# Sketch-based Geometric Monitoring of Distributed Stream Queries

Minos Garofalakis
Technical University of Crete
minos@softnet.tuc.gr

Daniel Keren
University of Haifa
dkeren@cs.haifa.ac.il

Vasilis Samoladas
Technical University of Crete
vsam@softnet.tuc.gr

## 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 technical insights for the effective use of the geometric method lies in exploiting a much lower-dimensional 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.

## 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 [12]), or power and bandwidth restrictions (e.g., in wireless sensornets, where communication is the key determinant of sensor battery life [26]). 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 correlating data from different tables 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 [7]. Monitoring the precise value of such holistic queries without continuously centralizing all the data seems hopeless; luckily, when tracking statistical behavior and patters in large scale systems, *approximate answers* (with reasonable approximation error guarantees) are typically sufficient. This often allows algorithms to effectively tradeoff efficiency with approximation quality (e.g., using sketch-based stream approximations [7]).

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 [7, 31]. 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) [31].

**Prior Work.** The bulk of work on data-stream processing has focused on developing space-efficient, one-pass algorithms for performing a wide range of *centralized, one-shot computations* on massive data streams; examples include computing quantiles [21], estimating distinct values [18] and set-expression cardinalities [16], counting frequent elements (i.e., "heavy hitters") [4, 10, 28], approximating large Haar-wavelet coefficients [20], and estimating join sizes and stream norms [1, 2, 15]. As already mentioned, all the above methods work in a centralized, one-shot setting and, therefore, do not consider communication-efficiency issues. Other work has proposed methods that carefully optimize site communication costs for approximating different queries in a distributed setting, including quantiles [22] and heavy hitters [27]; however, the underlying assumption is that the computation is triggered either periodically or in response to a one-shot request. Such techniques are not immediately applicable for *continuous-monitoring*, where the goal is to continuously provide real-time, guaranteed-quality estimates over a distributed collection of streams. Morphing such one-shot solutions to continuous problems entails propagating each change and recomputing the solutions which is communication inefficient, or involves periodic updates and other heuristics that can no longer provide real-time estimation guarantees.

Monitoring distributed data streams has attracted substantial research interest in recent years [6, 29]. Early work has looked at the monitoring of *single values*, and building appropriate models and filters to avoid propagating updates if these are insignificant compared to the value of a simple aggregate (e.g., to the SUM of the distributed values). [30] proposes a scheme based on "adaptive filters" — that is, bounds around the value of distributed variables, which shrink or grow in response to relative stability or variability, while ensuring that the total uncertainty in the bounds is at most a user-specified bound. [23] proposes building a Kalman Filter for individual values, and only propagating an update in a value if it falls more than $\delta$ away from the predicted value. The BBQ system [14] builds a dynamic, multi-dimensional probabilistic model of a set of distributed sensor readings to drive acquisitional query processing; this was later extended to the continuous case in the Ken system [5]. A common aspect of all these earlier works is that they typically consider only a small number of monitored values per site, and assume that it is feasible to locally monitor and/or build a model for each such value. In contrast, our problem setup is much more complex, as each resource-limited site monitors a *streaming distribution of a large number of values* and cannot afford to explicitly capture or model each value separately.

Closest in spirit to our work are the results of [3] and [13], as well as our work on tracking distributed quantiles [8] and join aggregates [7]. All these efforts explicitly consider the tradeoff between accuracy and communication for monitoring a class of continuous queries over distributed streams. With the exception of [7], these earlier papers focus solely on a narrow class of distributed-monitoring queries (e.g., top-$k$ values or one-dimensional quantiles), resulting in special-purpose solutions applicable only to the specific form of queries at hand. More recently, [25, 31] have proposed an approach for efficiently monitoring the value of a *general function/query* over distributed data relative to a given threshold. Their solution relies on interesting geometric arguments for breaking up a global threshold condition on a function into "safe" local conditions that can be checked locally at each site. Still, [25, 31] focus on monitoring a *distributed trigger* condition rather than a distributed query result with approximation-error guarantees; perhaps more importantly, they assume that the full state of the stream can be maintained at both the remote sites and the coordinator. [7] considers monitoring the same class of sketch-based query estimates as we do. Their proposed approach is again purpose-built for the specific type of queries; furthermore, as our experimental results show, the effective combination of the generic geometric monitoring method of [25, 31] and sketch-based query estimates (as proposed in this paper) can give significant performance benefits over the approach in [7].[1]

**Our Contributions.** In this paper, we propose novel algorithmic techniques for efficiently tracking sketch-based approximations for a broad class of complex aggregate queries over massive, distributed data streams. Our tracking protocols are based on a novel combination of the geometric method of Sharfman et al. [25, 31] for monitoring general threshold conditions over distributed streams and AMS sketch estimators for querying massive streaming data [1, 2, 15]. The effective incorporation of sketching techniques significantly expands the scope of the original geometric method, allowing it to efficiently track a broad class of complex queries over massive, high-dimensional distributed data streams with provable error guarantees. More specifically, we focus on the class of stream queries supported by AMS sketching tools, including general inner products (i.e., join aggregates), as well as the special cases of $L_2$-norms (i.e., self-join sizes) and range aggregates (e.g., for tracking quantiles, histograms, wavelets, and heavy-hitters over the streams) [7]. One of our key technical insights is that, by exploiting properties of AMS sketches, our algorithms can perform highly-efficient geometric monitoring in a *much lower-dimensional space*. Another major technical challenge lies in effectively dealing with the highly non-linear median operator (that is required for estimation over AMS sketches) in the context of geometric function monitoring. We propose novel geometric algorithms for tracking medians computed over AMS sketches of the streams for different types of distributed stream queries of high practical interest. Our experimental study with real-life data sets demonstrates the practical benefits of our approach, showing consistent gains of up to 35% in terms of total communication cost compared to the current state-of-the-art method [7]; furthermore, our techniques demonstrate even more impressive benefits (of over 100%) when focusing on the communication costs of data (i.e., sketch) shipping in the system.

**Roadmap.** The remainder of this paper is organized as follows. Section 2 discusses background material on distributed streaming, sketches, and the geometric method. In Section 3, we present our novel geometric monitoring schemes for sketch-based approximate query tracking. Section 4 presents the results of our experimental study. Finally, we conclude the paper and discuss future directions in Section 5.

## 2. PRELIMINARIES AND PROBLEM SETUP

**System Architecture.** We consider a distributed-computing environment, comprising a collection of $k$ *remote sites* and a designated

[1]Note that [7] also proposes the idea of using *prediction models* for local data streams, which is orthogonal to the work presented in this paper. In fact, the application of prediction models within the geometric monitoring method has already been explored in a recent paper [17].

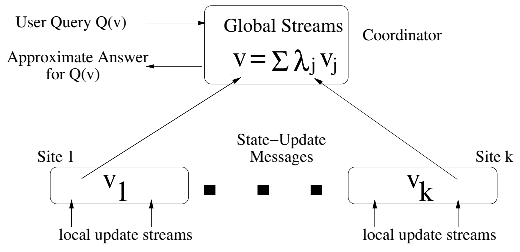


Figure 1: Distributed stream processing architecture.

*coordinator site*. Streams of data updates arrive continuously at remote sites, while the coordinator site is responsible for generating approximate answers to (possibly, continuous) user queries posed over the *unions* of remotely-observed streams (across all sites). Following earlier work in the area [3, 7, 8, 13, 30], our distributed stream-processing model does not allow direct communication between remote sites; instead, as illustrated in Figure 1, a remote site exchanges messages only with the coordinator, providing it with state information on its (locally-observed) streams. Note that such a hierarchical processing model is, in fact, representative of a large class of applications, including network monitoring where a central Network Operations Center (NOC) is responsible for processing network traffic statistics (e.g., link bandwidth utilization, IP source-destination byte counts) collected at switches, routers, and/or Element Management Systems (EMSs) distributed across the network.

Each remote site $j \in \{1, \ldots, k\}$ observes (possibly, several) local update streams that incrementally render a *local statistics vector* $\boldsymbol{v}_j$ capturing the current local state of the observed stream(s) at site $j$. As an example, in the case of IP routers monitoring the number of `TCP` and `UDP` packets exchanged between source and destination IP addresses, the local statistics vector $\boldsymbol{v}_j$ has $2 \times 2^{64}$ entries capturing the up-to-date frequencies for specific (source, destination) pairs observed in `TCP` and `UDP` packets routed through router $j$. (For instance, the first (last) $2^{64}$ entries of $\boldsymbol{v}_j$ could be used for `TCP` (respectively, `UDP`) packet frequencies.) All local statistics vectors $\boldsymbol{v}_j$ in our distributed streaming architecture change dynamically over time — when necessary, we make this dependence explicit, using $\boldsymbol{v}_j(t)$ to denote the state of the vector at time $t$ (assuming a consistent notion of "global time" in our distributed system). The unqualified notation $\boldsymbol{v}_j$ typically refers to the *current* state of the local statistics vector.

We define the *global statistics vector* $\boldsymbol{v}$ of our distributed stream(s) as any *weighted average* (i.e., convex combination) of the local statistics vectors $\{\boldsymbol{v}_j\}$; that is, $\boldsymbol{v} = \sum_{j=1}^{k} \lambda_j \boldsymbol{v}_j$, where $\sum_j \lambda_j = 1$ and $\lambda_j \geq 0$ for all $j$. (Again, to simplify notation, we typically omit the explicit dependence on time when referring to the current global vector.) Our focus is on the problem of effectively answering user queries (or, functions) over the global statistics vector at the coordinator site. Rather than one-time query/function evaluation, we assume a continuous-querying environment which implies that the coordinator needs to *continuously maintain* (or, *track*) the answers to queries as the local update streams $\boldsymbol{v}_j$ evolve at individual remote sites. There are two defining characteristics of our problem setup that raise difficult algorithmic challenges for our query tracking problems:

• *The distributed nature and large volumes of local streaming data* imply important communication and space/time efficiency concerns. Naïve schemes that accurately track query answers by forcing remote sites to ship every remote stream update to the coordinator are clearly impractical, since they can impose an inordinate burden on the underlying communication infrastructure (especially, for high-rate data streams and large numbers of remote sites). Furthermore, the voluminous nature of the local data streams implies that effective streaming tools are needed at the remote sites in order to manage the streaming local statistics vectors in sublinear space/time. A main part of our approach is to adopt the paradigm of continuous tracking of *approximate* query answers at the coordinator site with strong guarantees on the quality of the approximation. This allows our schemes to effectively trade-off space/time/communication efficiency and query-approximation accuracy in a precise, quantitative manner.

