Monitoring Distributed Streams using Convex Decompositions

Arnon Lazerson[1] **, Izchak Sharfman**[1]**, Daniel Keren**[2]**, Assaf Schuster**[1]**, Minos Garofalakis**[3] **and Vasilis Samoladas**[3]

[1]Faculty of Computer Science, Israeli Institute of Technology; [2] Department of Computer Science, Haifa University;

[3] School of Electronic and Computer Engineering, Technical University of Crete.

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 monitored condition across sites. The recently proposed geometric method, based on the notion of covering spheres, offers a generic methodology for splitting an arbitrary (non-linear) global condition into a collection of local site constraints, and has been applied to massive distributed stream-monitoring tasks, achieving state-of-the-art performance. In this paper, we present a far more general geometric approach, based on the *convex decomposition* of an appropriate subset of the domain of the monitoring query, and formally prove that it is *always* guaranteed to perform at least as good as the covering spheres method. We analyze our approach and demonstrate its effectiveness for the important case of sketch-based approximate tracking for *norm, range-aggregate, and join-aggregate queries*, which have numerous applications in streaming data analysis. Experimental results on real-life data streams verify the superiority of our approach in practical settings, showing that it substantially outperforms the covering spheres method.

1. INTRODUCTION

A task of increasing importance is the online monitoring of queries over continuous data streams. In a growing number of applications, such as content distribution networks, collaborating mobile devices, and IP network monitoring tasks, a large amount of streaming transient data is sequentially produced. Typically, massive data sizes and rapid transmission rates make it infeasible to store and index all the data for future processing. Thus, algorithms that access each data item only once, and process it in a short amount of time, are desirable. Furthermore, the amount of memory used by the algorithm is required to be sub-linear in the size of the input stream(s). Algorithms adhering to these strict requirements are often referred to as *streaming algorithms*. A fundamental query task in this setting is that of monitoring "threshold crossings": Given a function $f()$ and a threshold T, the query is defined by "is $f(v) \leq T$?", where v is a (large) dynamic vector that captures the current state (e.g., frequency distribution) of the streaming data. This query model has been used in numerous applications [13], either directly or as the main building block for other queries such as top-k and "heavy-hitter" items, quantiles, and so on [26]. Furthermore, it can be naturally extended to the more general problem of *approximate function monitoring*, where the goal is to track the value of a function $f(v)$ to within user-prescribed error bounds [11]. Given the streaming nature of the data, we are, of course, interested in continuous (standing) queries; that is, the query is "always there" and an alert must be issued whenever $f(v) > T$.

The difficulty in processing high-volume streams is compounded by the fact that in many real-life situations, local streams are generated at *physically distributed* nodes, and we are interested in monitoring a function over the global union of the streams, while minimizing communication across the network. In other words, each node holds a dynamic local data vector, the global streaming vector v is defined as the average (or, sum) of these local vectors, and the query is defined using the function value $f(v)$. Clearly, the naive solution of aggregating the local streams to a central location and processing them there is infeasible, as it incurs a huge communication overhead. It is therefore desirable to map the global, network-wide query to *local* queries/constraints, which can be independently checked at the nodes. To ensure correctness, these local queries should be *"safe"*, meaning that as long as they all hold (no "local violations"), the global query must also hold.

Prior Work. Earlier work on monitoring distributed streams via local constraints addressed the simpler cases of thresholding *linear* functions [17, 16], top-k monitoring [4], and ratio queries [14]. In [21] local conditions were placed on one-dimensional variables for monitoring the value of a polynomial. In [15], perturbative analysis of eigenvalues was applied to determine local conditions on data volume matrices at the nodes of a distributed system, in order to monitor system health. Monitoring entropy and related functions was studied in [3], and a theoretical study of the monitoring problem is provided in [9]. Decision trees over a large distributed network were handled in [5]. In [25] an algorithm is provided for determining whether the norm of the average vector in a distributed system is below a certain threshold. Other work also addressed distributed monitoring of monotonic functions [20], and state monitoring via local constraints [19]. A major breakthrough for streaming algorithms was the introduction of *sketches* [2], which constitute efficient data-summarization tools allowing the small-space approximation of inner product and norm queries with pre-set bounds on

Proceedings of the VLDB Endowment, Vol. 8, No. 5

the error magnitude and probability. In [8] sketches were employed for communication-efficient, approximate monitoring of join aggregates over distributed streams.

