \mathbb{R}^d \to \mathbb{R}$ be an arbitrary function from the space of d-dimensional vectors to the reals. f is called the monitored function. We are interested in determining at any given time, t, whether or not $f(\boldsymbol{v}(t)) > r$, where r is a predetermined threshold value.

We used the coordinator-based setting presented in [34]. The algorithm constructs a vector called the *estimate vector*, denoted by $\boldsymbol{e}(t)$. The estimate vector is constructed from the local statistics vectors collected from the nodes at certain times, as dictated by the algorithm. The last statistics vector collected from the node p_i is denoted by $\boldsymbol{v'}$. Each node remembers the last statistics vector collected from it. The estimate vector is the weighted average of the latest statistics vectors collected from the nodes, i.e., $\boldsymbol{e}(t) = \frac{\sum_{i=1}^{n} w_i \boldsymbol{v'}}{\sum_{i=1}^{n} w_i}$.

From time to time, as dictated by the algorithm, an updated statistics vector is collected from one or more nodes, and the estimate vector is updated. At any given time the estimate vector is known to all nodes.

We designate a coordinator node and denote it by p_1. The coordinator is responsible for collecting local statistics vectors from the nodes, calculating the estimate vector, and distributing it to the nodes. In both settings, each node p_i maintains a parameter called the statistics

delta vector. This vector is denoted by $\Delta\boldsymbol{v}_i(t)$. The statistics delta vector held by the node p_i is the difference between the current local statistics vector and the last statistics vector collected from the node, i.e., $\Delta\boldsymbol{v}_i(t) = \boldsymbol{v}_i(t) - \boldsymbol{v}'$.

In both settings, each node p_i also maintains a parameter called the *drift vector*. This vector is denoted by $\boldsymbol{u}_i(t)$. The algorithm employs a mechanism for balancing the local statistics vectors of a subset of the nodes. Consider the case where at a certain time t the statistics delta vector in two equally weighted nodes, p_i and p_j, cancel each other out: that is $\Delta\boldsymbol{v}_i(t) = -\Delta\boldsymbol{v}_j(t)$. As shown in [34], balancing the local statistics vectors held by p_i and p_j can improve the efficiency of the algorithm. The coordinator facilitates this balancing by sending each node a *slack vector*, denoted by $\boldsymbol{\delta}_i$. The sum of the slack vectors sent to the nodes is $\boldsymbol{0}$. The drift vector held by each node is calculated as follows:

$$\boldsymbol{u}_i(t) = \boldsymbol{e}(t) + \Delta\boldsymbol{v}_i(t) + \frac{\boldsymbol{\delta}_i}{w_i}$$

Throughout this work, all the proposed geometric algorithms make use of this balancing process.

3.2.2 *Geometric Interpretation*

At the heart of the geometric approach is the ability to decompose the monitoring task into local constraints on streams. As data arrives on the streams, each node verifies that the local constraint on its stream has not been violated. It is shown in [34] that as long as none of these constraints have been violated, the query result is guaranteed to remain unchanged, and thus no communication is required. This cannot be done solely by observing the value of the monitored function on each stream. Therefore, an estimated global statistics vector, called the estimate vector, is known to all nodes. The estimate vector is said to be correct at a given time if the value of the monitored function on the estimate vector and the value of the monitored function on the global statistics vector at that time (this value is unknown to any singe node) are on the same side of the threshold. Given an initially correct estimate vector, our goal is to set local constraints on each stream such

that as long as no constraints have been violated, the estimate vector remains correct, and thus no communication is required. The method for decomposing the monitoring task is based on the following, easily verifiable observation: at any given time the weighted average of the drift vectors held by the nodes is equal to the global statistics vector,

$$\frac{\sum_{i=1}^{n} w_i u_i(t)}{\sum_{i=1}^{n} w_i} = v(t)$$

This property is known as the *convexity property* of the drift vectors. The geometric interpretation of this property is that the global statistics vector is in the convex hull of the drift vectors held by the nodes,

$$v(t) \in Conv(u_1(t), u_2(t), \ldots, u_n(t)) \tag{3.1}$$

This observation enables us to take advantage of Theorem 3.2.1 in order to decompose the monitoring task.

Theorem 3.2.1. *Let* $x, y_1, y_2, \ldots, y_n \in \mathbb{R}^d$ *be a set of vectors in* $\mathbb{R}^d$. *Let* $Conv(x, y_1, y_2, \ldots, y_n)$ *be the convex hull of* $x, y_1, y_2, \ldots, y_n$. *Let* $B(x, y_i)$ *be a ball centered at* $\frac{x+y_i}{2}$ *and with a radius of* $\left\|\frac{x-y_i}{2}\right\|$, *i.e.* $B(x, y_i) = \left\{z \middle| \left\|z - \frac{x+y_i}{2}\right\| \leqslant \left\|\frac{x-y_i}{2}\right\|\right\}$, *then* $Conv(x, y_1, y_2, \ldots, y_n) \subset \bigcup_{i=1}^{n} B(x, y_i)$.

Theorem 3.2.1 is used to bound the convex hull of $n+1$ vectors in $\mathbb{R}^d$ by the union of n d-dimensional balls. In our case it is used to bound the convex hull of the estimate vector and the drift vectors i.e., $Conv(x, y_1, y_2, \ldots, y_n)$, by a set of n balls, where each ball is constructed independently by one of the nodes. Each node, p_i, constructs a ball $B(x, y_i)$ which is centered at $\frac{e(t)+u_i(t)}{2}$ and has a radius of $\left\|\frac{x-y_i}{2}\right\|$. Note that at any given time each node has all the information required to independently construct its ball. Theorem 3.2.1 states that $Conv(e(t), u_1(t), u_2(t), \ldots, u_n(t)) \subset \bigcup_i B(e(t), u_i(t))$.

The application of Theorem 3.2.1 is illustrated in Figure 2, which depicts a setup comprised of 5 nodes, each holding a statistics vector $v_i(t) \in \mathbb{R}^2$. The drift vectors held by the nodes $(u_1(t), \ldots, u_t(t))$, the global statistics vector $v(t)$ and the estimate vector $e(t)$ are depicted, as are the balls constructed by the nodes. The convex hull of the drift vectors is highlighted in gray, and one can see that, as the theorem

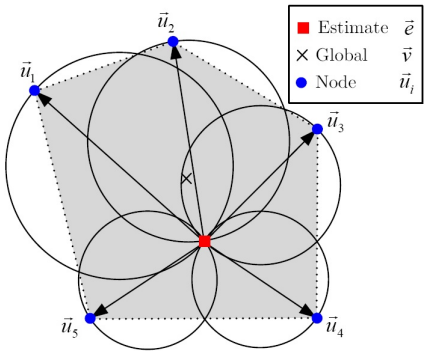


Figure 2. Illustration of Theorem 3.2.1. The drift vectors held by 5 nodes and the balls constructed by them are depicted. The convex hull of the drift vectors is highlighted in gray. As stated by the theorem, the union of the balls bounds the convex hull.

states, the area defined by the convex hull is bounded by the set of balls.

3.2.3 *Local Constraints*

The local constraint on each stream is set as follows: the monitored function f and threshold r can be seen as inducing a coloring over $\mathbb{R}^d$. The vectors $\{x|f(x) > r\}$ are said to be green, while the vectors $\{y|f(y) < r\}$ are said to be red. The local constraint each node maintains is to check whether the ball $B(e(t), u_i(t))$ (the ball centered at $\frac{e(t)+u_i(t)}{2}$ and having a radius of $\left\|\frac{e(t)-u_i(t)}{2}\right\|$ is monochromatic, i.e., whether all the vectors contained in the ball have the same color. Testing for monochromicity is done by finding the maximal and minimal values of f in the ball. This is done locally at each node hence has no effect on the communication load.

If all the local constraints are upheld, the estimate vector is correct: because all the balls contain the estimate vector, and all the balls are monochromatic, the set of vectors defined by the union of all the balls is monochromatic as well. Since the union of all the balls contains the convex hull of the drift vectors and the estimate vector $Conv(e(t), u_1(t), u_2(t), \ldots, u_n(t))$, and according to Equation (3.1) the global statistics vector is contained in the convex hull of the drift vectors, the estimate vector and the global statistics vector have the same color. Therefore, they are on the same side of the threshold, i.e., the estimate vector is correct.

4

COMPUTING THE SELF-JOIN

We have a distributed environment consisting of n nodes and a *global vector* ν_t at time t. The global vector is the sum of the *local vectors* $\nu_t^{(i)}$ observed at each node i at time t: $\nu_t = \sum_{i=1}^{n} \nu_t^{(i)}$.

We predefine one of the n nodes as the coordinator, which is responsible for answering our query; the self join of the global vector: $f(\nu_t) = \|\nu_t\|^2$.

Throught the section, we use the symbols summarized in Table 1.

4.1 BASIC THRESHOLD CROSSING PROBLEM

Before discussing the more complicated function monitor problem, let's illustrate the simpler threshold crossing problem. In a threshold crossing query, we are interested if this function value crosses a predetermined threshold τ.