• *General, non-linear queries/functions* imply fundamental and difficult challenges for distributed monitoring. For the case of linear functions, a number of approaches have been proposed that rely on the key idea of allocating appropriate *"slacks"* to the remote sites based on their locally-observed function values (e.g., [3, 24, 30]). Unfortunately, it is not difficult to find examples of simple *non-linear* functions on one-dimensional data, where it is basically impossible to make any assumptions about the value of the global function based on the function values observed locally at the sites [31]. This renders conventional slack-allocation schemes inapplicable in our setting.

As a concrete example of complex function tracking, consider the aforementioned global vector $\boldsymbol{v} = \langle \boldsymbol{t}, \boldsymbol{u} \rangle$ of `TCP` and `UDP` packet frequencies observed over a collection of IP routers. where $\boldsymbol{t}$, $\boldsymbol{u}$ are the subvectors of $\boldsymbol{v}$ corresponding to `TCP` and `UDP` traffic, respectively. Tracking the (non-linear) inner-product function $f(\boldsymbol{v}) = \boldsymbol{t} \cdot \boldsymbol{u} = \sum_i \boldsymbol{t}[i]\boldsymbol{u}[i]$ (i.e., the size of the join of the two traffic distributions over (source, destination)) can allow the NOC to effectively monitor the strength of the correlation across the two types of traffic in the underlying set of routers. Clearly, simple slack-allocation techniques [3, 24, 30] cannot be applied here.

**AMS Stream 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 centralized settings [1, 2, 11, 15, 20]. The key idea in such sketching techniques is to represent a streaming frequency vector $\boldsymbol{v}$ using a much smaller (typically, randomized) *sketch* vector (denoted by $\texttt{sk}(\boldsymbol{v})$) that (1) can be easily maintained as the updates incrementally rendering $\boldsymbol{v}$ are streaming by, and (2) provide probabilistic guarantees for the quality of the data approximation. The widely used AMS sketch (proposed by Alon, Matias, and Szegedy in their seminal paper [2]) defines $i^{th}$ sketch entry $\texttt{sk}(\boldsymbol{v})[i]$ as the random variable $\sum_k \boldsymbol{v}[k] \cdot \xi_i[k]$, where $\{\xi_i\}$ 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 (logarithmic) space [2]. Note that, by construction, each entry of $\texttt{sk}(\boldsymbol{v})$ is essentially a *randomized linear projection* (i.e., an inner product) of the $\boldsymbol{v}$ vector (using the corresponding $\xi$ family), that can be easily maintained (using a simple counter) over the input update stream. Another important property is the *linearity* of AMS sketches: Given two "parallel" sketches (built using the same $\xi$ families) $\texttt{sk}(\boldsymbol{v}_1)$ and $\texttt{sk}(\boldsymbol{v}_2)$, the sketch of the union of the two underlying streams (i.e., the streaming vector $\boldsymbol{v}_1 + \boldsymbol{v}_2$) is simply the component-wise sum of their sketches; that is, $\texttt{sk}(\boldsymbol{v}_1 + \boldsymbol{v}_2) = \texttt{sk}(\boldsymbol{v}_1) + \texttt{sk}(\boldsymbol{v}_2)$. This linearity makes such sketches particularly useful in *distributed* streaming settings [7].

The following theorem summarizes some of the basic estimation properties of AMS sketches for (centralized) stream query processing. (Throughout, the notation $x \in (y \pm z)$ is equivalent to $|x - y| \leq |z|$.) We use $f_{\text{AMS}}()$ to denote the standard *AMS estimator function*, involving both averaging and median-selection operations over the components of the sketch-vector inner product [1, 2]. Formally, each sketch vector can be conceptually viewed as a two-dimensional $n \times m$ array, where $n = O(\frac{1}{\epsilon^2})$, $m = O(\log(1/\delta))$ and $\epsilon$, $1-\delta$ denote the desired bounds on error and probabilistic confidence (respectively), and the AMS estimator function is defined as:

$$f_{\text{AMS}}(\texttt{sk}(\boldsymbol{v}), \texttt{sk}(\boldsymbol{u})) = \underset{i=1,\ldots,m}{\text{median}}\{\frac{1}{n}\sum_{l=1}^{n} \texttt{sk}(\boldsymbol{v})[l,i] \cdot \texttt{sk}(\boldsymbol{u})[l,i]\}.$$

THEOREM 2.1 ([1, 2]). *Let* $\texttt{sk}(\boldsymbol{v})$ *and* $\texttt{sk}(\boldsymbol{u})$ *denote two parallel sketches comprising* $O(\frac{1}{\epsilon^2}\log(1/\delta))$ *counters, built over the streams* $\boldsymbol{v}$ *and* $\boldsymbol{u}$*. Then, with probability at least* $1-\delta$, $f_{AMS}(\texttt{sk}(\boldsymbol{v}), \texttt{sk}(\boldsymbol{u})) \in (\boldsymbol{v}\cdot\boldsymbol{u} \pm \epsilon\|\boldsymbol{v}\|\|\boldsymbol{u}\|)$*. The processing time required to maintain each sketch is* $O(\frac{1}{\epsilon^2}\log(1/\delta))$ *per update.*

Thus, AMS sketch estimators can effectively approximate *inner-product queries* $\boldsymbol{v}\cdot\boldsymbol{u} = \sum_i \boldsymbol{v}[i]\cdot\boldsymbol{u}[i]$ over streaming data vectors and tensors. Such inner products naturally map to *join and multi-join aggregates* when the the vectors/tensors capture the frequency distribution of the underlying join attribute(s) [15]. Furthermore, they can capture several other interesting query classes, including range and quantile queries [19], heavy hitters and top-$k$ queries [4], and approximate histogram and wavelet representations [9, 20, 32]. An interesting special case is that of the (squared) $L_2$ *norm* (or, *self-join*) query (i.e., $\boldsymbol{u} = \boldsymbol{v}$): Theorem 2.1 implies that the AMS estimator $f_{\text{AMS}}(\texttt{sk}(\boldsymbol{v}), \texttt{sk}(\boldsymbol{v}))$ (or, simply $f_{\text{AMS}}(\texttt{sk}(\boldsymbol{v}))$) is within $\epsilon$ relative error of the true squared $L_2$ norm $\|\boldsymbol{v}\|^2 = \sum_k(\boldsymbol{v}[k])^2$; that is, $f_{\text{AMS}}(\texttt{sk}(\boldsymbol{v})) \in (1\pm\epsilon)\|\boldsymbol{v}\|^2$. To provide $\epsilon$ relative-error guarantees for the general inner-product query $\boldsymbol{v}\cdot\boldsymbol{u}$, Theorem 2.1 can be applied with error bound $\epsilon' = \epsilon(\boldsymbol{v}\cdot\boldsymbol{u})/(\|\boldsymbol{v}\|\|\boldsymbol{u}\|)$, giving a total sketching space requirement of $O(\frac{\|\boldsymbol{v}\|^2\|\boldsymbol{u}\|^2}{\epsilon^2(\boldsymbol{v}\cdot\boldsymbol{u})^2}\log(1/\delta))$ counters [1].

A drawback of AMS sketches is that every streaming update must "touch" every component of the sketch vector (to update the corresponding randomized linear projection). This can be problematic for massive, rapid-rate data streams, especially when a tight error guarantee $\epsilon$ is required. The *Fast-AMS sketch* [7] solves this problem by guaranteeing *logarithmic-time* (i.e., $O(\log(1/\delta))$) sketch-update costs, while offering the same space/accuracy tradeoff as the basic AMS sketch (through a more careful analysis) [7]. (Our implementation in Section 4 employs the Fast-AMS variant.)

**The Geometric Method.** Sharfman et al. [31] consider the fundamental problem of *distributed threshold monitoring*; that is, determine whether $f(\boldsymbol{v}) < \tau$ or $f(\boldsymbol{v}) > \tau$, for a given (general) function $f()$ over the global statistics vector and a fixed threshold $\tau$. Their key idea is that, since it is generally impossible to connect the locally-observed values of $f()$ to the global value $f(\boldsymbol{v})$, one can employ geometric arguments to monitor the *domain* (rather than the range) of the monitored function $f()$. More specifically, assume that at any point in time, each site $j$ has informed the coordinator of some prior state of its local vector $\boldsymbol{v}_j^p$; thus, the coordinator has an estimated global vector $\boldsymbol{e} = \boldsymbol{v}^p = \sum_{j=1}^k \lambda_j \boldsymbol{v}_j^p$. Clearly, the updates arriving at sites can cause the local vectors $\boldsymbol{v}_j$ to drift too far from their previously reported values $\boldsymbol{v}_j^p$, possibly leading to a violation of the $\tau$ threshold. Let $\Delta\boldsymbol{v}_j = \boldsymbol{v}_j - \boldsymbol{v}_j^p$ denote the local *delta vector* (due to updates) at site $j$, and let $\boldsymbol{u}_j = \boldsymbol{e} + \Delta\boldsymbol{v}_j$ be the *drift vector* from the previously reported estimate at site $j$. We can then express the current global statistics vector $\boldsymbol{v}$ in terms of the drift vectors:

$$\boldsymbol{v} = \sum_{j=1}^{k}\lambda_j(\boldsymbol{v}_j^p + \Delta\boldsymbol{v}_j) = \boldsymbol{e} + \sum_{j=1}^{k}\lambda_j\Delta\boldsymbol{v}_j = \sum_{j=1}^{k}\lambda_j(\boldsymbol{e}+\Delta\boldsymbol{v}_j).$$

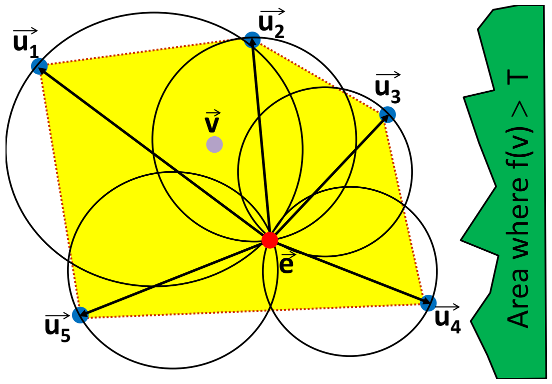


Figure 2: Estimate vector $\boldsymbol{e}$, delta vectors $\Delta\boldsymbol{v}_j$ (arrows out of $\boldsymbol{e}$), convex hull enclosing the current global vector $\boldsymbol{v}$ (dotted outline), and bounding balls $B(\boldsymbol{e}+\frac{1}{2}\Delta\boldsymbol{v}_j, \frac{1}{2}\|\Delta\boldsymbol{v}_j\|)$.