All these earlier papers focus solely on a specific class of distributed-monitoring queries, resulting in special-purpose techniques applicable only to the type of queries at hand. Recently, [22, 23] have proposed a general, *geometric monitoring* approach for efficiently tracking the value of a *general function/query* over distributed data relative to a given threshold. Their solution relies on an interesting geometric argument, that employs the methodology of *Covering Spheres (CS)* for breaking up a global threshold condition on a function into "safe" local conditions that can be checked locally at each site. Followup work [18] has proposed various extensions to the CS method (e.g., using ellipsoids rather than spheres), and has introduced a more general notion of *Safe Zones (SZs)* for distributed monitoring. The CS method has very recently been applied to the problem of efficient outlier detection in sensor networks [7], as well as for sketch-based approximate monitoring of norm, range-aggregate, and join-aggregate queries over distributed streams [11], achieving the current state-of-the-art results in both settings. CS was also applied in a predictive model [12] and to monitoring system health [10].

Our Contributions. In this paper, we propose a novel method for monitoring over distributed streams, which is also rooted in geometry. We first lay its theoretical basis and prove that it *always* performs at least as good as the CS method, meaning that if CS does not experience a local violation, neither does the proposed algorithm. Then, we apply it to the above-mentioned queries, achieving a substantial improvement over the state-of-the-art. Our contributions can be summarized as follows.

• We introduce a novel method based on the *Convex Decomposition (CD)* of an appropriate subset of the query domain for monitoring threshold-crossing queries over distributed streams. Our CD method is *provably better* (in a sense to be made precise later) than the covering spheres method. A notable difference is that CD works by identifying convex subsets of the *inadmissible region*, as opposed to previous work which sought a convex subset of the *admissible region*.

• We detail the application of our CD methodology to the important tasks of sketch-based approximate monitoring for norm, range-aggregate, and join-aggregate queries. Once again, our analysis clearly demonstrates the effectiveness of CD compared to the state-of-the-art CS techniques for these problems [11]. We chose to apply the CD method to these functions due to their great practical importance, and also due to their complex nature (they are not linear, monotonic of convex), which defies most monitoring methods. Also, it allowed us to "head on" compare performance with the state-of-the-art.

• We carry out an experimental evaluation with real-life data streams that verifies the superiority of our CD method in practical settings, showing that it substantially outperforms the CS technique.

2. PRELIMINARIES

Problem Setup. We consider a distributed-computing environment, comprising a collection of m remote nodes and a designated coordinator node. Streams of data updates arrive continuously at remote nodes, while the coordinator is responsible for continuously tracking user queries posed over the *union* of remotely-observed streams (across all nodes). Each remote node $j \in \{1, \ldots, m\}$ observes a local update stream that incrementally render a dynamic *local statistics vector* v_j capturing the current local state (e.g., frequency distribution) of the observed stream(s) at node 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 v_j has 2×2^{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 v_j could be used for `TCP` (respectively, `UDP`) packet frequencies.) We define the *global statistics vector* v of our distributed stream(s) as the *average* of the local statistics vectors $\{v_j\}$; that is, $v = \frac{v_1+\ldots+v_m}{m}$.

We are interested in effectively tracking user queries over the global statistics vector at the coordinator. More specifically, our focus is on the fundamental problem of monitoring *distributed threshold-crossing queries*; that is, determine whether $f(v) \leq T$, for a given (general) function $f()$ over the global statistics vector and a fixed threshold T. Threshold-crossing queries form a key building block for several classes of distributed stream monitoring queries (e.g., top-k, heavy-hitters, quantiles) [26], and the more general problem of distributed approximate function tracking [11].

Convexity. Convex sets are closed under averaging, and the global statistics vector is defined as the average of the local vectors. Therefore, convexity plays a crucial role in the proposed CD method. We remind of some relevant definitions:

- A *convex combination* of vectors $\{v_i\}$ is a vector of the form $\sum_i \lambda_i v_i$, where λ_i are scalars satisfying $\lambda_i \geq 0, \sum_i \lambda_i = 1$.
- The *convex hull* of vectors $\{v_i\}$ is the set consisting of all the convex combinations of $\{v_i\}$.
- A set is *convex* if it equals its convex hull.

For example, the convex hull of a finite set of points in the plane is the smallest (w.r.t inclusion) convex polygon which contains the points. We note here that both CS and CD methods can handle the case in which the global statistics vector is not necessarily the average, but *any* convex combination, of the local vectors.