One trivial way for solving this problem is for every time instant t, each local node sends its local vector $\nu_i^{(t)}$ to the coordinator and the coordinator calculates the value of $f(\nu_t)$. If $f(\nu_t)$ crosses the threshold τ, we issue an alert. This method is highly communication-inefficient because the local nodes need to contact the coordinator at each time t. We seek to find a more efficient way of calculating the threshold crossing, by finding appropriate local conditions in each node i. Then, the node i will test only its local conditions at each time t, without communicating with the coordinator. The node must remain silent (no communication overhead) as long as there is no threshold violation.

The most effective solution known so far is the geometric method discussed in Section 3.2. Each local node finds a safe zone for its local node $\nu_i^{(t)}$. As long as the value of $\nu_i^{(t)}$ remains in this zone, it is guaranteed that the global vector ν_t won't cross the threshold. Note that this method tests the *domain* of the function and not the function value itself.

Table 1. Notation Summary

Notation	Description
n	Total number of nodes.
θ	Total error bound we want to guarantee for query answering.
ϵ	Sketching error.
$\boldsymbol{v}_t^{(i)}$	Local statistics vector at time t at node i.
$\boldsymbol{v}_t$	Global statistics vector at time t: $\boldsymbol{v}_t = \sum_{i=1}^{n} \boldsymbol{v}_t^{(i)}$.
$\tilde{\boldsymbol{v}}_t^{(i)}$	Local sketched statistics vector at time t at node i.
$\tilde{\boldsymbol{v}}_t$	Global sketched statistics vector at time t: $\tilde{\boldsymbol{v}}_t = \sum_{i=1}^{n} \tilde{\boldsymbol{v}}_t^{(i)}$.
$\boldsymbol{\delta}_t^{(i)}$	Local drift vector of node i at time t.
$\boldsymbol{\delta}_t$	Total drift vector at time t: $\boldsymbol{\delta}_t = \sum_{i=1}^{n} \boldsymbol{\delta}_t^{(i)}$.
$\boldsymbol{x}_t$	A point inside the local ball that each node constructs.
$\tilde{\boldsymbol{x}}_t$	The sketched version of $\boldsymbol{x}_t$.
$\boldsymbol{e}_t$	The estimation vector sent from the coordinator to each local node.

At time $t = 1$ a global synchronization occurs at our system so each local node i sends $\boldsymbol{v}_i^{(t)}$ to the coordinating node and $\boldsymbol{v}_1$ is computed and distributed to all nodes (that is the initial estimation vector). Whenever there is a global synchronization at time t and $\boldsymbol{v}_t$ is computed by the coordinator, the resulting vector is the estimation vector, that is the last estimated position of the global vector.

At subsequent times, each node observes modifications to its local vector, the drift vectors, which are denoted as $\mathbf{u}_n^{(t)}$.

4.2 FROM THRESHOLD CROSSING TO VALUE MONITORING

In the function value monitoring problem, we want the coordinator to provide query answers of $f(\boldsymbol{v}_t)$ within θ error bounds. If we allow a global synchronization step at time $t = s$, where each node n sends its local vector $\boldsymbol{v}_s^{(n)}$ to the coordinator site, then the coordinator can calculate the function value of $\boldsymbol{v}_s = \sum_{i=1}^{n} \boldsymbol{v}_s^{(n)}$, which will then distribute to the nodes as the estimate vector. We want to guarantee that the value provided as answer from the coordinator is bounded by:

$$(1-\theta)f(\boldsymbol{v}_t) \leqslant f(\boldsymbol{v}_s) \leqslant (1+\theta)f(\boldsymbol{v}_t)$$

The running quantity $\boldsymbol{v}_t$ is bounded by the balls of each local node, constructed with center the estimate vector $\boldsymbol{v}_s$ sent from the coordinator to each local node at time $t = s$ and diameter the local drift vector

multiplied by the number of nodes n: $n\delta_t^{(i)}$ of the node i. So, to ensure the above equation, each node i must construct locally a ball $B_i(\boldsymbol{v}_s, \delta_t^{(i)}) = \left\{ \boldsymbol{z} \middle| \left\| \boldsymbol{z} - \frac{\boldsymbol{v}_s + \delta_t^{(i)}}{2} \right\| \leqslant \left\| \frac{\boldsymbol{v}_s - \delta_t^{(i)}}{2} \right\| \right\}$ and check locally if:

$$(1-\theta) \max_{\boldsymbol{x}_t \in B_i} f(\boldsymbol{x}_t) \leqslant f(\boldsymbol{v}_s) \leqslant (1+\theta) \min_{\boldsymbol{x}_t \in B_i} f(\boldsymbol{x}_t)$$

and if $f(\cdot) > 0$, then this is the same as:

$$\begin{aligned} \max_{\boldsymbol{x}_t \in B_i} f(\boldsymbol{x}_t) &\leqslant \frac{f(\boldsymbol{v}_s)}{1-\theta} \\ \min_{\boldsymbol{x}_t \in B_i} f(\boldsymbol{x}_t) &\geqslant \frac{f(\boldsymbol{v}_s)}{1+\theta} \end{aligned} \tag{4.1}$$

For the case where $f(\cdot) < 0$, the bounds are reversed.

So by setting the above thresholds at each local node, we can guarantee that the value $f(\boldsymbol{v}_s)$ used by our coordinator to answer our queries, will be within θ bounds from the actual function value of the global vector.

4.3 USING SKETCH SUMMARIES

To further reduce communication between nodes and the coordinator, we can use sketches instead of the original vectors. The sketch linearity property makes the convexity property hold, so it enables us this option. As already stated, we will use the Fast-AGMS sketch summaries, which are proven to be the best choice for estimating join sizes.

Let's call $\tilde{\boldsymbol{v}}_t^{(n)}$ the Fast-AGMS sketch of the local vector $\boldsymbol{v}_t^{(n)}$ and $\tilde{\boldsymbol{v}}_t$ the Fast-AGMS sketch of the global vector $\boldsymbol{v}_t$, where t denotes time.

The use of sketches in the query answer estimation (where the function used is $f(\boldsymbol{v}_t) = \|\boldsymbol{v}_t\|^2$) introduces an additional *sketching error* ϵ:

$$\|\tilde{\boldsymbol{v}}_t\|^2 \in (1 \pm \epsilon) \|\boldsymbol{v}_t\|^2 \Leftrightarrow f(\tilde{\boldsymbol{v}}_t) \in (1 \pm \epsilon) f(\boldsymbol{v}_t) \tilde{\boldsymbol{v}}_t \tag{4.2}$$

So we must adjust the local conditions found in Eq.(4.1) to take into consideration these sketching errors:

$$\begin{aligned} \frac{f(\tilde{\boldsymbol{v}}_s)}{1+\epsilon} \leqslant f(\boldsymbol{v}_s) \leqslant \frac{f(\tilde{\boldsymbol{v}}_s)}{1-\epsilon} \\ \frac{f(\tilde{\boldsymbol{x}}_t)}{1+\epsilon} \leqslant f(\boldsymbol{x}_t) \leqslant \frac{f(\tilde{\boldsymbol{x}}_t)}{1-\epsilon} \end{aligned} \tag{4.3}$$

Combining the above two equations with Eq. (4.1), we get:

$$\begin{aligned} \min_{\tilde{\mathbf{x}}_t \in B_i} \left(\frac{f(\tilde{\mathbf{x}}_t)}{1+\epsilon} \right) &\geqslant \frac{f(\tilde{\mathbf{v}}_s)}{(1-\epsilon)(1+\theta)} \\ \max_{\tilde{\mathbf{x}}_t \in B_i} \left(\frac{f(\tilde{\mathbf{x}}_t)}{1-\epsilon} \right) &\leqslant \frac{f(\tilde{\mathbf{v}}_s)}{(1+\epsilon)(1-\theta)} \end{aligned} \tag{4.4}$$

and solving for $f(\tilde{\mathbf{x}}_t)$:

$$\begin{aligned} \min_{\tilde{\mathbf{x}}_t \in B_i} f(\tilde{\mathbf{x}}_t) &\geqslant \frac{f(\tilde{\mathbf{v}}_s)(1+\epsilon)}{(1-\epsilon)(1+\theta)} \\ \max_{\tilde{\mathbf{x}}_t \in B_i} f(\tilde{\mathbf{x}}_t) &\leqslant \frac{f(\tilde{\mathbf{v}}_s)(1-\epsilon)}{(1+\epsilon)(1-\theta)} \end{aligned} \tag{4.5}$$

The following conditions must be met:

$$\begin{aligned} \frac{1+\epsilon}{(1-\epsilon)(1+\theta)} &< 1 \\ \frac{1-\epsilon}{(1+\epsilon)(1-\theta)} &> 1 \end{aligned} \tag{4.6}$$

and by combining them, gives us the condition: $\theta > \frac{2\epsilon}{1-\epsilon}$.

4.4 USING *mean* FOR SKETCH ESTIMATION

REASONING FOR USING THE MEAN An interesting property of the AGMS-style sketches presented in [33], is that if we use AGMS-sketches and average over k independent estimators $Y = \frac{1}{k}X$, each one having the form $X = \tilde{\mathbf{v}}_t^2$, then Y follows a normal distribution and we can use the mean, instead of the median to reduce the variance of the estimator. If each node holds $\tilde{\mathbf{v}}_t^{(n)}$ which is the local average of the sketches produced by k independent sketching families, then we can assume that $\tilde{\mathbf{v}}_t$ follows a normal distribution and the join can be estimated by its mean: $f(\tilde{\mathbf{v}}_t) = \frac{1}{n}\sum_{j=1}^{n} v_j^2$, where $\tilde{\mathbf{v}}_t = [v_1, \ldots, v_n]$.