That is, the current global vector is a convex combination of drift vectors and, thus, guaranteed to lie somewhere within the convex hull of the delta vectors around $\boldsymbol{e}$. Figure 2 depicts an example in $d=2$ dimensions. The current value of the global statistics vector lies somewhere within the shaded convex-hull region; thus, as long as the convex hull does not overlap the inadmissible region (i.e., the region $\{\boldsymbol{v}\in\mathbb{R}^2 : f(\boldsymbol{v}) > \tau\}$ in Figure 2), we can guarantee that the threshold has not been violated (i.e., $f(\boldsymbol{v}) \le \tau$)).

The problem, of course, is that the $\Delta\boldsymbol{v}_j$'s are spread across the sites and, thus, the above condition cannot be checked locally. To transform the global condition into a local constraint, we place a $d$-dimensional *bounding ball* $B(\boldsymbol{c}, r)$ around each local delta vector, of radius $r = \frac{1}{2}\|\Delta\boldsymbol{v}_j\|$ and centered at $\boldsymbol{c} = \boldsymbol{e} + \frac{1}{2}\Delta\boldsymbol{v}_j$ (see Figure 2). It can be shown that (in any dimensionality $d$) the union of all these balls completely covers the convex hull of the drift vectors [31]. This observation effectively reduces the problem of monitoring the global statistics vector to the local problem of each remote site monitoring the ball around its local delta vector.

More specifically, given the monitored function $f()$ and threshold $\tau$, we can partition the $d$-dimensional space into two sets $V = \{\boldsymbol{v} : f(\boldsymbol{v}) > \tau\}$ and $\overline{V} = \{\boldsymbol{v} : f(\boldsymbol{v}) \le \tau\}$. (Note that these sets can be arbitrarily complex, e.g., they may comprise multiple disjoint regions of $\mathbb{R}^d$.) The basic protocol is now quite simple: Each site monitors its delta vector $\Delta\boldsymbol{v}_j$ and, with each update, checks whether its bounding ball $B(\boldsymbol{e}+\frac{1}{2}\Delta\boldsymbol{v}_j, \frac{1}{2}\|\Delta\boldsymbol{v}_j\|)$ is *monochromatic*, i.e., all points in the ball lie within the same region ($V$ or $\overline{V}$). If this is not the case, we have a *local threshold violation*, and the site communicates its local $\Delta\boldsymbol{v}_j$ to the coordinator. The coordinator then initiates a *synchronization process* that typically tries to resolve the local violation by communicating with only a subset of the sites in order to "balance out" the violating $\Delta\boldsymbol{v}_j$ and ensure the monochromicity of all local bounding balls [31]. In the worst case, the delta vectors from all $k$ sites are collected, leading to an accurate estimate of the current global statistics vector, which is by definition monochromatic (since all bounding balls have 0 radius).

In more recent work, Sharfman et al. [25] demonstrate that their geometric monitoring method can employ properties of the function and the data to guide the choice of a global *reference point* and local *bounding ellipsoids* for defining the local constraints. Furthermore, they show that the local bounding balls/ellipsoids defined by the geometric method are actually special cases of a more general theory of *Safe Zones (SZs)*, which can be broadly defined as *convex subsets of the admissible region* of a threshold query. It is not difficult to see that, as long as the local drift vectors stay within such a SZ, the global vector is guaranteed (by convexity) to be within the admissible region of the query [25].

## 3. SKETCH-BASED APPROXIMATE GEOMETRIC MONITORING

In this section, we develop our approach for geometric monitoring of non-linear, inner-product queries using AMS sketches. The sketching idea offers an effective streaming dimensionality-reduction tool that significantly expands the scope of the original geometric method [31], allowing it to handle massive, high-dimensional distributed data streams in an efficient manner with approximation-quality guarantees. The key technical observation is that, by exploiting properties of the AMS estimator function, geometric monitoring can now take place in a *much lower-dimensional space*, allowing for communication-efficient monitoring. Effectively dealing with the highly non-linear median operator in the AMS estimator also mandates novel algorithmic solutions. We start by showing how our approximate function monitoring problem can be transformed into low-dimensional threshold crossing queries for the geometric method. To simplify notation, in the remainder of this section, we use $\tilde{\boldsymbol{v}}_i$ to denote the AMS sketch at remote sites and let $\tilde{\boldsymbol{v}} = \sum_j \tilde{\boldsymbol{v}}_j$ denote the global AMS sketch of the entire distributed stream; similarly, $\tilde{\boldsymbol{v}}^p_j$, $\tilde{\boldsymbol{v}}^p = \sum_i \tilde{\boldsymbol{v}}^p_j$ denote the local/global sketch values last communicated to the coordinator.

### 3.1 From Threshold Crossing to Approximate Function Monitoring

Consider the task of monitoring (at the coordinator) the value of a function $f()$ over the (full) global statistics vector $\boldsymbol{v}$ to within $\theta$ relative error. (Our discussion here focuses on relative error – the case of monitoring to within bounded *absolute* error can be handled in a similar manner, e.g., using the absolute to relative error transformation outlined under Theorem 2.1.) Since the coordinator only holds the estimated value of the global statistics vector $\boldsymbol{v}^p$ based on the most recent site updates, our monitoring protocol would have to guarantee that the estimated function value carries at most $\theta$ relative error compared to the up-to-date value $f(\boldsymbol{v}) = f(\boldsymbol{v}(t))$, that is $f(\boldsymbol{v}^p) \in (1 \pm \theta) f(\boldsymbol{v})$, which is obviously equivalent to monitoring two threshold queries on $f(\boldsymbol{v})$:

$$f(\boldsymbol{v}) \geq \frac{f(\boldsymbol{v}^p)}{1+\theta} \quad \text{and} \quad f(\boldsymbol{v}) \leq \frac{f(\boldsymbol{v}^p)}{1-\theta}.$$

Now, since we assume that the remote sites only maintain *AMS sketches* of their local vectors, all we have at our disposal are the sketched versions of the $\boldsymbol{v}$ and $\boldsymbol{v}^p$ vectors (denoted by $\tilde{\boldsymbol{v}}$ and $\tilde{\boldsymbol{v}}^p$, respectively), and the corresponding function values are approximated through the AMS estimator function $f_{\text{AMS}}()$ (Theorem 2.1). This, of course, implies a *sketching error* $\epsilon$ in the function values which can be bounded with the help of Theorem 2.1 so that $f_{\text{AMS}}(\tilde{\boldsymbol{v}}) \in (1 \pm \epsilon) f(\boldsymbol{v})$ with high probability (whp). Since our end goal is to guarantee that the sketch-based estimate available at the coordinator $f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p)$ is within $\theta$ relative error, the above threshold monitoring conditions become:

$$f(\boldsymbol{v}) \geq \frac{f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p)}{1+\theta} \quad \text{and} \quad f(\boldsymbol{v}) \leq \frac{f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p)}{1-\theta},$$

and, since $f_{\text{AMS}}(\tilde{\boldsymbol{v}}) \in (1 \pm \epsilon) f(\boldsymbol{v})$, it is not difficult to see that these two conditions are satisfied (whp) as long as:

$$f_{\text{AMS}}(\tilde{\boldsymbol{v}}) \geq \frac{f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p)(1+\epsilon)}{1+\theta} \quad \text{and} \quad f_{\text{AMS}}(\tilde{\boldsymbol{v}}) \leq \frac{f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p)(1-\epsilon)}{1-\theta}. \tag{1}$$

These are exactly the threshold conditions that our approximate function monitoring protocols will need to track. Note that $f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p)$ in the above expression is a constant (based on the latest communication of the coordinator with the remote sites). When either of the above conditions is violated, some (possibly all) remote sites must *flush* their current local stream estimates to the coordinator, updating $\tilde{\boldsymbol{v}}^p$ so that the difference between $f_{\text{AMS}}(\tilde{\boldsymbol{v}})$ and $f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p)$ is again small. Also, observe that the condition $\frac{1+\epsilon}{1+\theta} \leq \frac{1-\epsilon}{1-\theta}$ always holds as long as $\theta \geq \epsilon$, which is obviously the case (the overall error guarantee cannot be tighter than the incurred sketching error).

As discussed earlier, for remote site $j$, $\tilde{\boldsymbol{v}}_j$ denotes the sketch of local stream updates, and $\tilde{\boldsymbol{v}}^p_j$ the sketch last flushed to the coordinator. Exploiting the linearity of AMS sketches, remote site $j$ maintains $\Delta\tilde{\boldsymbol{v}}_j = \tilde{\boldsymbol{v}}_j - \tilde{\boldsymbol{v}}^p_j$, corresponding to stream updates since the last flush. At the time of the next flush, the remote site simply transfers $\Delta\tilde{\boldsymbol{v}}_j$ to the coordinator and resets its local delta sketch to zero. (If the stream updates sketched in $\Delta\tilde{\boldsymbol{v}}_j$ are few, in order to reduce communication cost, the remote site may send the updates verbatim to the coordinator.)

### 3.2 Applying the Geometric Method: Overview

Having reduced approximate distributed stream monitoring to appropriate threshold-crossing conditions (Eqn. (1)), we now turn our attention to the issue of effectively applying the geometric method to the problem at hand. A direct application would take the distributed stream sketch $\tilde{\boldsymbol{v}}$ as the global statistics vector, scaling each local sketch $\tilde{\boldsymbol{v}}_j$ by the number of sites $k$ to obtain the local statistics vectors (in order to satisfy the convex combination requirement of the geometric method); then, the geometric method could be employed to monitor the two threshold conditions on $f_{\text{AMS}}(\tilde{\boldsymbol{v}})$ in the $(n \times m)$-dimensional sketching space (Section 2).

Unfortunately, such a direct application of the geometric method turns out to perform poorly in practice, giving high communication overheads. The problem here is that, even though sketch vectors are a compressed, $(n \times m)$-dimensional representation of the full stream, they can still get fairly large, especially when tight error bounds are required. Thus, when a local threshold violation occurs at a remote site, it triggers a balancing process that requires some of the sites to transmit their local statistics (i.e., sketches) to the coordinator, imposing high communication overheads.