The Covering Spheres (CS) Method. We now describe some basic notions of the CS method, which will be used as a baseline to compare with the CD method; along the way, we will also introduce some definitions which will be used in Section 3.

As observed by Sharfman et al. [22, 23], it is generally impossible to relate the locally-observed values of $f(v_j)$ to the global value $f(v)$; instead, their key idea is to employ geometric arguments to monitor the *domain* (rather than the range) of the monitored function $f()$. More formally, given the threshold query $f(v) \leq T$, define the set $A \triangleq \{v | f(v) \leq T\}$ as the query's **admissible region**. Clearly, the condition $f(v) \leq T$ is equivalent to the condition $v \in A$, and this *geometric* condition is the one being monitored.

Running example – lower bound of norm. Assume that the condition being monitored is over a function of two variables x, y, and is defined by $x^2 + y^2 \geq 1$. Then A is the complement of the unit disk in the plane $\mathcal{R}^2$.

Assume that at some point in time, each node j has informed the coordinator of some initial/prior state of its local vector $v_j^{\textbf{init}}$; thus, the coordinator has an estimated global vector $p \triangleq \sum_{j=1}^{m} v_j^{\textbf{init}}/m$, also referred to as the **reference point**. Clearly, the updates arriving at nodes can cause the local vectors v_j to drift too far from their previously reported values $v_j^{\textbf{init}}$, possibly leading to a violation of the T threshold. Let $d_j \triangleq v_j - v_j^{\textbf{init}}$ denote the local *drift vector*

(due to updates) at node j. We can then express the current global statistics vector v in terms of the drift vectors as follows:

$$v = \frac{\sum_{j=1}^{m}(v_j^{\text{init}} + d_j)}{m} = p + \frac{\sum_{j=1}^{m} d_j}{m} = \frac{\sum_{j=1}^{m}(p + d_j)}{m}.$$

That is, the current global vector is the average of the (translated) drift vectors $p + d_j$, $j = 1, \ldots, m$, and, thus, guaranteed to lie somewhere within the convex hull of the drift vectors around p. Thus, as long as the convex hull does not overlap the inadmissible region $\overline{A} = \{v : f(v) > T\}$, we can guarantee that the threshold has not been violated (i.e., $f(v) \leq T$).

The problem, of course, is that the d_j's are spread across the sites and, thus, the above condition on the convex hull cannot be checked locally. This global condition can be transformed into local geometric constraints with the help of the *bounding lemma* (formally shown in [23]) which states that, in any dimensionality, the convex hull of a set of points x_i is contained in the union of spheres whose diameters are the segments $\overline{x_i, x}$, for any point x.

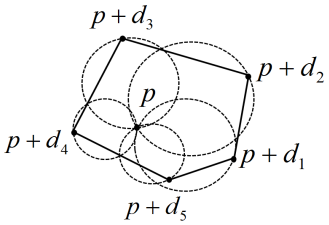

Figure 1: The bounding lemma: The union of the spheres with diameters $\overline{p, p + d_j}$ contains the convex hull of $\{p + d_j\}$.

Thus, since the global statistics vector v lies inside the convex hull of the vectors $\overline{p+d_j}$, $j = 1, \ldots m$ and the union of the spheres with diameters $\overline{p, p + d_j}$ completely covers this convex hull (Fig. 1), the problem of monitoring v has been effectively reduced to the local problem of each remote node monitoring the sphere around its local drift vector: If at some point a node's local sphere steps into the inadmissible region $\overline{A}$, then we have a *local violation*, and the node communicates its local drift d_j to the coordinator. The coordinator then initiates a *synchronization process* that typically tries to resolve the local violation by "balancing out" the violating drift with data from other nodes [22, 23].

Geometric Monitoring using Convex Safe Zones. In followup work, Keren et al. [18] propose a simple, generic geometric monitoring strategy that can be formally shown to encompass the CS method as a special case. Briefly, their strategy relies on defining a certain *convex subset* C of the admissible region A (i.e., a *convex admissible subset*), which is then used to define *Safe Zones (SZs)* for the (translated) local drift vectors: *Node j simply monitors the condition* $p + d_j \in C$. The correctness of this generic monitoring scheme follows directly from the convexity of C, and our earlier observation that the global statistics vector v always lies in the convex hull of $p + d_j$, $j = 1, \ldots, m$: If $p + d_j \in C$ for all nodes j then, by convexity, this convex hull (and, therefore v) lies completely within C and, therefore, the admissible region (since $C \subseteq A$). (Note that the convexity of C plays a crucial role in the above correctness argument.) Fig. 2 depicts some of the key concepts in geometric monitoring using a convex admissible subset C.