EFFICIENT CALCULATION USING THE MEAN In order for each local node n, to check if $f(\tilde{\mathbf{v}}_t^{(n)})$ has crossed any of the two thresholds calculated in Eq. (4.5), it must check if the function values of the ball B defined by the center $\tilde{\mathbf{v}}_t^{(n)}$ and having a radius of $\tilde{\mathbf{u}}_t^{(n)}$, where $\tilde{\mathbf{u}}_t^{(n)}$ the drift vector of $\tilde{\mathbf{v}}_t^{(n)}$ are monochromatic or not.

If we can efficiently calculate the maximum and the minimum function value inside this ball B, then we can compare them with the threshold and see if we have a crossing or not.

This problem is the same as finding the minimum and the maximum values of x, where x is constrained in a ball of center a (that's our sketch vector $\tilde{v}_t^{(n)}$) and with a radius of r (that's the distance of the drift vector from the sketch vector $r = \left|\tilde{v}_t^{(n)} - \tilde{u}_t^{(n)}\right|$):

$$\max / \min \left\{ f(x) = x^2 \right\} : |x - a|^2 \leqslant r^2$$

Theorem 4.4.1. *The value of* $f(x) = x^2$ *inside the ball* $|x - a|^2 \leqslant r^2$ *is the distance of* x *from the origin and it is minimized/maximized in the two points* $x_m = a - r\frac{a}{|a|}$, $x_M = a + r\frac{a}{|a|}$ *respectively that are found on the intersection of the surface of the ball with the line that starts from the origin and passes through the center* a *of the ball.*

Proof. We want to show that $|x_m|^2 \leqslant |x|^2 \leqslant |x_M|^2, \forall x \in |x - a|^2 \leqslant r^2$. The squares of the magnitudes of x_m and x_M are:

$$|x_m|^2 = |a|^2 + r^2 - 2|ar| = (|a| - r)^2$$

$$|x_M|^2 = |a|^2 + r^2 + 2|ar| = (|a| + r)^2$$

Now x can be written as $x = a + u$ with $|u| \leqslant r$ and its magnitude satisfies the following equations from triangle inequality:

$$|a| - |u| \leqslant |a + u| \leqslant |a| + |u|$$

If we square the above equations and take into consideration that $|u| \leqslant r$ we conclude our proof. □

We also present in Appendix A, a computational solution for finding x_m, x_M using Lagrange multipliers.

4.5 USING *median* FOR SKETCH ESTIMATION

At first we state some properties of the median. We define the operator $M_i[x] = \underset{i}{\text{median}}\{x_i\}$ for shorthand.

Theorem 4.5.1. *If* $f : \mathbb{R} \to \mathbb{R}$ *is monotonic (increasing or decreasing) then* $M_i[f(x)] = f(M_i[x])$

Corollary 4.5.1. $M_i[\alpha x + \beta] = \alpha M_i[x] + \beta$

Theorem 4.5.2. *If* $\mathbf{x_1} \prec \mathbf{x_2}$ *(that is* $\mathbf{x_1}$ *dominates* $\mathbf{x_2}$*) then* $M_i[\mathbf{x_1}] \leqslant M_i[\mathbf{x_2}]$

Corollary 4.5.2. *If* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ *then* $M_i[\mathbf{x}] + \min\{\mathbf{y}\} \leqslant M_i[\mathbf{x}+\mathbf{y}] \leqslant M_i[\mathbf{x}] + \max\{\mathbf{y}\}$

Corollary 4.5.3. $M_i[\mathbf{x}]$ *is continuous. That is* $M_i[\mathbf{a}]$ *is defined,* $\lim_{x\to a} M_i[\mathbf{x}]$ *exists and* $\lim_{x\to a} M_i[\mathbf{x}] = M_i[\mathbf{a}]$

The original definition for the self-join of a vector using sketches is:

$$f(\tilde{\mathbf{v}}_t) = \underset{j=1,\ldots,n}{\text{median}} \left\{ \frac{1}{m} \sum_{i=1}^{m} \tilde{\mathbf{v}}_t[i,j]^2 \right\}$$

Here, $\tilde{\mathbf{v}}_t$ is seen as an $m \times n$ array, where we first take the average sum of the squares of each column and then take the median of these resulting averages.

We define each column of this matrix as $\boldsymbol{v_i} \in \mathbb{R}^m$ and $\boldsymbol{v_i}^2 \in \mathbb{R}$ is the sum of the squares of each column. Now, the problem is transformed into finding these $\boldsymbol{v_i}$, that maximize (respectively minimize) $M_i\left[\frac{1}{m}\boldsymbol{v_i}^2\right]$ and because the scaling factor $\frac{1}{m}$ does not modify our answer, it is the same as maximizing/minimizing $M_i\left[\boldsymbol{v_i}^2\right]$ with the constraint that $\boldsymbol{v} = [\boldsymbol{v_1} \ldots \boldsymbol{v_n}] \in \mathbb{R}^{m\times n}$ is constrained within a ball of center $\mathbf{c} = [\mathbf{c_1} \ldots \mathbf{c_n}] \in \mathbb{R}^{m\times n}$ and a radius $r \in \mathbb{R}$: $|\boldsymbol{v} - \mathbf{c}| \leqslant r^2$.

Assume that $\mathbf{s} = [\mathbf{s_1} \ldots \mathbf{s_n}] \in \mathbb{R}^{m\times n}$ is a solution that maximizes $M_i\left[\mathbf{s_i}^2\right]$ and that each $\mathbf{s_i}$ has a distance ρ_i from $\mathbf{c_i}$: $\rho_i = |\mathbf{s_i} - \mathbf{c_i}|$.

The key observation in our algorithm is that a solution $M_i\left[\mathbf{s_i}^2\right]$ will always be dominated by a solution $M_i\left[\left(\mathbf{c_i} + \rho_i \frac{\mathbf{c_i}}{|\mathbf{c_i}|}\right)^2\right] = M_i\left[(|\mathbf{c_i}| + \rho_i)^2\right]$, so the initial abstract problem of finding the maximum $M_i\left[\mathbf{s_i}^2\right]$ is now broken down into the more specific problem of finding the right ρ_i for each $\mathbf{s_i}$. Also notice that $\sum_{i=1}^{n} \rho_i^2 = r^2$:

$$\sum_{i=1}^{n} \rho_i^2 = \sum_{i=1}^{n} |\mathbf{s_i}|^2 + \sum_{i=1}^{n} |\mathbf{c_i}|^2 - 2\sum_{i=1}^{n} \mathbf{s_i}\mathbf{c_i} = |\mathbf{s}|^2 + |\mathbf{c}|^2 - 2\mathbf{s}\mathbf{c} = r^2$$

So the problem now is try to distribute r^2 to $\rho_1, \ldots, \rho_n$ in order to maximize $M_i\left[\mathbf{s_i}^2\right]$. We propose an iterative algorithm (Algorithm 1) to solve this problem. The algorithm is based on the following Theorem:

Theorem 4.5.3. *All centers* $\mathbf{c_i}$ *that have received a radius* ρ_i*, will be equal to the maximum median* $\hat{\mu} = \max\{M_i\,[\mathbf{c_i} + \rho_i]\}$*. That is:*

$$\forall i: \quad \rho_i > 0 \Rightarrow \mathbf{c_i} + \rho_i = \hat{\mu}$$

Proof. We define $\mathcal{S} = \{\mathbf{s_i} = \mathbf{c_i} + \rho_i,\ \rho_i \geqslant 0\}$ as the set with all the possible solutions $\mathbf{s_i}$ that have received a radius ρ_i.

If $|\mathcal{S}| = 1$ then it will contain $M_i\,[c_i]$ which is the median when none of $\mathbf{s_i}$ has received any radius. All of the available r^2 will be given to $M_i\,[c_i]$ which will either continue to be the median, or it will become greater than the next ordered element $\mathbf{c_{i+1}}$. In the first case, $\mathbf{s_i}$ will be the maximum median so it is proven. In the latter case, $\mathbf{c_{i+1}}$ will also receive a radius, so $|\mathcal{S}| > 1$, which contradicts our original assumption.

If $|\mathcal{S}| > 1$, let's argue that one of them, say $\mathbf{s_j}$, has received a radius but is not equal to the maximum median. That is there is $j : \rho_j \geqslant 0$ for which $\mathbf{c_j} + \rho_j \neq \hat{\mu}$.