To address this issue, we develop a novel technique that allows us to track the threshold conditions on $f_{\text{AMS}}(\tilde{\boldsymbol{v}})$ through geometric monitoring in a much lower-dimensional space. More specifically, consider a sketch $\boldsymbol{x}$ as a two-dimensional $n \times m$ array, and let $\boldsymbol{x}[i]$ $(i = 1, \ldots m)$ denote the $n$-vector corresponding to the $i^{th}$ column of the sketch matrix. We define the local statistics vector for remote site $j$ as the *$m$-dimensional error vector* $\boldsymbol{d}_j$, where

$$\boldsymbol{d}_j[i] \;=\; \|\Delta\tilde{\boldsymbol{v}}_j[i]\| \;=\; \|\tilde{\boldsymbol{v}}_j[i] - \tilde{\boldsymbol{v}}^p_j[i]\|,$$

for $i = 1, \ldots, m$, and the global statistics vector as the ($m$-dimensional) average error vector $\boldsymbol{d} = \frac{1}{k}\sum_{j=1}^{k} \boldsymbol{d}_j$. In what follows, we show how to construct functions $F_u()$ and $F_l()$ of $\boldsymbol{d}$ that provide lower and upper bounds on $f_{\text{AMS}}(\tilde{\boldsymbol{v}})$; that is,

$$F_l(\boldsymbol{d}) \;\leq\; f_{\text{AMS}}(\tilde{\boldsymbol{v}}) \;\leq\; F_u(\boldsymbol{d}).$$

We can then monitor the threshold-crossing conditions on $f_{\text{AMS}}(\tilde{\boldsymbol{v}})$ (Eqn. (1)) using the geometric method for $F_u(\boldsymbol{d})$ and $F_l(\boldsymbol{d})$ in the $m$-dimensional space of error vectors $\boldsymbol{d}$. It is important to note that this optimization implies huge communication savings: Sketch matrices are typically very "thin", i.e., $n >> m$, since $n$ depends quadratically on the sketching error $\epsilon$, whereas $m$ depends only logarithmically on the desired confidence $\delta$ [2, 1, 9, 15].

Another major technical challenge that arises is how to effectively test the monochromicity of bounding balls in the resulting lower-dimensional space with respect to threshold conditions involving the highly non-linear median operator present in the AMS estimator (as well as the upper/lower bound functions $F_u()$ and $F_l()$). Our techniques and analyses make use of three well-known properties of the median operator:

**Monotonicity:** If $\boldsymbol{x}[i] \leq \boldsymbol{y}[i]$ for all $i$, then $\mathrm{median}_i\{\boldsymbol{x}[i]\} \leq \mathrm{median}_i\{\boldsymbol{y}[i]\}$.

**Distributivity:** For any monotone function $f()$, $\mathrm{median}_i\{f(\boldsymbol{x}[i])\} = f(\mathrm{median}_i\{\boldsymbol{x}[i]\})$.

**Homogeneity:** $\forall\lambda \in \mathbb{R}$, $\mathrm{median}_i\{\lambda\boldsymbol{x}[i]\} = \lambda\, \mathrm{median}_i\{\boldsymbol{x}[i]\}$.

We propose a number of novel algorithmic techniques to address the aforementioned technical challenges for three different types of distributed stream queries of high practical interest. We start with the easier cases of $L_2$-norm (i.e., self-join) and range queries, and then extend our approach to the case of general inner-product (i.e., binary-join) queries.

## 3.3 Monitoring Self-Joins

In the case of (approximate) self-join/$L_2$-norm queries, our goal is to track an estimate of the (squared) norm of a frequency vector using AMS sketches. Thus, we need to monitor the values of the AMS estimator function

$$f_{\text{AMS}}(\tilde{\boldsymbol{v}}) = \underset{i=1,\ldots,m}{\mathrm{median}}\{\frac{1}{n}\sum_{l=1}^{n}(\tilde{\boldsymbol{v}}[l,i])^2\} = \underset{i=1,\ldots,m}{\mathrm{median}}\{\frac{1}{n}\|\tilde{\boldsymbol{v}}[i]\|^2\} \quad (2)$$

where $\tilde{\boldsymbol{v}}$ is an $n\times m$-sized AMS sketch and $\tilde{\boldsymbol{v}}[i]$ is the $i^{th}$-th column of the sketch. Using the distributivity of the median operator, the threshold-crossing conditions in Eqn. (1) become:

$$\sqrt{n\frac{1+\epsilon}{1+\theta}f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p)} \leq \underset{i=1,\ldots,m}{\mathrm{median}}\{\|\tilde{\boldsymbol{v}}[i]\|\} \leq \sqrt{n\frac{1-\epsilon}{1-\theta}f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p)}.$$

We now develop "safe" threshold conditions over $\mathbb{R}^m$ for the above monitoring problem using upper/lower bound functions defined over the $m$-dimensional error vector $\boldsymbol{d}$. By definition, at site $j$, $\boldsymbol{d}_j[i] = \|\tilde{\boldsymbol{v}}_j[i] - \tilde{\boldsymbol{v}}_j^p[i]\|$; thus, applying the the triangle inequality, we have

$$\|\tilde{\boldsymbol{v}}[i] - \tilde{\boldsymbol{v}}^p[i]\| \leq \sum_{j=1}^{k}\|\tilde{\boldsymbol{v}}_j[i] - \tilde{\boldsymbol{v}}_j^p[i]\| = \sum_{j=1}^{k}\boldsymbol{d}_j[i] = k\boldsymbol{d}[i], \quad (3)$$

or, equivalently,

$$\|\tilde{\boldsymbol{v}}^p[i]\| - k\boldsymbol{d}[i] \leq \|\tilde{\boldsymbol{v}}[i]\| \leq \|\tilde{\boldsymbol{v}}^p[i]\| + k\boldsymbol{d}[i].$$

Then, by monotonicity of the median, it is sufficient to monitor the following threshold conditions over $\boldsymbol{d} \in \mathbb{R}^m$:

$$F_u(\boldsymbol{d}) = \underset{i}{\mathrm{median}}\{\|\tilde{\boldsymbol{v}}^p[i]\| + k\boldsymbol{d}[i]\} \leq \sqrt{n\frac{1-\epsilon}{1-\theta}f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p)}$$

$$F_l(\boldsymbol{d}) = \underset{i}{\mathrm{median}}\{\|\tilde{\boldsymbol{v}}^p[i]\| - k\boldsymbol{d}[i]\} \geq \sqrt{n\frac{1+\epsilon}{1+\theta}f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p)}.$$

**Geometric Monitoring for the Median.** By dividing both sides of the above threshold conditions over $\mathbb{R}^m$ by $\pm k$ and by virtue of the homogeneity of the median, both conditions take the general form

$$F(\boldsymbol{d}) = \underset{i=1,\ldots,m}{\mathrm{median}}\{\boldsymbol{a}[i] + \boldsymbol{d}[i]\} \leq \zeta,$$

where $\boldsymbol{a}$ is a constant $m$-dimensional vector and $\zeta \in \mathbb{R}$.

**Algorithm 1**: Computing the distance of a vector to the region defined by a median threshold.

```
Data: $c = [c[1], \ldots, c[m]]$, $a = [a[1], \ldots, a[m]]$:
      $m$-dimensional vectors; $\zeta$: real.
Result: The distance of $c$ to the region
      $\{x \in \mathbb{R}^m | \mathrm{median}_i\{a[i] + x[i]\} \geq \zeta\}$.
begin
    let $z = a + c$
    Sort($z$, ascending)
    $r = 0$
    for $i \longleftarrow \lfloor \frac{m+1}{2} \rfloor$ to $m$ do
        if $z[i] < \zeta$ then
            $r$ += $(\zeta - z[i])^2$
            $z[i] = \zeta$
    return $\rho_j = \sqrt{r}$
end
```

To monitor such conditions using the geometric method, we must be able, given a bounding ball $B(\boldsymbol{c}, \rho)$ in $\mathbb{R}^m$, to efficiently decide whether the ball is monochromatic; that is, whether $\mathrm{median}_i\{\boldsymbol{a}[i] + \boldsymbol{x}[i]) \leq \zeta$, for all $\boldsymbol{x} \in B(\boldsymbol{c}, \rho)$, This can be done by determining the Euclidean distance $\rho_\zeta$ of the ball center $\boldsymbol{c}$ from the closest point in the inadmissible region $Z = \{\boldsymbol{x} \in \mathbb{R}^m | F(\boldsymbol{x}) \geq \zeta\}$; then, the ball $B(\boldsymbol{c}, \rho)$ is monochromatic if and only if $\rho \leq \rho_\zeta$.

We now show how to efficiently compute this distance to the inadmissible region. Clearly, if $\mathrm{median}_i\{\boldsymbol{a}[i]+\boldsymbol{c}[i]\} \geq \zeta$, then $\rho_\zeta = 0$. Else, we can employ a greedy algorithm to find a point $\boldsymbol{z}$ on the boundary of the inadmissible region $Z$ (with $\mathrm{median}_i\{\boldsymbol{a}[i] + \boldsymbol{z}[i]\}) = \zeta$), such that no other point in $Z$ is closer to the ball center $\boldsymbol{c}$ (note that this point $\boldsymbol{z}$ is not necessarily unique). Algorithm 1 constructs such a boundary point $\boldsymbol{z}$ in a greedy manner: Starting with $\boldsymbol{z} = \boldsymbol{a} + \boldsymbol{c}$, it takes all coordinates from rank $\lfloor \frac{m+1}{2} \rfloor$ to rank $m$ that are $< \zeta$ and sets them equal to $\zeta$ in order to reach the boundary; then, it returns the distance $\rho_\zeta = \|\boldsymbol{c} - \boldsymbol{z}\|$. The following theorem summarizes our analysis (due to space constraints, the proof is deferred to the full paper).

THEOREM 3.1. *Algorithm 1 correctly computes the minimum Euclidean distance of point* $\mathbf{c} \in \mathbb{R}^m$ *from the inadmissible region* $Z = \{\boldsymbol{x} \in \mathbb{R}^m | \mathrm{median}_i\{\boldsymbol{a}[i] + \boldsymbol{x}[i]\} \geq \zeta\}$ *in time* $O(m \log m)$.