Note that, while the convexity of C is needed for the *correctness* of the monitoring scheme, it is clear that the size of C plays

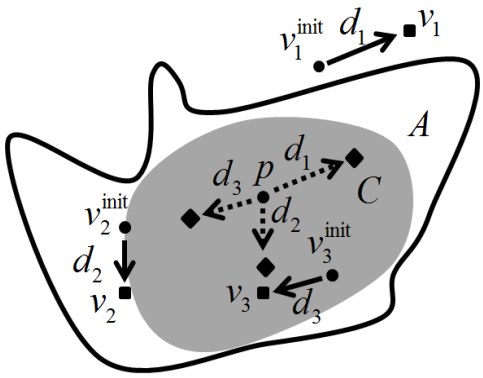

Figure 2: Geometric monitoring using a convex admissible subset C for three nodes. The reference point, p, is the average of the three initial data vectors at the nodes, v_j^{init}, marked by circles (note that the average of the initial vectors has to be in C, but not the individual v_i^{init}). The current values of the data vectors at the nodes, v_j, based on the local drift vectors d_j. Node j continually checks whether the vector $p + d_j$ (marked by diamonds) is inside C; as long as this local condition holds $v \in C \subseteq A$ and no action is needed.

a critical role in its *efficiency*: Obviously, a larger C implies fewer local violations and, thus, smaller communication/synchronization overheads. This, in turn, implies a fairly obvious dominance relationship over geometric distributed monitoring schemes: Given two geometric algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ (for the same distributed monitoring problem) that use the convex admissible subsets C_1 and C_2 (respectively), algorithm $\mathcal{A}_1$ is *provably superior* to $\mathcal{A}_2$ if $C_2 \subset C_1$. Note that, in the simple case of *linear* functions $f()$, the admissible region A itself is convex, and therefore one can choose $C = A$. However, for more complicated, non-linear functions, as the ones treated here, A is non-convex and quite complex. Thus, finding a "large" convex subset of A is a crucial component of geometric monitoring, and a main focal point of our work.[1]

Interestingly, the CS method can also be cast in terms of a convex admissible subset (denoted by C_S) that is implicitly defined as follows:

DEFINITION 1. *The convex admissible subset C_S defined by the CS method (with reference point p) is the set of all points q for which the sphere with diameter $\overline{p, q}$ is contained in A.*

In our running example, where A is the complement of the unit disk, the condition for a point q to be in C_S is that the disk whose diameter is the segment $\overline{pq}$ does not intersect the unit disk – see Fig. 3:

Since the disk's center is at $(p+q)/2$ and its radius is $\|p-q\|/2$, the condition can be written as

$$\|p + q\|/2 \geq 1 + \|p - q\|/2 \qquad (1)$$

After some simple manipulations, this condition can be expressed as a quartic (i.e. fourth-degree) inequality in q's coordinates, hence the safe-zone's boundary is a quartic curve.

As discussed in [18], C_S can be defined in a more direct manner using half-space intersections. Consider a simple case, in which the admissible region A equals the entire plane except for one point r

[1]Since C is tested for containment of $p + d_j$, its choice should depend both on A and p; intuitively, it should be both "large" and p should be far from its boundary.

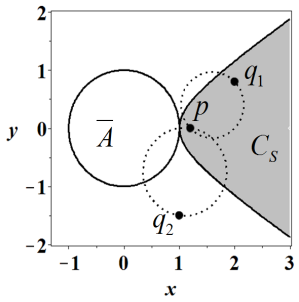

Figure 3: An example of the construction of C_S, the convex admissible subset used in the CS method, for the running example in which A is the complement of a disk. C_S (in grey) comprises all points q such that the sphere with diameter $\overline{p,q}$ is contained in A. Thus, $q_1 \in C_S$ but $q_2 \notin C_S$.

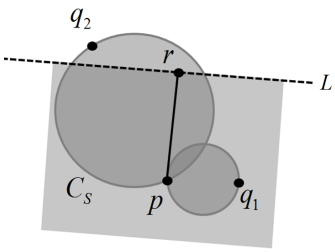

Figure 4: The set C_S for reference point p when $\overline{A} = \{r\}$ is exactly the half-space depicted in gray: A sphere with diameter $\overline{p,q}$ does not contain r if and only if q is in the half-space supported by L (This holds for any dimensionality.)