Suppose that $\mathbf{c_j} + \rho_j \geqslant \hat{\mu}$. Hence, $\mathbf{c_j} + \rho_j \geqslant \mathbf{c_i} + \rho_i$, $\forall(\mathbf{c_i} + \rho_i) \in \mathcal{S}$ with $i \neq j$. Then, we can distribute the quantity $\epsilon = \mathbf{c_j} + \rho_j - \hat{\mu}$ to these $\mathbf{c_i} + \rho_i \in \mathcal{S}$ with $i \neq j$ and take a set $\mathcal{S}' = \{\mathbf{c_i} + p_i + \epsilon_i\}$ such that $\mathbf{c_i} + \rho_i + \epsilon_i \geqslant \hat{\mu}\,\forall(\mathbf{c_i} + \rho_i + \epsilon_i) \in \mathcal{S}$. Then, $\hat{\mu}$ is no longer $\max\{M_i\,[\mathbf{c_i} + \rho_i]\}$.

Accordingly we can prove it, for the case that $\mathbf{c_j} + \rho_j \leqslant \hat{\mu}$. □

Based on Theorem 4.5.3, we have that the maximum median $\hat{\mu}$ will be equal to:

$$\hat{\mu} = \mathbf{c}_{\frac{n}{2}} + \rho_{\frac{n}{2}} = \mathbf{c}_{\frac{n}{2}+1} + \rho_{\frac{n}{2}+1} = \ldots = \mathbf{c}_{\frac{n}{2}+k-1} + \rho_{\frac{n}{2}+k-1}$$

So, we need to find that k for the following equation to be true:

$$\sum_{i=n/2}^{n/2+k-1} (\mu - \mathbf{c_i})^2 = r^2$$

Expanding the square, we can see that k is found by solving this second degree equation:

$$k\mu^2 - 2\mu \sum_{i=n/2}^{n/2+k-1} \mathbf{c_i} + \sum_{i=n/2}^{n/2+k-1} \mathbf{c_i}^2 - r^2 = 0$$

$$\mu = \frac{\sum_{i=n/2}^{n/2+k-1} \mathbf{c_i} + \sqrt{\sum_{i=n/2}^{n/2+k-1} \mathbf{c_i} - k(\sum_{i=n/2}^{n/2+k-1} \mathbf{c_i}^2 - r^2)}}{k} \tag{4.7}$$

Our algorithm progresses by increasing k, while $\mu \leqslant \mathbf{c_{n/2+k}}$.

Algorithm 1: MaxMedian

Data: $\mathbf{c} = [|\mathbf{c_1}|, \ldots, |\mathbf{c_n}|]$: array with center magnitudes, r^2: radius of the ball
Result: Find ρ_i so $M_i \left[\mathbf{s_i}^2\right]$ becomes maximum

```
begin
    avail ⟵ r² ; // available radius to distribute among ρ_i
    sort(c, ascending)
    k ⟵ 1
    while true do
        m ⟵ Equation (4.7)
        if m ⩽ c_{n/2+k} then
            return m
        else
            k ⟵ k + 1
end
```

5

HANDLING RANGE QUERIES

We now turn our attention to a different type of problem, which models a join with a constant relation. In a frequent special case, that of range queries, the constant relation ζ may contain exactly one record for each value of the join attribute in a given subset. Thus, $\zeta \cdot \nu_t$ is the number of tuples of ν_t "in range" ζ. However, any constant relation ζ can be used, provided that its Fast-AGMS sketch $\tilde{\zeta} = [\zeta_1, \ldots, \zeta_n], \zeta_i \in \mathbb{R}^m$ is available. The relevant threshold function to monitor is

$$f(\tilde{\mathbf{x}}) = M_i\,[\zeta_i x_i] \tag{5.1}$$

We shall tackle this problem by trying to bound the ball radius. So, assume that we wish to compute

$$r_> = \inf\{\|\tilde{\mathbf{x}} - \tilde{\mathbf{c}}\| \;\big|\; f(\tilde{\mathbf{x}}) \geqslant \tau\}$$

In the trivial case where $f(\tilde{\mathbf{c}}) \geqslant \tau$, it is trivially $r_> = 0$. So, assume $\mu = f(\tilde{\mathbf{c}}) < \tau$. Our goal is to compute a minimum-length change u to c, so that

$$f(c + u) = M_i\,[\zeta_i(c_i + u_i)] \geqslant \tau$$

while $r_>^2 = \sum_i u_i^2$ is minimum.

Let us consider the properties of such an optimum u. First, we note that if $\zeta_i = 0$, then also $u_i = 0$. Then, for $\zeta_i \neq 0$,

$$\zeta_i(c_i + u_i) \leqslant \zeta_i(c_i + \|u_i\| \frac{\zeta_i}{\|\zeta_i\|}) = \zeta_i c_i + \|\zeta_i\|\,\|u_i\|$$

and we can change u_i to $\|u_i\| \frac{\zeta_i}{\|\zeta_i\|}$ while retaining optimality.

Finally, the main observation is that $u_i \neq 0$ *implies* $\zeta_i(c_i + u_i) = \tau$, for, if that is not the case, then u is not optimal; we can substitute u_i by a slightly shorter vector, without affecting the overall median, but slightly decreasing the overall distance $r_>$.

Now, let us consider the set of indices

$$L = \{i | \zeta_i c_i < \tau\}.$$

Since we have $M_i[\zeta_i c_i] < \tau$, it is clear that L contains at least half of the total indices, $|L| \geqslant (n+1)/2$. An optimal solution can be found by selecting a subset $U \subseteq L$ of size $|U| = |L| - (n-1)/2$, and, for each $i \in U$, computing an appropriate u_i such that $\zeta_i(c_i + u_i) = \tau$. Clearly, $u_i = \rho_i \zeta_i / \|\zeta_i\|$ with

$$\rho_i = (\tau - \zeta_i c_i) / \|\zeta_i\|$$

and $r_>^2 = \sum_{i \in U} \rho_i^2$.

Overall, the algorithm we propose is the following:

Algorithm 2: Computation of $r_>$ for range queries

```
Data: c = [c_1, ..., c_n]: array with center
τ: target median
Result: Find minimum r_> = ‖u‖ such that M_i[ζ_i(c_i + u_i)] ⩾ τ
begin
    L = {i | ζ_i c_i < τ}
    if |L| < (n+1)/2 then
        return 0; // Trivial case
    for i ∈ L do
        ρ_i = (τ − ζ_i c_i)/‖ζ_i‖      // If ζ_i = 0 ρ_i = ∞
    k = |L| − (n−1)/2
    U = {select k elements of L with the least ρ_i}
    return sqrt(Σ_{i∈U} ρ_i^2)
end
```

The algorithm for $r_<$ can be derived in a straightforward way similarly to the one above.

6

EXTENDING TO INNER-JOINS

We now turn our attention to a more complicated monitoring problem, where the global statistic comprises of the concatenation of two sketches

$$\tilde{\mathbf{v}}_t = < \tilde{\mathbf{x}}_t, \tilde{\mathbf{y}}_t >$$

corresponding to two relations, and the monitored function is the sketch estimate of the inner product of the sketched vectors, corresponding to the size of the inner join:

$$f(\tilde{\mathbf{v}}_t) = M_i\,[x_i y_i]$$

where $\tilde{\mathbf{x}}_t = [x_1, \ldots, x_n]$ and $\tilde{\mathbf{y}}_t = [y_1, \ldots, y_n]$.

For this problem, essentially the same reasoning applied to the range query case leads us to a bound-of-ball-radius solution. Again, we focus on the computation of $r_>$.

Thus, we are given two centers $\tilde{\mathbf{a}} = [a_1, \ldots, a_n]$ and $\tilde{\mathbf{b}} = [b_1, \ldots, b_n]$. Our goal is to minimize $r_>^2 = \sum_i (p_i - a_i)^2 + (q_i - b_i)^2$ under the constraint

$$M_i\,[p_i q_i] \geqslant \tau.$$

Let us consider the set of indices

$$L = \{i \mid a_i b_i < \tau\}$$

If $|L| < (n+1)/2$, we are done, since $r_> = 0$ is the trivial optimum solution. Else, we must compute the cost ρ_i for each index $i \in L$ and choose a subset $U \subseteq L$ of size $|L| - (n-1)/2$ of indices of least cost ρ_i, which will yield the minimum radius $r_>$.