## 3.4 Monitoring Range Aggregates

We now turn our attention to a different special type of inner-product queries, namely the inner product of a distributed data stream with a *constant* vector $\boldsymbol{b}$. An important special case here is that of *range aggregates*, in which the constant vector $\boldsymbol{b}$ simply contains non-zero values for a subset $S$ of values in the joint data distribution in the streaming vector $\boldsymbol{v}$, and zero everywhere else; thus, $\boldsymbol{b} \cdot \boldsymbol{v} = \sum_{i\in S}\boldsymbol{b}[i]\boldsymbol{v}[i]$, i.e., the distribution aggregate (e.g., the number of tuples) in range $S$. These aggregates can, of course, be estimated using an AMS sketch estimator $f_{\text{AMS}}(\tilde{\boldsymbol{v}}, \tilde{\boldsymbol{b}})$ (where $\tilde{\boldsymbol{b}}$ is the constant sketch vector for $\boldsymbol{b}$), with the quality guarantees outlined in Theorem 2.1. Such approximate range aggregates over AMS sketches have been utilized in several important streaming applications, including the construction of effective quantile, histogram, and wavelet summaries over streaming data [9, 19, 20, 32]. For instance, in the case of wavelets, we are interested in estimating large wavelet coefficients, which are inner product of the data distribution with constant wavelet-basis vectors [9].[2]

Within our framework, we are asked to monitor the estimator

$$f_{\text{AMS}}(\tilde{\boldsymbol{v}}, \tilde{\boldsymbol{b}}) = \underset{i=1,\dots,m}{\text{median}}\{\frac{1}{n}\tilde{\boldsymbol{b}}[i]\tilde{\boldsymbol{v}}[i]\}.$$

The global statistic again consists of a single sketch (since $\boldsymbol{b}$ is constant). Thus, we can start from Eqn. (3), and we obtain

$$\left|\boldsymbol{b}[i](\tilde{\boldsymbol{v}}[i] - \tilde{\boldsymbol{v}}^p[i])\right| \le \|\boldsymbol{b}[i]\|\|\tilde{\boldsymbol{v}}[i] - \tilde{\boldsymbol{v}}^p[i]\| \le k\boldsymbol{d}[i]\|\boldsymbol{b}[i]\|,$$

which yields the two threshold conditions:

$$\underset{i}{\text{median}}\{\boldsymbol{b}[i]\tilde{\boldsymbol{v}}^p[i] + k\boldsymbol{d}[i]\|\boldsymbol{b}[i]\|\} \le n\frac{1-\epsilon}{1-\theta}f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p, \boldsymbol{b}) \quad (4)$$

$$\underset{i}{\text{median}}\{\boldsymbol{b}[i]\tilde{\boldsymbol{v}}^p[i] - k\boldsymbol{d}[i]\|\boldsymbol{b}[i]\|\} \ge n\frac{1+\epsilon}{1+\theta}f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p, \boldsymbol{b}). \quad (5)$$

**Geometric Monitoring for the Median of Linear Forms.** By dividing both sides by $\pm k$ and by virtue of the homogeneity of the median, both conditions take the form

$$F(\boldsymbol{d}) = \underset{i}{\text{median}}\{\boldsymbol{a}[i] + \boldsymbol{b}[i]\boldsymbol{d}[i]\} \le \zeta$$

for $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^m$, where $\boldsymbol{b}$ has nonnegative components and $\zeta \in \mathbb{R}$.

The monochromicity question for ball $B(\boldsymbol{c}, \rho)$ can be addressed in a spirit similar to that of Section 3.3. One small complication arises by the fact that, in this case, $\rho_\zeta$ may be undefined! This may occur only if some entries in $\boldsymbol{b}$ are zero, so that the inadmissible region $Z = \{\boldsymbol{x} \in \mathbb{R}^m \mid F(\boldsymbol{x}) \ge \zeta\}$ is empty. To handle this complication smoothly, we apply a standard algebraic perturbation trick; we assume that, when $\boldsymbol{b}[i] = 0$ and $\boldsymbol{a}[i] < \zeta$, then $\boldsymbol{a}[i] + \boldsymbol{b}[i](+\infty) \ge \zeta$. Thus, region $Z$ is never empty, although some of its elements may have infinite coordinates.

As earlier, we wish to minimize $\rho_\zeta^2 = \sum_{i=1}^m(\boldsymbol{x}[i]-\boldsymbol{c}[i])^2$, where $\boldsymbol{x} \in Z$. For each $i = 1, ..., m$, let $r_i^2$ denote the minimum of $(\boldsymbol{x}[i] - \boldsymbol{c}[i])^2$, such that $\boldsymbol{a}[i] + \boldsymbol{b}[i]\boldsymbol{x}[i] \ge \zeta$. It is easy to derive that,

$$r_i^2 = \begin{cases} 0 & \text{if } \boldsymbol{a}[i] + \boldsymbol{b}[i]\boldsymbol{c}[i] \ge \zeta \\ +\infty & \text{if } \boldsymbol{a}[i] < \zeta \text{ and } \boldsymbol{b}[i] = 0 \\ (\frac{\zeta - \boldsymbol{a}[i]}{\boldsymbol{b}[i]} - \boldsymbol{c}[i])^2 & \text{if } \boldsymbol{a}[i] + \boldsymbol{b}[i]\boldsymbol{c}[i] < \zeta \text{ and } \boldsymbol{b}[i] \ne 0 \end{cases}$$

Then, $\rho_\zeta^2$ is equal to the sum of the $(m+1)/2$ smallest $r_i^2$s (treating ties arbitrarily). Again, the ball is monochromatic if and only if $\rho \le \rho_\zeta$.

## 3.5 Monitoring General Inner Products

We now turn our attention to a more complicated monitoring problem, where the global statistic comprises of the concatenation of two sketches $\langle\tilde{\boldsymbol{v}}, \tilde{\boldsymbol{u}}\rangle$ corresponding to two distributed streams, and the monitored function is the sketch estimate of the inner product of the sketched vectors, corresponding to the size of the inner product (i.e., join):

$$\begin{aligned} f_{\text{AMS}}(\tilde{\boldsymbol{v}}, \tilde{\boldsymbol{u}}) &= \underset{i=1,\dots,m}{\text{median}}\{\frac{1}{n}\sum_{l=1}^{n}\tilde{\boldsymbol{v}}[l,i]\tilde{\boldsymbol{u}}[l,i]\} \\ &= \underset{i=1,\dots,m}{\text{median}}\{\frac{1}{n}\tilde{\boldsymbol{v}}[i]\tilde{\boldsymbol{u}}[i]\} \end{aligned}$$

In addition, error vectors are also concatenated, denoted as $\langle\boldsymbol{d}_v, \boldsymbol{d}_u\rangle$.

[2]Note that sketching is employed here since the range queries of interest are *not fixed* (i.e., can vary over time), and a search over the sketch summary is needed to discover the ranges of interest as the stream distribution changes. In simpler scenarios where the range aggregate of interest is fixed, slack-allocation techniques for tracking linear aggregates can be used (e.g., [24]).

We now develop bounds for the monitored function using the error vectors. From Eqn. (3), we can write

$$\tilde{\boldsymbol{v}}[i] = \tilde{\boldsymbol{v}}^p[i] + k\boldsymbol{d}_v[i]\boldsymbol{q}_{v,i} \quad \text{and} \quad \tilde{\boldsymbol{u}}[i] = \tilde{\boldsymbol{u}}^p[i] + k\boldsymbol{d}_u[i]\boldsymbol{q}_{u,i},$$

where $\boldsymbol{q}_{v,i}$ and $\boldsymbol{q}_{u,i}$ are (unknown) vectors of length at most 1. Thus,

$$\begin{aligned} \tilde{\boldsymbol{v}}[i]\tilde{\boldsymbol{u}}[i] = \tilde{\boldsymbol{v}}^p[i]\tilde{\boldsymbol{u}}^p[i] + k\boldsymbol{d}_v[i]\boldsymbol{q}_{v,i}\tilde{\boldsymbol{u}}^p[i] + k\boldsymbol{d}_u[i]\boldsymbol{q}_{u,i}\tilde{\boldsymbol{v}}^p[i] \\ + k^2\boldsymbol{d}_v[i]\boldsymbol{d}_u[i]\boldsymbol{q}_{v,i}\boldsymbol{q}_{u,i} \end{aligned}$$

Exact maximization/minimization of the above condition is possible but yields formulas that are too unwieldy. We provide slightly weaker upper and lower bounds by treating each term in the above sum separately. Then, applying median monotonicity, we get the following conditions:

$$\begin{aligned} &\underset{i}{\text{median}}\{\tilde{\boldsymbol{v}}^p[i]\tilde{\boldsymbol{u}}^p[i] + k\boldsymbol{d}_v[i]\|\tilde{\boldsymbol{u}}^p[i]\| + k\boldsymbol{d}_u[i]\|\tilde{\boldsymbol{v}}^p[i]\| \\ &\qquad + k^2\boldsymbol{d}_v[i]\boldsymbol{d}_u[i]\} \le n\frac{1-\epsilon}{1-\theta}f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p, \tilde{\boldsymbol{u}}^p) \\ &\underset{i}{\text{median}}\{\tilde{\boldsymbol{v}}^p[i]\tilde{\boldsymbol{u}}^p[i] - k\boldsymbol{d}_v[i]\|\tilde{\boldsymbol{u}}^p[i]\| - k\boldsymbol{d}_u[i]\|\tilde{\boldsymbol{v}}^p[i]\| \\ &\qquad - k^2\boldsymbol{d}_v[i]\boldsymbol{d}_u[i]\} \ge n\frac{1+\epsilon}{1+\theta}f_{\text{AMS}}(\tilde{\boldsymbol{v}}^p, \tilde{\boldsymbol{u}}^p) \end{aligned}$$

**Geometric Monitoring for the Median of Bilinear Forms.** By dividing both sides by $\pm k^2$ and by virtue of the homogeneity of the median, both conditions take the form:

$$F(\boldsymbol{x}, \boldsymbol{y}) = \underset{i}{\text{median}}\{\boldsymbol{x}[i]\boldsymbol{y}[i] + \boldsymbol{a}[i]\boldsymbol{x}[i] + \boldsymbol{b}[i]\boldsymbol{y}[i] + \boldsymbol{g}[i]\} \le \zeta,$$

with variables $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^m$, and constants $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{g} \in \mathbb{R}^m$, where $\boldsymbol{a}, \boldsymbol{b}$ have nonnegative components, and $\zeta \in \mathbb{R}$.

In order to apply the geometric method, we need to determine the monochromicity of balls of error vectors (in the combined $2m-$dimensional space. In other words, we need to determine whether a ball defined by $(\boldsymbol{x} - \boldsymbol{c})^2 + (\boldsymbol{y} - \boldsymbol{c}')^2 \le \rho^2$ (where $\boldsymbol{c}, \boldsymbol{c}'$ are $m$-vectors) intersects the interior of the inadmissible region $Z = \{\langle\boldsymbol{x}, \boldsymbol{y}\rangle \in \mathbb{R}^{2m} \mid F(\boldsymbol{x}, \boldsymbol{y}) \ge \zeta\}$. For this problem, essentially the same reasoning applied to the range query case leads us to a bound-of-ball-radius solution.