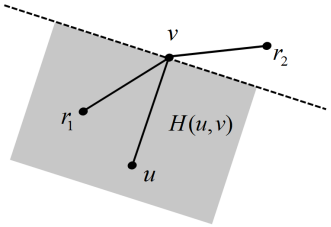

Figure 5: The construct $H(u,v)$. Note that $\langle r_1 - v, u - v\rangle > 0$ and $\langle r_2 - v, u - v\rangle < 0$.

(e.g., if we are monitoring the condition $||v - r||^2 > 0$). It is easy to see that in this case, C_S is the half-space depicted in Fig. 4. For notational convenience, given two vectors u, v, define $H(u,v)$ to be the half-space supported[2] by the hyperplane perpendicular to the segment $\overline{u,v}$, passing through v and containing u. Thus, in Fig. 4, $C_S = H(p,r)$. Note that the construct is not symmetric, i.e., $H(u,v) \neq H(v,u)$; furthermore, $H(u,v) = \{x|\angle(xvu) \leq 90^\circ\} = \{x|\langle x - v, u - v\rangle \geq 0\}$, where $\langle x,y\rangle = \sum x_i y_i$ denotes the *inner product* of vectors x and y (Fig. 5).

In the case of general $\overline{A}$, in order to obtain C_S, *every point* in $\overline{A}$ has to be excluded in a similar manner. This leads to the following geometric characterization of C_S.

THEOREM 1. *([18]). $C_S = \bigcap_{r \in \overline{A}} H(p,r)$. That is, C_S equals the intersection of the half-spaces $H(p,r)$ for all r in the inadmissible region $\overline{A}$.*

An easy corollary shows that it is, in fact, sufficient to just intersect the half-spaces $H(p,r)$ for all r *at the boundary of* $\overline{A}$ (denoted by $\partial\overline{A}$); that is, $C_S = \cap_{r \in \partial\overline{A}} H(p,r)$.[3] To recap, the construction of C_S can be viewed as follows: every point r in the boundary of $\overline{A}$ is "separated from p" by the hyperplane passing through r and which is perpendicular to the segment $\overline{p,r}$. In Fig. 6 this process is depicted for two points r_1, r_2, with the "separating hyperplanes" denoted by $S(r_1), S(r_2)$. C_S is equal to the intersection of *all such half-spaces* $H(p,r)$. Convexity of C_S follows since the individual half-spaces are convex, and the intersection of any family of convex sets is also convex.

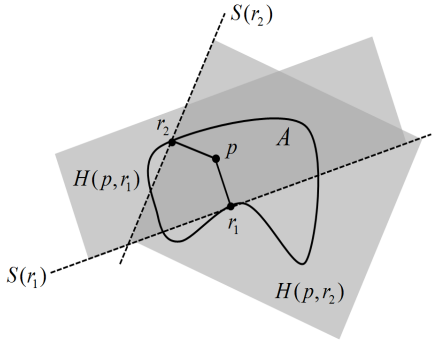

Figure 6: An equivalent construction of C_S by intersecting half-spaces, depicted for two points, r_1, r_2 on $\overline{A}$'s boundary. C_S is obtained by intersecting all such half-spaces.

The key to obtaining a better set, which contains C_S, is to intersect *less* half-spaces, thus obtaining a larger set. The method to achieve this, which is next described, also relies heavily on the notion of convexity.

3. THE CONVEX DECOMPOSITION (CD) METHOD

The CS method generally achieves good results and is generic, i.e., it can be applied to any function defined over the average of the local vectors at the nodes. The question that naturally arises has to do with the performance of the CS method and the SZs defined by its convex admissible subset C_S: In this section, we demonstrate how C_S can be drastically improved in some cases, by intersecting *much fewer half-spaces* (Theorem 1) in order to obtain a *larger* convex admissible subset. For instance, when $\overline{A}$ is *convex*, the process can be improved by separating $\overline{A}$ from p by a *single* hyperplane, yielding a convex subset of A which contains C_S, and is also simpler to define[4]. This provides the motivation for first considering the case of convex $\overline{A}$; then, we consider the more general case when $\overline{A}$ is a union of convex sets.