The cost ρ for some index i (which we drop for clarity) can be specified as a constrained optimization problem: given a, b, minimize

$\rho^2 = (p-a)^2 + (q-b)^2$ under the constraint $pq = \tau$. We now consider the following cases:

CASE $a = b$ In this case, we wish to minimize $(p-a)^2 + (q-a)^2$ for $pq \geqslant \tau > a^2$. Given any pair of vectors p, q with $pq \geqslant \tau$, it is easy to check that vectors $p' = \|p\| \frac{a}{\|a\|}$ and $q' = \|q\| \frac{a}{\|a\|}$ have $p'q' \geqslant \tau$ and also

$$(p'-a)^2 + (q'-a)^2 \leqslant (p-a)^2 + (q-a)^2.$$

Thus, for an optimal pair p, q, both vectors will be collinear to a. Solving the corresponding one-dimensional problem yields a cost

$$\rho^2 = 2(\sqrt{\tau} - \|a\|)^2$$

CASE $a = -b$ We wish to minimize $(p-a)^2 + (q+a)^2$ for $pq \geqslant \tau > -a^2$. Using Lagrange multipliers,

$$\Lambda = (p-a)^2 + (q+a)^2 - \lambda(pq - \tau)$$

and from $\partial\Lambda/\partial p = \partial\Lambda/\partial q = 0$ we get

$$2(p-a) - \lambda q = 2(q+a) - \lambda p = 0 \tag{6.1}$$

Thus, $(\lambda - 2)(p+q) = 0$. Now, we have two cases:

$\lambda = 2$ implies $p - q = a$ (by Eq. 6.1) and $\tau \geqslant -a^2/4$. In this case, the quantity to minimize is

$$\rho^2 = (p-a)^2 + (q+a)^2 = p^2 + q^2 = (p-q)^2 + 2pq = a^2 + 2\tau$$

$\lambda \neq 2$ implies $p + q = 0$ and thus $pq = -p^2 = \tau \leqslant 0$. Also, $(2+\lambda)p = 2a$ implies that

$$\rho^2 = 2(p-a)^2 = 2(\sqrt{-\tau} - |a|)^2$$

Putting everything together, we have

$$\rho^2 = \begin{cases} 0 & \text{if } \tau < -a^2, \\ 2(\sqrt{-\tau} - a)^2 & \text{if } -a^2 \leqslant \tau \leqslant -a^2/4, \\ 2\tau + a^2 & \text{if } -a^2/4 < \tau. \end{cases}$$

CASE $a \neq \pm b$ We employ Lagrange multipliers to this case. The Lagrangian is

$$\Lambda = (p-a)^2 + (q-b)^2 - \lambda(pq - \tau)$$

and we obtain the system

$$\frac{\partial \Lambda}{\partial p} = 2(p-a) - \lambda q = 0 \tag{6.2}$$
$$\frac{\partial \Lambda}{\partial q} = 2(q-b) - \lambda p = 0 \tag{6.3}$$
$$\frac{\partial \Lambda}{\partial \lambda} = pq - \tau = 0 \tag{6.4}$$

The first two equations resolve to the system

$$\begin{aligned} 2p - \lambda q &= 2a \\ -\lambda p + 2q &= 2b \end{aligned}$$

The determinant of this system is $4 - \lambda^2$, with roots $\lambda = \pm 2$. For $\lambda = 2$ the system implies $a = -b$ and for $\lambda = -2$ the system implies $a = b$. Since we assume $a \neq \pm b$, the determinant is non-zero and we solve to get

$$p = \frac{4a + 2\lambda b}{4 - \lambda^2} \tag{6.5}$$
$$q = \frac{2\lambda a + 4b}{4 - \lambda^2} \tag{6.6}$$

By substituting into Eq. 6.4 we get

$$(4a + 2\lambda b)(2\lambda a + 4b) = m(4 - \lambda^2)^2 \tag{6.7}$$

which is a depressed quartic equation. One of its real solutions yields the optimal p, q, from which the cost ρ can be computed.

7

EXPERIMENTS AND CONCLUSIONS

7.1 EXPERIMENTS

7.1.1 *Using a static reference vector*

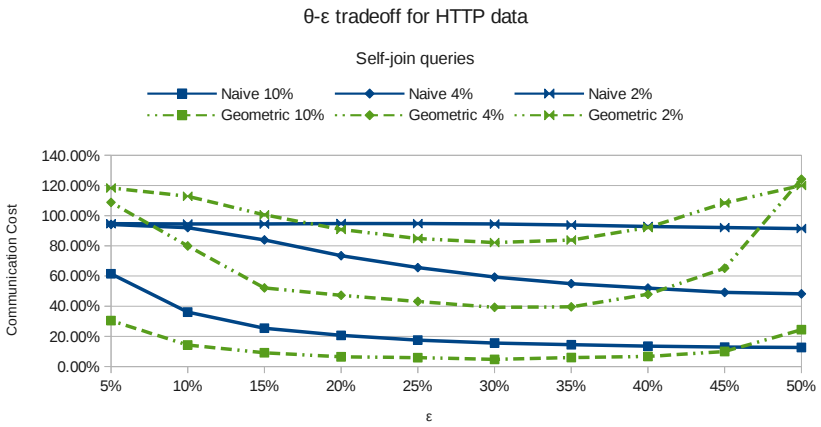
We use similar datasets as the ones presented in [10] for our experiments. The first dataset is available in `http://ita.ee.lbl.gov/html/contrib/WorldCup.html` and contains HTTP requests made to the World Cup '94 web site. The second dataset is available here `http://crawdad.cs.dartmouth.edu/meta.php?name=ibm/watson#N100AD` and contains SNMP records about network users such as number of packets and bytes from/to each user's machine. The data are collected from a corporate research center (IBM Watson research center) over several weeks. From the World Cup '94 dataset, we measured the size attribute and from the Crawdad dataset the shortRet attribute.

The experiments performed evaluated the communication cost required by each method, by affecting the methods parameters. Specifically the sketching error ϵ and the model error θ.

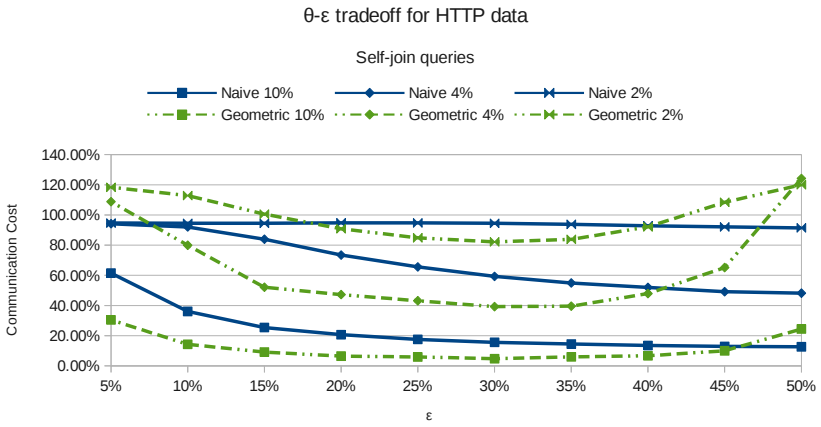
In Figure ((a)), we can see the graph for the World Cup '94 dataset and in Figure ((b)) the graph for the IBM dataset.

We only considered the range 5-50% for ϵ, where the geometric method makes sense. We present the whole range, to explain what is going on in each case, but we can also argue that the geometric method seems to perform better by selecting appropriate parameters which don't have to be guessed at each time.

For the range 5-15%, the geometric method does not perform as good, because of the lack of good exploitation of sketches. Let's see how each method operates in this range. In the Naive method, each site calculates its local sketch and because there are no prediction models, when a violation occurs, it sends to the coordinator only its id (its stream id and its node id) and the local sketch. Furthermore, there is



(a) World Cup (HTTP) dataset



(b) CRAWDAD dataset

Figure 3. $\theta - \epsilon$ tradeoff for self-join queries

Table 2. High threshold τ_h and low threshold τ_l variations

	$\epsilon+\theta=10\%$		$\epsilon+\theta=4\%$		$\epsilon+\theta=2\%$	
$\epsilon/(\epsilon+\theta)$	τ_h	τ_l	τ_h	τ_l	τ_h	τ_l
5%	1.09	0.92	1.04	0.97	1.02	0.98
10%	1.08	0.94	1.03	0.97	1.01	0.99
15%	1.06	0.95	1.02	0.98	1.01	0.99
20%	1.04	0.96	1.02	0.98	1.01	0.99
25%	1.03	0.98	1.01	0.99	1.01	1.00
30%	1.01	0.99	1.00	1.00	1.00	1.00
35%	1.00	1.01	1.00	1.00	1.00	1.00
40%	0.98	1.02	0.99	1.01	1.00	1.00
45%	0.97	1.04	0.99	1.01	0.99	1.01
50%	0.95	1.05	0.98	1.02	0.99	1.01

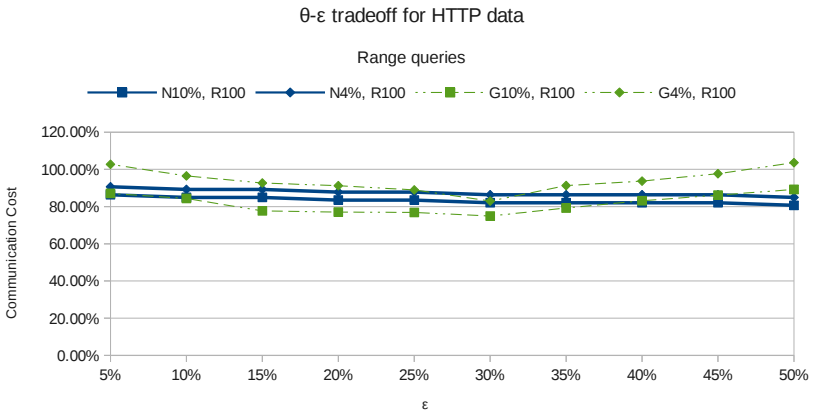
no additional communication from the coordinator. In the geometric method, when a violation occurs, each site sends its sketch, but there is also an additional communication cost from the coordinator which sends the estimate vector back to all the local nodes and additionally the slack vectors used for the balancing process. Bearing also in mind, that for small values of the sketching error ϵ, the sketch size is bigger. The safe zone is also bigger, but due to how the F2-norm increases (quadratically), we have several local violations especially in the start. When this happens, the sites (and the coordinator) prefer to send the difference of the sketch from the last time, instead of sending the sketch. Hence, we have no good exploitation of the sketches.