Given a ball $B(\langle\boldsymbol{c}, \boldsymbol{c}'\rangle, \rho)$, let some vector $\langle\boldsymbol{x}, \boldsymbol{y}\rangle \in Z$ be a nearest neighbor of $\langle\boldsymbol{c}, \boldsymbol{c}'\rangle$. Then,

$$\rho_\zeta^2 = (\boldsymbol{x}-\boldsymbol{c})^2+(\boldsymbol{y}-\boldsymbol{c}')^2 = \sum_{i=1}^{m}(\boldsymbol{x}[i]-\boldsymbol{c}[i])^2+(\boldsymbol{y}[i]-\boldsymbol{c}'[i])^2 \quad (6)$$

We compute the (squared) distance $\rho_\zeta^2$ of the center $\langle\boldsymbol{c}, \boldsymbol{c}'\rangle$ to region $Z$ by computing, for each component $i = 1, \dots, m$ of the median operator in $F()$, a squared coefficient, $r_i^2$. Each $r_i^2$ corresponds to a term in the sum of Eqn. (6). Then, the (squared) distance of center $\langle\boldsymbol{c}, \boldsymbol{c}'\rangle$ to the boundary of the inadmissible region $Z$ is obtained by summing the $\lfloor(m+1)/2\rfloor$ smallest $r_i^2$.

We now turn to the computation of $r_i^2$. To keep notation clean, we drop the $i$-index from the variables: Let $c$ and $c'$ be the $i$-th coordinates of $\boldsymbol{c}$ and $\boldsymbol{c}'$, respectively (and, similarly, for $a, b, g, x, y$). If $cc' + ac + bc' + g \ge \zeta$, then $r^2 = 0$. Else, we need to compute $x$ and $y$ which minimize $(x-c)^2+(y-c')^2$ but set the corresponding component of the median to $\zeta$, that is,

$$r^2 = \inf\{(x-c)^2 + (y-c')^2 \mid xy + ax + by + g = \zeta\}$$

To simplify the problem, first rewrite

$$xy + ax + by + g = (x + b)(y + a) + g - ab$$

By substituting $p = x + b$ and $q = y + a$, we have:

$$r^2 = \inf\{(p - \alpha)^2 + (q - \beta)^2 \mid pq = \tau\},$$

where $\alpha = c + b$, $\beta = c' + a$ and $\tau = \zeta + ab - g$.

Now, if $\tau = 0$, then $r^2 = \min(\alpha, \beta)$; else, substitute $q = \tau/p$ to obtain $(p - \alpha)^2 + (\frac{\tau}{p} - \beta)^2$. Taking the derivative equal to 0 reduces to the quartic equation

$$f(p) = p^4 - \alpha p^3 + \tau\beta p - \tau^2 = 0.$$

One of its real roots yields the smallest $r^2$ (real roots exist since $f(0) < 0$).

## 3.6 Synchronization Policies for Remote Sites

When the coordinator determines from the geometric method that there is a global violation in the monitoring (that is, the global error vector $\boldsymbol{d}$ is no longer within the admissible region), the coordinator signals some remote sites to flush their updates $\Delta\tilde{\boldsymbol{v}}_j$. While several flushing policies are possible, we describe two alternatives.

**Eager Synchronization.** This is a simple policy, where all remote sites synchronize concurrently. Each remote site $j$ transmits its current stream updates $\Delta\tilde{\boldsymbol{v}}_j$. After the end of this process, the system reaches a state where, for all $j$, $\tilde{\boldsymbol{v}}_j = \tilde{\boldsymbol{v}}_j^p$ and thus $\boldsymbol{d}_j = \boldsymbol{d} = 0$. Then, new bounds for the error are computed and broadcast to all remote sites and stream processing begins anew.

**Lazy Synchronization.** In eager syncronization, even sites whose local updates are few are forced to synchronize. This may be wasteful, and unnecessary; these sites are probably not contributing to the error significantly. A lazy approach would be for the coordinator to syncronize a minimum number of sites, necessary to restore global bounds. Remote sites are ranked in (descending) order of the number of unflushed updates (other choices, such as the distance of $\boldsymbol{d}_j$ to the inadmissible zone, are possible, but our experiments indicated that they do not perform as well). Then, sites are asked to flush sequentially, until, after some flush, the global error $\boldsymbol{d}$ is again restored within the (updated) bounds.

## 4. EXPERIMENTAL STUDY

In this section, we discuss the empirical evaluation of our techniques using real-life data sets. We start by discussing our testbed and methodology.

**Data Sets and Techniques.** We use the same real-life data sets as [7] for our experiments. The first data set, **WCup**[3], was drawn from the Internet Traffic Archive and contains HTTP requests sent to the servers hosting the World Cup 1998 web site (totaling approximately 1.35 billion requests over a three-month period). The second data set, **Cdad**[4], comprises SNMP network usage data obtained from CRAWDAD (the Community Resource for Archiving Wireless Data at Dartmouth). It consists of measurements of total network traffic every five minutes over a four month period at a large number of access points (approximately 200) inside a corporate research center (IBM Watson). We tracked the distribution of the `size` attribute from **WCup** and the `shortRet` attribute from **Cdad**, since both these attributes take a very large number of values thus making streaming estimation challenging.

From each data set, we construct a distributed stream for a number of remote sites, by hashing the site field from the data set to the desired number of remote sites in each experiment (**WCup** relates to 26 sites and **Cdad** to 27). Thus, skew in the datasets also appears in our streams. We focus primarily on self-join queries over these streams, as these queries are not parameterized and their sketching error is predictable.

We experimented with our sketch-based geometric monitoring schemes using both the eager and the lazy synchronization policy (denoted by **GM-lazy** and **GM-eager**, respectively). To demonstrate their effectiveness, our methods are contrasted against the sketch-based monitoring technique of [7] (denoted by **CG**). In a nutshell, **CG** is a purely "push-based" monitoring protocol: Each site $j$ continuously tracks the value of its relative delta sketch vector norm $\frac{\|\tilde{\boldsymbol{v}}_j - \tilde{\boldsymbol{v}}_j^p\|}{\|\tilde{\boldsymbol{v}}_j\|}$ checking that it is below an upper bound that depends on $\epsilon, \theta$, and the number of sites (determined by the analysis in [7]). When that upper bound is violated, the site simply sends the coordinator its local delta sketch vector (or, the local updates themselves, if smaller), resetting its delta to zero, and resumes its local tracking.

All three methods were implemented using the Fast-AMS sketching technique [7]. Furthermore, since our **GM** schemes are *static* (i.e., do not try to predict the evolution of local/global statistics vectors), we compared them against the static variant of the **CG** technique [7]. As mentioned earlier, the idea of using dynamic *prediction models* (as suggested in [7]) is essentially orthogonal to the ideas in this paper, and prediction models have recently been shown to significantly improve the performance of geometric monitoring as well [17]. We defer the comparison of the dynamic, prediction-based variants of the schemes to the full version of this paper.

**Metrics.**

Our main focus is on the *communication cost* incurred by our method. We distinguish two parts in the total communication traffic. The first part, *data communication*, comprises messages transmitted from remote sites to the coordinator, when remote sites flush their sketched updates $\Delta\tilde{\boldsymbol{v}}_j$ (or, the list of update records verbatim, if smaller). The second part comprises the *monitoring overhead* of the geometric method, for tracking the global error vector $\boldsymbol{d}$. Studying data communication in isolation, provides a better contrast to the method of [7], since this is the only type of communication performed by that method. In their technique, flushes happen by each remote site when a purely local condition is violated. Our technique, in its effort to delay flushes (improving the effectiveness of sketching) by balancing local errors, incurs additional monitoring overhead. Naturally, we are interested in the cost of this overhead, relative to the gains in data communication costs.

In addition, the separation of these traffic costs makes sense because, in principle, it is possible that the coordinator for the protocols of the geometric method is not co-located in the same machine with the site which collects the global stream updates. For example, if communication channels among remote sites are good, the role of coordinator may be assigned to one of the sites. Furthermore, the traffic patterns of these two types of communication differ significantly: Data communication traffic consists of large messages, travelling from remote sites to the collection site only. Geometric monitoring traffic, on the other hand, consists of small messages, which can easily fit in a single UDP datagram. Upstream traffic (from remote sites to coordinator) is almost equal in volume to downstream traffic (from coordinator to sites). Moreover, downstream traffic consists of identical messages to all sites, and thus it can be implemented by multicast channels. For these reasons, distinguishing between these types of communication can highlight the suitability of each technique in different distributed settings.

Another interesting metric is the *scalability* of our monitoring schemes as the the number of sites collecting the distributed stream becomes larger. In the technique of [7], each site's local condition

[3] `http://ita.ee.lbl.gov/html/contrib/WorldCup.html`

[4] `http://crawdad.cs.dartmouth.edu/meta.php?name=ibm/watson#N100AD`

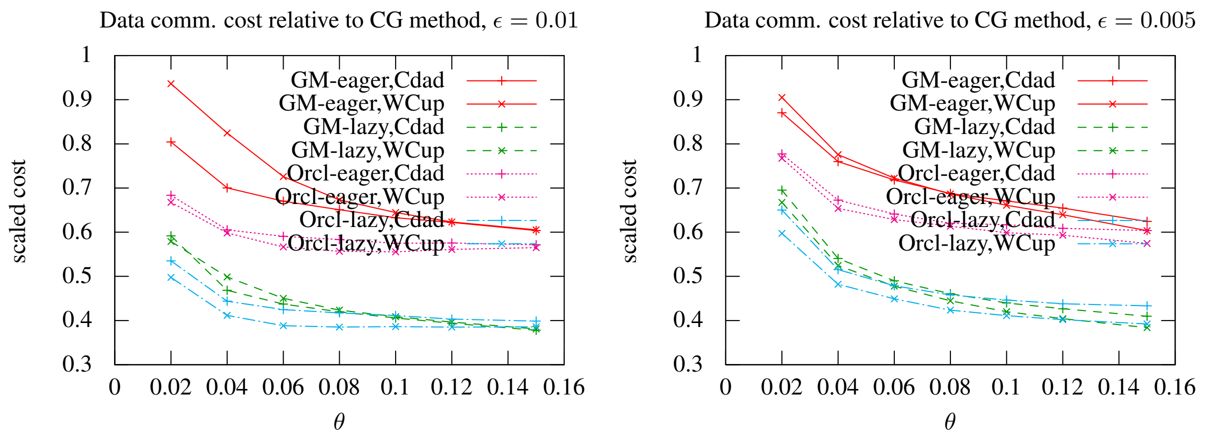


Figure 3: Self-join, data communication costs, as a fraction of the cost of **CG**.