For the range 20-30%, there is a good balance of the sketch size and the safe zone size, which causes the geometric method to perform better. Local violations don't occur as frequently, and when they do, communication is being made by using the sketches instead of sending the history of updates. Here we can see differences in the communication cost of up to 20%.

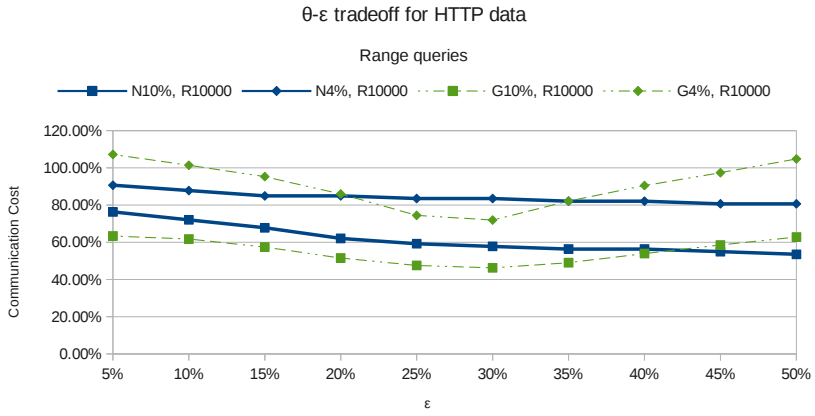
For the range 35-50%, the geometric method starts to lose its performance gains, because of the two thresholds τ_l and τ_h coming close to each other. In Table 2, we present the values of τ_l and τ_h for the different error margins.

The performance on the IBM dataset is similar. We used this dataset to show that the behavior is not tailored to one specific dataset.

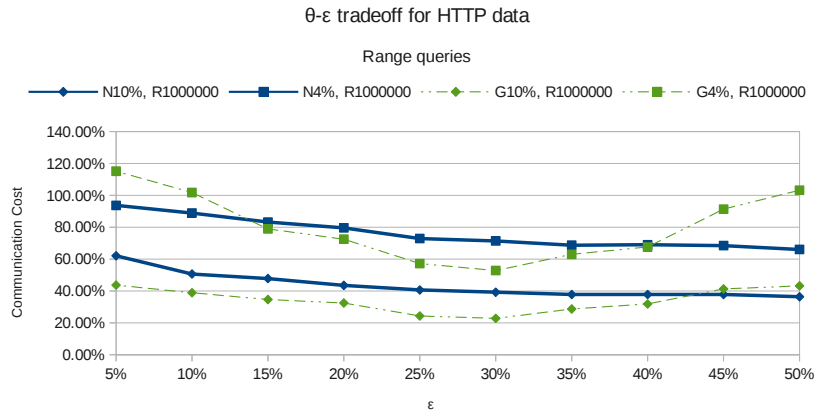
Figure (a), presents the communication cost for a range query [0, 100]. As we can see, for small ranges, the communication cost is higher. The



(a) 0, 100 range



(b) 0, 10000 range



(c) 0, 1000000 range

Figure 4. $\theta - \epsilon$ tradeoff for range queries

local conditions for the naive method, depend on the small number of elements that contribute to the F2 norm and due to the small magnitude of the value, it tends to change more frequently.

Figure (b), presents a range query [0, 10000]. Here the communication cost drops significantly from the [0, 100] case. The benefits of the geometric method are again obvious for the $\epsilon = 30\%$ case.

Figure (c) presents the last range query [0, 1000000]. As the range increases, the behavior starts to resemble that of the inner-join case. The results are comparable to the inner-join scenario.

7.1.2 *Using a dynamic reference vector*

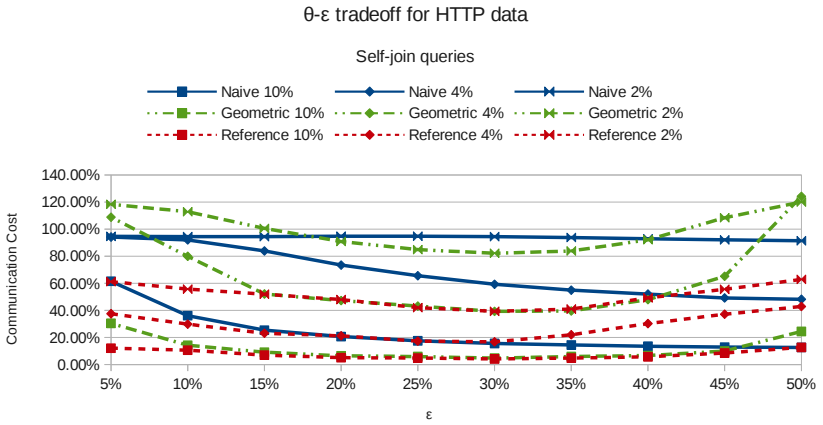
According to [35], a technique for reducing the local violations in each node, is changing the estimate vector that is calculated from the coordinator in a synchronization step to a different vector (a *reference vector*), which has the property to be further from the threshold surface. This tends to give bigger safe zones, hence more space for the drift vectors to move.

The process for determining the new reference vector is the following: In a synchronization phase, we determine the estimate vector $\boldsymbol{e}$ and also $\boldsymbol{e}^*$ which is the vector on the threshold surface that is closest to $\boldsymbol{e}$. Using Algorithm 2, we can obtain the vector $\mathbf{u}$ which is the closest to the threshold surface from the ball center and use it as $\boldsymbol{e}^*$.

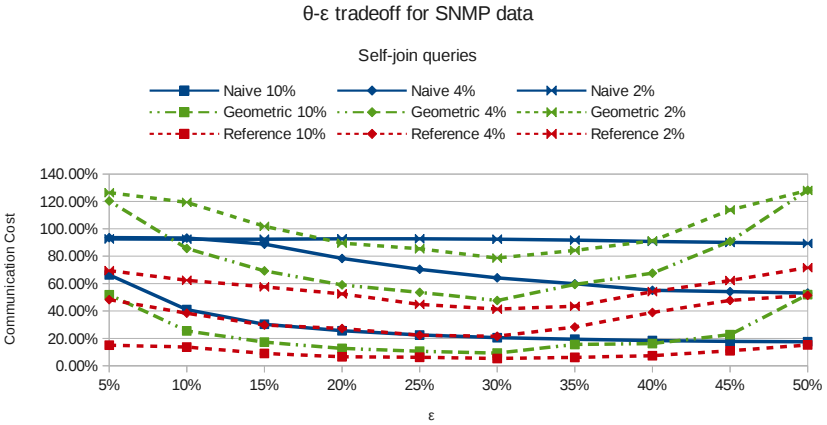
Once $\boldsymbol{e}^*$ has been computed, the coordinator sets the reference vector to be equal to the estimate vector and iteratively examines new reference vectors by doubling the distance of the previous reference vector from $\boldsymbol{e}^*$, i.e. the reference vector $\mathbf{r}_i$ examined in the i-th iteration is $\boldsymbol{e}^* + 2^i(\boldsymbol{e} - \boldsymbol{e}^*)$. In each iteration we calculate the vector on the threshold surface that is closest to $\mathbf{r}_i$. If this vector is $\boldsymbol{e}^*$, we proceed to the next iteration, or else we settle to $\mathbf{r}_i$ from the previous iteration.

In Figure ((a)), we can see the graph for the World Cup '94 dataset and in Figure ((b)) the graph for the IBM dataset containing the new results by using the reference vector $\mathbf{r}_i$.

7.2 CONCLUSIONS



(a) World Cup (HTTP) dataset



(b) CRAWDAD dataset

Figure 5. $\theta - \epsilon$ tradeoff for self-join queries using a reference vector

8

WORST CASE ANALYSIS

The geometric method triggers a communication, when at least one of the balls is not monochromatic. According to Eq.(4.5) there are two thresholds that must be checked $\tau_h = \frac{(1-\epsilon)f(\tilde{\mathbf{v}}_s)}{(1-\theta)(1+\epsilon)}$ and $\tau_l = \frac{(1+\epsilon)f(\tilde{\mathbf{v}}_s)}{(1+\theta)(1-\epsilon)}$.

For our case to be comparable with [10], we consider that each update causes an element of the local stream to increase by +1 and that the global statistics vector is calculated as the sum (and not the average) of the local statistics vectors. That is $\mathbf{v}'_t = \sum_{i=1}^{n} \mathbf{v}_t^{(i)} = n\mathbf{v}_t$ and its sketched versions $\tilde{\mathbf{v}}'_t = \sum_{i=1}^{n} \tilde{\mathbf{v}}_t^{(i)} = n\tilde{\mathbf{v}}_t$ respectively.

As we can see, the new global statistics vector $\tilde{\mathbf{v}}'_t$ can be obtained from the old one scaled by a constant factor n. In the remaining section, we denote $\tilde{\mathbf{v}}_t$ the sketch of the new *scaled* global statistics vector.

Each local vector receives an update that changes one of its coordinates by +1. If $\tilde{\mathbf{v}}_t^{(i)}$ the local vector then an update can be modelled by a vector that has one at one of its coordinates and zero elsewhere:

$$\delta = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}$$

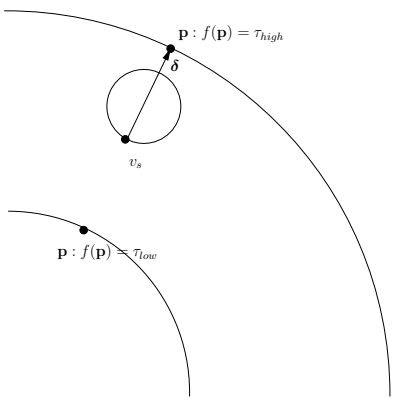


Figure 6. Worst-case communication

The norm of each local statistics vector will change at most by 1:

$$\left\|\tilde{\mathbf{v}}_t^{(i)} + \boldsymbol{\delta}\right\| \leqslant \left\|\tilde{\mathbf{v}}_t^{(i)}\right\| + \|\boldsymbol{\delta}\| = \left\|\tilde{\mathbf{v}}_t^{(i)}\right\| + 1 \tag{8.1}$$

The worst case scenario is for all updates to have a direction towards the threshold surface of τ_h (because we assume only insertions). The worst case is depicted in Fig.6. If $\mathbf{p}$ is the point on the upper threshold surface that is closest to the estimation vector, that is:

$$f(\mathbf{p}) = \frac{(1-\epsilon)f(\tilde{\mathbf{v}}_s)}{(1-\theta)(1+\epsilon)} = \tau_h$$

$$\mathbf{p} = \arg\min_{\mathbf{p}} \|\tilde{\mathbf{v}}_s - \mathbf{p}\|$$

Then, the worst case communication is for each local update to be collinear with the vector that starts at $\tilde{\mathbf{v}}_s$ and ends at $\mathbf{p}$. Let's call this vector $\mathbf{d}$. According to Eq.(8.1), after N updates, the local sketched statistics vectors will change as follows:

$$\tilde{\mathbf{v}}_{s+N}^{(i)} = \tilde{\mathbf{v}}_s^{(i)} + \frac{N n \mathbf{d}}{\|\mathbf{d}\|} \tag{8.2}$$

and so will the global sketched statistics vectors:

$$\tilde{\mathbf{v}}_{s+N} = \sum_{i=1}^{n} \tilde{\mathbf{v}}_{s+N}^{(i)} = \tilde{\mathbf{v}}_s + \frac{N n \mathbf{d}}{\|\mathbf{d}\|}$$

The local ball that each node constructs will change according to Eq.(8.2). If we set $\tau = \frac{1-\epsilon}{(1-\theta)(1+\epsilon)}$, we will have a communication at time t' when:

$$\|\tilde{\mathbf{v}}_{t'}\| = \tau^{1/2} \|\tilde{\mathbf{v}}_s\|$$

and the new upper threshold will become $(\tau^{1/2})^2 \|\tilde{\mathbf{v}}_s\|$. So generally, while:

$$\|\tilde{\mathbf{v}}_t\| \leqslant \tau^{m/2} \|\tilde{\mathbf{v}}_s\|, \quad t > s$$

we have $m-1$ communications.

To have $m-1$ communications after N updates the following condition must be true:

$$\left\| \tilde{\mathbf{v}}_s + Nn\frac{\mathbf{d}}{\|\mathbf{d}\|} \right\| \leqslant \tau^{m/2} \|\tilde{\mathbf{v}}_s\| \tag{8.3}$$

From the triangle inequality:

$$\left\| \tilde{\mathbf{v}}_s + Nn\frac{\mathbf{d}}{\|\mathbf{d}\|} \right\| \leqslant \|\tilde{\mathbf{v}}_s\| + Nn$$

and so Eq.(8.3) becomes:

$$\|\tilde{\mathbf{v}}_s\| + Nn \leqslant \tau^{m/2} \|\tilde{\mathbf{v}}_s\|$$

$$Nn \leqslant \tau^{m/2}(\|\tilde{\mathbf{v}}_s\| - 1)$$

$$m \geqslant 2\log_\tau \frac{Nn}{\|\tilde{\mathbf{v}}_s - 1\|}$$

and changing the logarithm base and replacing back τ:

$$m \geqslant \frac{(1-\theta)(1+\epsilon)\big(\log(Nn) - \log(\|\tilde{\mathbf{v}}_s - 1\|)\big)}{1-\epsilon}$$

So the worst case communication is:

$$O\left(\frac{(1-\theta)(1+\epsilon)\big(\log N + \log n) - \log(\|\tilde{\mathbf{v}}_s - 1\|)\big)}{1-\epsilon}\right) \tag{8.4}$$

The respective result found in [10] is:

$$O\left(\frac{n}{(\epsilon+2\psi)^4}\log\left(\frac{n}{\delta}\right)\log N\right)$$

with $\theta = (\epsilon + 2\psi)$ the total error (from sketch and prediction models), n the number of nodes, δ the confidence probability and N the number of updates.

Comparing the two results, we can see that our worst case communication is dependent logarithmically from the number of nodes n, so theoretically scales better for a very large number of nodes.

Part I

APPENDIX

A

CALCULATING MIN/MAX OF THE MEAN USING LAGRANGE

We want to maximize/minimize the function $f(\tilde{\mathbf{x}}_t) = \frac{1}{n}\sum_{i=1}^{n} x_i^2$ subject to $\sum_{i=1}^{j}(x_i - c_i)^2 = r^2$.

The problem is solved using Lagrange Multipliers. We create the function:

$$\Lambda(\mathbf{x}, \lambda) = \frac{1}{n}\sum_{i=1}^{n} x_i^2 + \lambda\left(\sum_{i=1}^{n}(x_i - c_i)^2 - r^2\right)$$

We then solve the following equations:

$$\frac{\partial \Lambda}{\partial x_i} = 0 \tag{.1}$$

$$\frac{1}{n}2x_i + 2\lambda(x_i - c_i) = 0 \tag{.2}$$

$$x_i + \lambda n(x_i - c_i) = 0 \tag{.3}$$

$$x_i(1 + \lambda n) - \lambda n c_i = 0 \tag{.4}$$

$$x_i = \frac{\lambda n c_i}{1 + \lambda n} \tag{.5}$$

We also have to solve:

$$\frac{\partial \Lambda}{\partial \lambda} = 0 \tag{.6}$$

$$\sum_{i=1}^{n}(x_i - c_i)^2 - r^2 = 0 \tag{.7}$$

Which according to Eq. (.5), becomes:

$$\sum_{i=1}^{n}\left(\frac{\lambda n c_i}{1 + \lambda n} - c_i\right)^2 - r^2 = 0$$

$$\sum_{i=1}^{n}\left(\frac{\lambda n c_i - c_i(1 + \lambda n)}{1 + \lambda n}\right)^2 - r^2 = 0$$

$$\sum_{i=1}^{n}\left(\frac{-c_i}{1 + \lambda n}\right)^2 - r^2 = 0$$

$$\frac{1}{(1 + \lambda n)^2}\sum_{i=1}^{n} c_i^2 = r^2$$

$$(1+\lambda n)^2 = \frac{\sum_{i=1}^{n} c_i^2}{r^2}$$

The quantity $\frac{\sum_{i=1}^{n} c_i^2}{r^2} > 0$, so we solve for λ:

$$\pm(1+\lambda n) = \frac{\sqrt{\sum_{i=1}^{n} c_i^2}}{r}$$

So we have the following 2 solutions for λ:

$$\begin{cases} 1+\lambda n = \frac{\sqrt{\sum c_i^2}}{r} \\ -1-\lambda n = \frac{\sqrt{\sum c_i^2}}{r} \end{cases} = \begin{cases} \lambda = \frac{\sqrt{\sum c_i^2}-r}{rn} \\ \lambda = -\frac{\sqrt{\sum c_i^2}+r}{rn} \end{cases}$$

Now we can solve for x_i:

$$x_i = \frac{(\sqrt{\sum c_i^2} \pm r)c_i}{\sqrt{\sum c_i^2}}$$

which represent the minimum/maximum points.

BIBLIOGRAPHY

[1] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96*, pages 20–29, 1996. doi: 10.1145/237814.237823. URL http://portal.acm.org/citation.cfm?doid=237814.237823. (Cited on pages 3, 5, 8, and 9.)

[2] Noga Alon, Phillip B. Gibbons, Yossi Matias, and Mario Szegedy. Tracking join and self-join sizes in limited storage. *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems - PODS '99*, (March):10–20, 1999. doi: 10.1145/303976.303978. URL http://portal.acm.org/citation.cfm?doid=303976.303978. (Cited on pages 3, 5, 8, 9, and 10.)

[3] Arvind Arasu and Gurmeet Singh Manku. Approximate counts and quantiles over sliding windows. *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems - PODS '04*, page 286, 2004. doi: 10.1145/1055558.1055598. URL http://portal.acm.org/citation.cfm?doid=1055558.1055598. (Cited on page 5.)

[4] Brian Babcock and Chris Olston. Distributed top-k monitoring. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 28–39, New York, New York, USA, 2003. ACM. ISBN 158113634X. doi: 10.1145/872763.872764. URL http://portal.acm.org/citation.cfm?doid=872757.872764http://portal.acm.org/citation.cfm?id=872764. (Cited on page 5.)