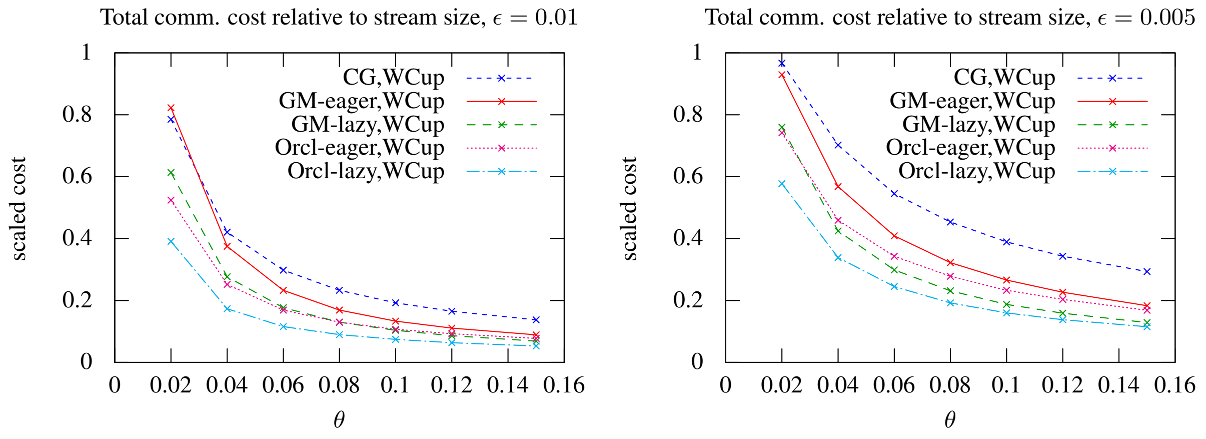


Figure 4: Self-join, total communication cost, as a fraction of stream size (**WCup** dataset).

becomes stricter, causing more frequent flushes—thus, less opportunity for communication reduction via sketching. For our technique, data communication is not increased with more sites, in fact it decreases slightly, as with more nodes there is greater opportunity for balancing. Unfortunately, the overhead of the geometric method increases significantly with the number of sites, rendering our techniques non-scalable to many sites. Although a proper study of scalability is outside the scope of this paper, we present some scalability results, in order to motivate further research.

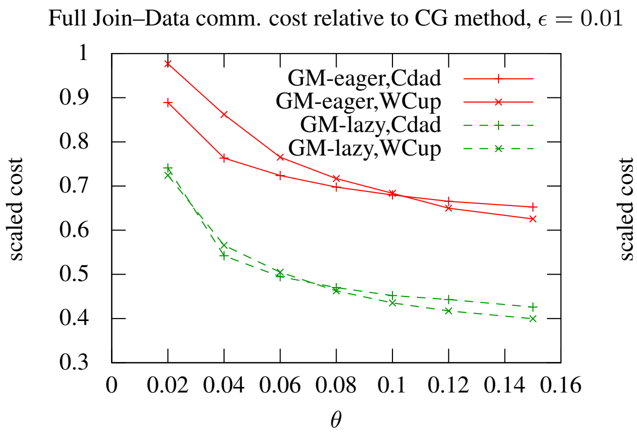
**Results: Communication Cost.** The results presented measure the communication cost incurred by our methods. In order to contrast better with the techniques of [7], we do not present absolute cost, by rather the cost scaled relative to the cost of the **CG** method (which has scaled cost 1).

In these experiments, the number of remote sites is 4. We used two different sketch sizes. In both, $\delta = 1/2^7$ (thus, sketches had 7 columns each). The first sketch is built for sketching error $\epsilon = 0.01$ and the second, larger sketch is built for $\epsilon = 0.005$.

Fig. 3 depicts the (relative) data communication cost for our methods, as a function of the total monitoring error $\theta$. It can be seen that both variants of our technique improve significantly upon the cost of the previous technique, with the lazy variant performing much better than the eager one.

A natural question that arises is how much further one can reduce data communication in this framework. To quantify the potential improvement, Fig. 3 depicts also the costs of an *unrealistic oracle-based scheme*, in which data is collected from sites (using either the eager or the lazy flushing policy) *only when a global violation occurs*. As can be seen, the cost of our lazy strategy is quite close to that of the lazy oracle-based one, leaving very little room for improvement. The costs of our eager strategy, while not as close to the eager oracle-based one, are still near. These results validate our claim that monitoring in a lower-dimensional space via the error vector $\boldsymbol{d}$ provides an excellent compromise between monitoring accuracy and monitoring cost.

The total communication costs are depicted in Fig. 4, as a function of stream size. For the technique of [7], this cost is equal to that depicted in Fig. 3, whereas our techniques incur additional cost related to error monitoring. Still, as shown in , this additional cost is well-worth. Our techniques still outperform that of [7], sometimes by up to 35%. Note that, to keep to plot clear, we only show the graphs for the **WCup** dataset. The graphs for the **Cdad** dataset almost coincide.



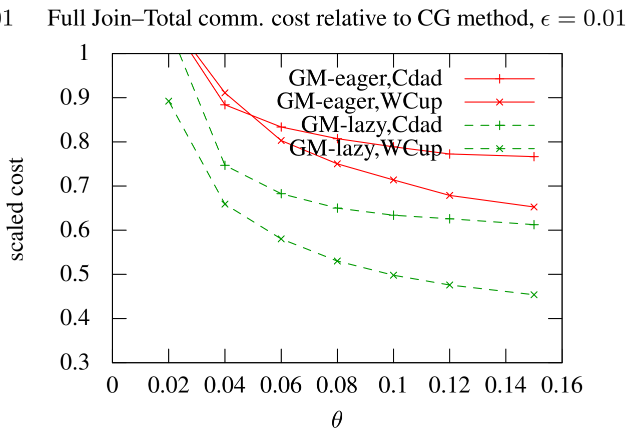
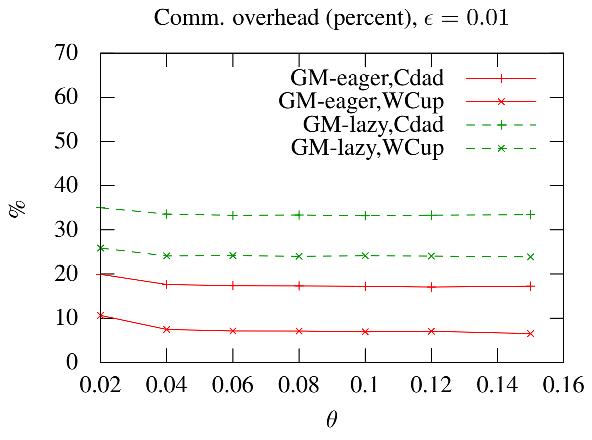


Figure 5: Full-join, data and total communication costs, as a fraction of the cost of **CG**, for sketching error $\epsilon = 0.01$.



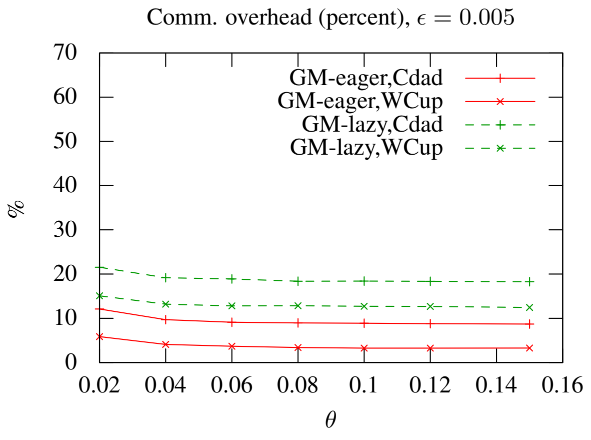


Figure 6: Self-join monitoring overhead as a percent of total cost.

In Fig. 6, the overhead incurred by geometric monitoring is shown as a function of $\theta$. Although the overhead remains relatively constant as $\theta$ increases, it is higher for $\epsilon = 0.01$ (where sketches are smaller, imposing less per-flush cost). Interestingly, the lazy method has significantly higher overhead (as a percent) over the eager method. This is due to two factors; first, because the eager method exhibits higher data communication, and second, because intuitively, the lazy method performs more rebalancing, as it delays flushing some sites. Still, the additional overhead is justified for the lazy method, because the savings in data communication are greater.

Other types of monitored queries behave similarly to the self-join query. Due to space restrictions we only present results for our most general full-join query. Fig. 5 presents data and total communication costs for monitoring the join of two streams with sketching error $\epsilon = 0.01$. From each data set we created two streams by splitting the records (**WCup** dataset was split on the `clientID` attribute and **Cdad** was split on the `site` attribute). The join attributes were the same ones tracked in the self-join experiments (`size` and `shortRet` respectively). The same broad effect, of much reduced data communication over **CG** with modest monitoring overhead is observed in this case as well.

## 4.1 Effect of sketch size

Sketching accuracy $\epsilon$ affects communication cost more significantly than the probability bound $\delta$. As sketch size grows with $\log(1/\delta)$, reasonable values of $\delta$ (say, from $2^{-11}$ to $2^{-7}$ will only affect the sketch size modestly.

Sketching accuracy affects sketch size more strongly, as it increases with $O(1/\epsilon^2)$. When overall accuracy $\theta$ is kept constant, the increased accuracy of larger sketches implies that the global sketch will need to be updated less frequently, incurring fewer, albeit larger messages. This implies a trade-off between number of messages and message size.

We now study the trade-off between sketching accuracy (and sketch size) and monitoring accuracy. Fig. 7 depicts data communication cost for overall accuracy $\theta = 0.04$, as the ratio $\epsilon/\theta$ varies from 0.1 to 0.95. Note that, this cost is normalized (to the total size of all stream updates), and not relative to CG (which is actually also shown in the graph).

The analysis of [7] indicates that their technique performs best for $\epsilon \approx \theta/2$. This is exhibited by our experiments, for our techniques also.

However, the benefit of our technique over that of [7] should be greater when $\epsilon$ is smaller than $\theta/2$, that is, when sketches are

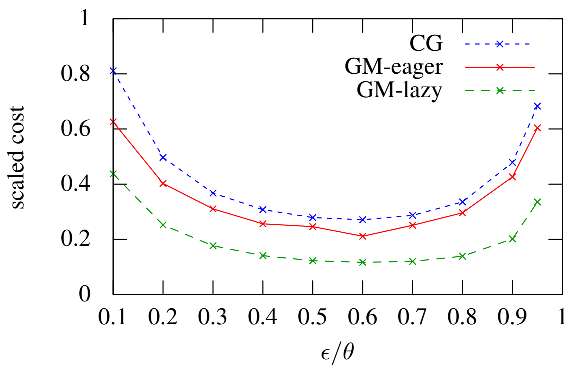


Figure 7: Data comm. cost relative to stream size, over $\epsilon/\theta$, for $\theta = 0.04$ (**WCup** dataset).