[5] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems - PODS '02*, page 1, 2002. doi: 10.1145/543614.543615. URL http://portal.acm.org/citation.cfm?doid=543613.543615. (Cited on page 2.)

[6] M Charikar. Finding frequent items in data streams. *Theoretical Computer Science*, 312(1):3–15, January 2004. ISSN 03043975. doi: 10.1016/S0304-3975(03)00400-6. URL http://linkinghub.elsevier.com/retrieve/pii/S0304397503004006. (Cited on page 5.)

[7] D. Chu, A. Deshpande, J.M. Hellerstein, and W. Hong. Approximate data collection in sensor networks using probabilistic models. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pages 48–48. IEEE, 2006. ISBN 0-7695-2570-9. doi: 10.1109/ICDE.2006.21. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1617416http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1617416. (Cited on page 5.)

[8] G Cormode and S Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, April 2005. ISSN 01966774. doi: 10.1016/j.jalgor.2003.12.001. URL http://linkinghub.elsevier.com/retrieve/pii/S0196677403001913. (Cited on pages 3 and 8.)

[9] Graham Cormode and Minos Garofalakis. Sketching Streams Through the Net : Distributed Approximate Query Tracking. *31st VLDB Conference*, pages 13–24, 2005. (Cited on page 7.)

[10] Graham Cormode and Minos Garofalakis. Approximate continuous querying over distributed streams. *ACM Trans. Database Syst.*, 33(2):1–39, 2008. ISSN 0362-5915. doi: http://doi.acm.org/10.1145/1366102.1366106. (Cited on pages 2, 3, 10, 11, 31, 37, and 39.)

[11] Graham Cormode and S. Muthukrishnan. What's hot and what's not: tracking most frequent items dynamically. *ACM Transactions on Database Systems*, 30(1):249–278, March 2005. ISSN 03625915. doi: 10.1145/1061318.1061325. URL http://portal.acm.org/citation.cfm?doid=1061318.1061325. (Cited on page 5.)

[12] Graham Cormode, Minos Garofalakis, S Muthukrishnan, and Rajeev Rastogi. Holistic aggregates in a networked world: distributed tracking of approximate quantiles. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 25–36, New York, NY, USA, 2005. ACM. ISBN 1-59593-

060-4. doi: http://doi.acm.org/10.1145/1066157.1066161. (Cited on page 5.)

[13] Graham Cormode, S. Muthukrishnan, and K. Yi. Algorithms for distributed functional monitoring. *ACM Transactions on Algorithms (TALG)*, 7(2):21, 2011. URL http://portal.acm.org/citation.cfm?id=1921667. (Cited on page 3.)

[14] Abhinandan Das, S. Ganguly, M. Garofalakis, and R. Rastogi. Distributed set-expression cardinality estimation. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 312–323. VLDB Endowment, 2004. URL http://portal.acm.org/citation.cfm?id=1316718. (Cited on page 5.)

[15] Amol Deshpande, Carlos Guestrin, S.R. Madden, J.M. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 588–599. VLDB Endowment, 2004. URL http://portal.acm.org/citation.cfm?id=1316741. (Cited on pages 1 and 5.)

[16] Alin Dobra, Minos Garofalakis, Johannes Gehrke, and Rajeev Rastogi. Processing complex aggregate queries over data streams. *Proceedings of the 2002 ACM SIGMOD international conference on Management of data - SIGMOD '02*, page 61, 2002. doi: 10.1145/564696.564699. URL http://portal.acm.org/citation.cfm?doid=564691.564699. (Cited on pages 5, 8, and 10.)

[17] Sumit Ganguly and Minos Garofalakis. Processing set expressions over continuous update streams. *Proceedings of the 2003*, page 265, 2003. doi: 10.1145/872788.872790. URL http://portal.acm.org/citation.cfm?doid=872757.872790http://portal.acm.org/citation.cfm?id=872757.872790. (Cited on page 5.)

[18] Sumit Ganguly, Minos Garofalakis, and Rajeev Rastogi. Processing Data-Stream Join Aggregates Using Skimmed Sketches. *In Proc. Int. Conf. on Extending Database Technology (EDBT)*, 2004. (Cited on page 7.)

[19] Sumit Ganguly, Deepanjan Kesh, and Chandan Saha. Practical Algorithms for Tracking Database Join Sizes. *FSTTCS*, 2005. (Cited on page 7.)

[20] PB Gibbons. Distinct sampling for highly-accurate answers to distinct values queries and event reports. *Proceedings of the International Conference on Very*, 2001. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.80.2580&rep=rep1&type=pdf. (Cited on page 5.)

[21] AC Gilbert, Yannis Kotidis, and S Muthukrishnan. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. *Proceedings of the*, 2001. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.119.5031&rep=rep1&type=pdf. (Cited on page 5.)

[22] Michael Greenwald and Sanjeev Khanna. Space-efficient online computation of quantile summaries. *ACM SIGMOD Record*, 30(2):58–66, June 2001. ISSN 01635808. doi: 10.1145/376284.375670. URL http://portal.acm.org/citation.cfm?doid=376284.375670. (Cited on page 5.)

[23] J Haas and M Hellerstein. Ripple Joins for Online Aggregation. *SIGMOD*, pages 287–298, 1999. (Cited on page 7.)

[24] L. Huang, X. Nguyen, M. Garofalakis, J. M. Hellerstein, M. I. Jordan, a. D. Joseph, and N. Taft. Communication-Efficient Online Detection of Network-Wide Anomalies. *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pages 134–142, 2007. doi: 10.1109/INFCOM.2007.24. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4215606. (Cited on page 6.)

[25] Ling Huang, Minos Garofalakis, A.D. Joseph, and Nina Taft. Communication-efficient tracking of distributed cumulative triggers. *27th International Conference on Distributed Computing Systems (ICDCS '07)*, pages 54–54, 2007. doi: 10.1109/ICDCS.2007.93. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4268207http://www.computer.org/portal/web/csdl/doi/10.1109/ICDCS.2007.93. (Cited on page 6.)

[26] Ankur Jain, E.Y. Chang, and Y.F. Wang. Adaptive stream resource management using kalman filters. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 11–22, New York, New York, USA, 2004. ACM. ISBN 1581138598. doi: 10.1145/1007568.1007573. URL http://portal.acm.org/citation.cfm?doid=1007568.1007573http://portal.acm.org/citation.cfm?id=1007573. (Cited on page 5.)

[27] D. Kempe, a. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 482–491. doi: 10.1109/SFCS.2003.1238221. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1238221. (Cited on page 7.)

[28] Ram Keralapura, Graham Cormode, and Jeyashankher Ramamirtham. Communication-efficient distributed monitoring of thresholded counts. *Proceedings of the 2006 ACM SIGMOD international conference on Management of data - SIGMOD '06*, page 289, 2006. doi: 10.1145/1142473.1142507. URL http://portal.acm.org/citation.cfm?doid=1142473.1142507. (Cited on page 6.)

[29] S. Madden and M.J. Franklin. Fjording the stream: an architecture for queries over streaming sensor data. *Proceedings 18th International Conference on Data Engineering*, pages 555–566, 2001. doi: 10.1109/ICDE.2002.994774. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=994774. (Cited on page 2.)

[30] a. Manjhi, V. Shkapenyuk, K. Dhamdhere, and C. Olston. Finding (Recently) Frequent Items in Distributed Data Streams. *21st International Conference on Data Engineering (ICDE'05)*, pages 767–778, 2004. doi: 10.1109/ICDE.2005.68. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1410191. (Cited on page 5.)

[31] G.S. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 346–357. VLDB Endowment, 2002.

URL http://portal.acm.org/citation.cfm?id=1287400. (Cited on page 5.)

[32] Chris Olston, Jing Jiang, and Jennifer Widom. Adaptive filters for continuous queries over distributed data streams. *Proceedings of the 2003 ACM SIGMOD international conference on on Management of data - SIGMOD '03*, page 563, 2003. doi: 10.1145/872824.872825. URL http://portal.acm.org/citation.cfm?doid=872757.872825. (Cited on page 5.)

[33] Florin Rusu and Alin Dobra. Sketches for size of join estimation. *ACM Transactions on Database Systems (TODS)*, 33(3):1–46, 2008. URL http://portal.acm.org/citation.cfm?id=1386121. (Cited on page 20.)

[34] Izchak Sharfman, Assaf Schuster, and Daniel Keren. A geometric approach to monitoring threshold functions over distributed data streams. *ACM Trans. Database Syst.*, 32(4):23, 2007. ISSN 0362-5915. doi: http://doi.acm.org/10.1145/1292609.1292613. (Cited on pages 2, 12, and 13.)

[35] Izchak Sharfman, Assaf Schuster, and Daniel Keren. Shape sensitive geometric monitoring. *Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems - PODS '08*, page 301, 2008. doi: 10.1145/1376916.1376958. URL http://portal.acm.org/citation.cfm?doid=1376916.1376958. (Cited on page 35.)

[36] Y. Zhu and D. Shasha. StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time. *Proceeding of the 28th VLDB Conference, Hong Kong, China*, 2002. URL http://www.mendeley.com/research/no-title-avail/. (Cited on page 2.)