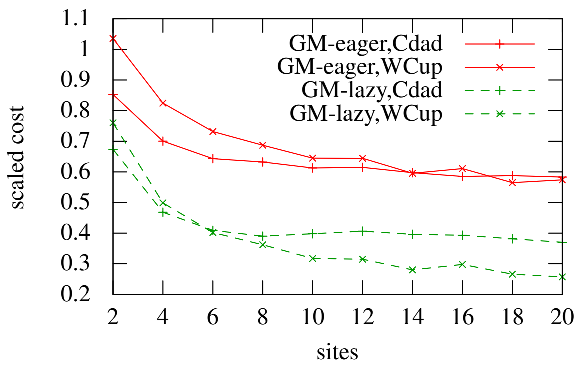


Figure 9: Data comm. cost over number of sites, as a fraction of the cost of **CG**, for $\epsilon = 0.01$ and $\theta = 0.04$.

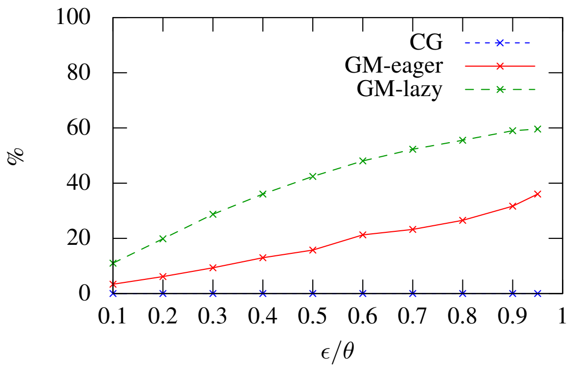


Figure 8: Comm. overhead (percent of total comm.), over $\epsilon/\theta$, for $\theta = 0.04$ (**WCup** dataset).

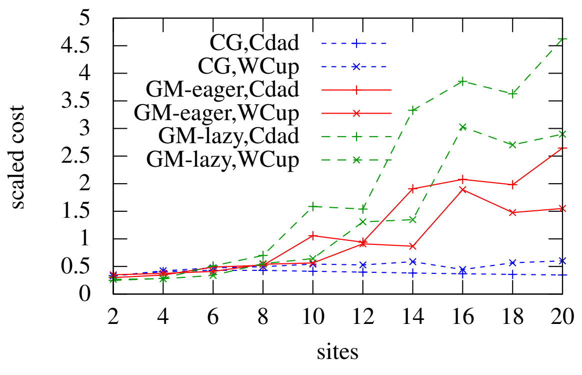


Figure 10: Total comm. over number of sites, as a fraction of stream size, for $\epsilon = 0.01$ and $\theta = 0.04$.

more accurate and larger, because balancing the global error can be done more effectively (and cheaply) when monitoring accuracy is relaxed. Indeed, Fig. 8 shows that the overhead grows significantly as $\theta$ approaches $\epsilon$. Again, note that the lazy technique has higher overhead than the eager technique (CG has overhead 0 in this plot).

In practice, it may be desirable that applications utilize sketches of small $\epsilon$, relying on relaxed monitoring precision in order to decrease communication. This is because $\epsilon$ cannot be adjusted on-the-fly, once a stream has started to be sketched, whereas adjusting $\theta$ can be done on-line very easily.

## 4.2 Scalability

To measure the behavior of our techniques as the number of sites grows, we conducted experiments where the number of sites monitoring a stream increases, keeping other parameters constant.

Fig. 9 depicts data communication cost (relative to the CG method). As expected, the advantage of our techniques in this aspect of the cost is maintained over the CG method—in fact, there is slow improvement. In fact, the data cost of the lazy synchronization over CG for WorldCup on streams of 20 sites is 4 times less.

Unfortunately, this does not render our techniques scalable, because as the number of sites grows, communication overhead becomes dominant. Fig. 10 depicts the total y(normalized) communication cost. It can be seen that the CG method maintains a 50% saving over the cost of the naive method. In our technique however, communication overhead dominates, to the extent that the total cost becomes 2–4.5 times higher than the cost of the naive method! Notice that the lazy method exhibits again about twice the overhead of the eager method.

The source of the problem seems to be in protocols of the geometric method itself (which we have not adapted in any way in this paper). As the number of sites increases, opportunities for rebalancing among sites increase commensurably. The protocols of the geometric method exhaust every opportunity to ensure that a global violation (triggering flushes) does not occur, without regard for the cost incurred by this rebalancing.

These results indicate that our techniques are only applicable beneficially to applications with a modest number of sites (up to 7). They also indicate an important direction for further research, namely, attempt to capture (at least some of) the benefit in data communication with a scalable rebalancing approach.

## 5. CONCLUSIONS

The problem addressed in this paper is monitoring of massive, distributed streaming data. The recently proposed geometric method has been combined with AMS sketches towards reducing the communication cost of tracking complex aggregate queries over distributed streams with strict error bounds.

To reduce communication cost, we utilized AMS sketches, similarly to previous work, but in a novel way; we developed a novel geometric method of dynamic balancing of error between remote sites, improving summarization at the sites before stream data has to be transferred over the network. We showed how to treat three fundamental types of aggregate queries: self-join, range and 2-way join between streams. Finally, we presented extensive empirical results to validate our performance claims for our techniques and demonstrate their practical viability.

**Extensions and Future Work.** In this paper, we applied the standard geometric method, as it appears in the literature. The techniques we developed exhibit much improved performance compared to previous techniques (particularly that of [7]) but fail to scale performance-wise when the number of remote sites increases. A fruitful problem of future research will be to enhance the standard geometric method, adapting it to the particularities of sketch-based monitoring, in order to improve scalability. Another promising direction for extension is the adoption of dynamic predictive error models. This idea has been shown in [7] to be beneficial to data communication and may also prove useful in reducing the overhead of the geometric method. We also intend to combine our techniques with other types of sketches from the literature and extend their applicability to new types of queries.

**Acknowledgments.** This work was partially supported by the European Commission under ICT-FP7- LIFT-255951 (Local Inference in Massively Distributed Systems).

## 6. REFERENCES

[1] N. Alon, P. B. Gibbons, Y. Matias, and M. Szegedy. “Tracking Join and Self-Join Sizes in Limited Storage”. In *ACM PODS*, 1999.

[2] N. Alon, Y. Matias, and M. Szegedy. “The Space Complexity of Approximating the Frequency Moments”. In *ACM STOC*, 1996.

[3] B. Babcock and C. Olston. “Distributed Top-K Monitoring”. In *ACM SIGMOD*, 2003.

[4] M. Charikar, K. Chen, and M. Farach-Colton. “Finding Frequent Items in Data Streams”. In *ICALP*, 2002.

[5] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong. “Approximate Data Collection in Sensor Networks using Probabilistic Models”. In *IEEE ICDE*, 2006.

[6] G. Cormode and M. Garofalakis. Streaming in a connected world: querying and tracking distributed data streams. In *ACM SIGMOD*, 2007.

[7] G. Cormode and M. Garofalakis. “Approximate Continuous Querying of Distributed Streams”. *ACM TODS*, 33(2), 2008.

[8] G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi. “Holistic Aggregates in a Networked World: Distributed Tracking of Approximate Quantiles”. In *ACM SIGMOD*, 2005.

[9] G. Cormode, M. Garofalakis, and D. Sacharidis. “Fast Approximate Wavelet Tracking on Streams”. In *EDBT*, 2006.

[10] G. Cormode and S. Muthukrishnan. “What’s Hot and What’s Not: Tracking Most Frequent Items Dynamically”. In *ACM PODS*, 2003.

[11] G. Cormode and S. Muthukrishnan. “An improved data stream summary: The count-min sketch and its applications”. In *Jrnl. of Algorithms*, 55(1), 2005.

[12] C. Cranor, T. Johnson, O. Spatscheck, and V. Shkapenyuk. “Gigascope: A Stream Database for Network Applications”. In *ACM SIGMOD*, 2003.

[13] A. Das, S. Ganguly, M. Garofalakis, and R. Rastogi. “Distributed Set-Expression Cardinality Estimation”. In *VLDB*, 2004.

[14] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong. “Model-Driven Data Acquisition in Sensor Networks”. In *VLDB*, 2004.

[15] A. Dobra, M. Garofalakis, J. Gehrke, and R. Rastogi. “Processing Complex Aggregate Queries over Data Streams”. In *ACM SIGMOD*, 2002.

[16] S. Ganguly, M. Garofalakis, and R. Rastogi. “Processing Set Expressions over Continuous Update Streams”. In *ACM SIGMOD*, 2003.

[17] N. Giatrakos, A. Deligiannakis, M. Garofalakis, I. Sharfman, and A. Schuster. “Prediction-based Geometric Monitoring over Distributed Data Streams”. In *ACM SIGMOD*, 2012.

[18] P. B. Gibbons. “Distinct Sampling for Highly-Accurate Answers to Distinct Values Queries and Event Reports”. In *VLDB*, 2001.

[19] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. “How to Summarize the Universe: Dynamic Maintenance of Quantiles”. In *VLDB*, 2002.

[20] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. “One-pass wavelet decomposition of data streams”. *IEEE TKDE*, 15(3), 2003.

[21] M. B. Greenwald and S. Khanna. “Space-Efficient Online Computation of Quantile Summaries”. In *ACM SIGMOD*, 2001.

[22] M. B. Greenwald and S. Khanna. “Power-Conserving Computation of Order-Statistics over Sensor Networks”. In *ACM PODS*, 2004.

[23] A. Jain, E. Y. Chang, and Y.-F. Wang. “Adaptive stream resource management using Kalman Filters”. In *ACM SIGMOD*, 2004.

[24] R. Keralapura, G. Cormode, and J. Ramamirtham. “Communication-efficient distributed monitoring of thresholded counts”. In *ACM SIGMOD*, 2006.

[25] D. Keren, I. Sharfman, A. Schuster, and A. Livne. “Shape-Sensitive Geometric Monitoring”. *IEEE TKDE*, 24(8), 2012.

[26] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. “The Design of an Acquisitional Query Processor for Sensor Networks”. In *ACM SIGMOD*, 2003.

[27] A. Manjhi, V. Shkapenyuk, K. Dhamdhere, and C. Olston. “Finding (Recently) Frequent Items in Distributed Data Streams”. In *IEEE ICDE*, 2005.

[28] G. S. Manku and R. Motwani. “Approximate Frequency Counts over Data Streams”. In *VLDB*, 2002.

[29] NII Shonan Workshop on Large-Scale Distributed Computation, Shonan Village, Japan, January 2012. `http://www.nii.ac.jp/shonan/seminar011/`.

[30] C. Olston, J. Jiang, and J. Widom. “Adaptive Filters for Continuous Queries over Distributed Data Streams”. In *ACM SIGMOD*, 2003.

[31] I. Sharfman, A. Schuster, and D. Keren. “A geometric approach to monitoring threshold functions over distributed data streams”. In *ACM SIGMOD*, 2006.

[32] N. Thaper, S. Guha, P. Indyk, and N. Koudas. “Dynamic Multidimensional Histograms”. In *ACM SIGMOD*, 2002.