Let us start with a simple case – our running example (Section 2), where $\overline{A}$ is the unit disk. C_S is constructed according to the method described in Theorem 1, depicted in Fig. 7 for $p = (a, 0)$. Three of the half-planes whose intersection equals C_S are shown – $H(p,r_i), i = 1,2,3$. Note that r_1 is on the x-axis. Evidently, while

[2]A half-space will be said to be *supported* by its boundary, which is of dimension smaller by 1 than that of the ambient space.

[3]It is not difficult to see that, for any $r' \in \overline{A} - \partial\overline{A}$ there exists an $r \in \partial\overline{A}$ (essentially the point where $\overline{p,r'}$ intersects the boundary) such that $H(p,r) \subseteq H(p,r')$.

[4]There is an interesting geometric *duality* here – it is easy to define C when either A or $\overline{A}$ is convex.

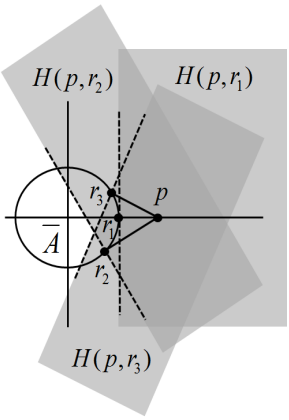

Figure 7: Construction of C_S for the running example ($\overline{A}$ is the unit disk). Three of the half-spaces whose intersection equals C_S are depicted.

the set C_S is *correct*, it can be improved. This follows since the half-plane $H(p, r_1)$ of Fig. 7 strictly contains it, and also satisfies the necessary convexity property.

Why does this happen? The answer in this simple case is clear from Fig. 7: after separating r_1 from p, the *entire* set $\overline{A}$ is separated from p, and there is therefore no need to intersect with $H(p, r_2)$, $H(p, r_3)$ etc. Further, intersecting with these half-spaces obviously *reduces* the quality of the resulting C, as it yields a set strictly contained in $H(p, r_1)$.

This simple observation can easily be extended to the case where $\overline{A}$ is *any* convex set, with the point r_1 in Fig. 7 replaced by the point in $\overline{A}$ which is closest to p (denoted $\overline{A}(p)$). We remind of the following well-known theorem:

THEOREM 2. *For any convex and closed set S and a point p such that $p \notin S$, there is a unique point in S, hereafter denoted $S^C(p)$, such that $||S^C(p) - p|| = \inf_{s \in S} ||s - p||$, that is, the minimal distance from S to p is obtained at $S^C(p)$*[5]*. Further, $H(p, S^C(p)) \cap S = \emptyset$.*

We naturally extend our half-space construct $H(p, r)$ to $H(p, S) \triangleq H(p, S^C(p))$, valid for a convex set S such that $p \notin S$. Clearly, if $\overline{A}$ is convex, then $H(p, \overline{A})$ contains C_S, since C_S is the intersection of *all* half-spaces $H(p, r), r \in \overline{A}$, and $H(p, \overline{A})$ is just one of these half-spaces.

The advantage of using $H(p, \overline{A})$ instead of C_S is clear: not only does $H(p, \overline{A})$ contain C_S, it is also simpler to define and to apply during query monitoring, since it is trivial to check whether a point belongs to a half-space (e.g., using the condition on the vector inner product, see Fig. 5).

It is, of course, desirable that the boundary of C be far from p; else, small drift vectors may cause a violation. The distance of p from the boundary of any convex admissible subset C is naturally defined as the minimum distance of p from any boundary point of C, and this is clearly upper-bounded by $||p - \overline{A}^C(p)||$. In that sense, the above construct satisfies a certain optimality criterion – it is *maximal* with respect to all C's which attain this distance upper bound. This is formalized in the following theorem (proof deferred to the full paper).

[5]There is a lot of work on efficiently computing $S^C(p)$, e.g., [6].

THEOREM 3. *Let C be any convex set such that $C \subseteq A$, $p \in C$, and dist$(p, \partial C) = ||p - \overline{A}^C(p)||$ (∂C is the boundary of C). Then, $C \subseteq H(p, \overline{A})$.*

So far, we have shown how to improve over C_S when $\overline{A}$ is convex. We next turn to the more general case.

3.1 The Case $\overline{A}$ = Union of Convex Sets

Assume that $\overline{A}$ is not convex, but that it can be represented as $\overline{A} = \bigcup_{i=1}^{N} S_i$, for convex sets S_i (S_i need not be disjoint). We call such a representation a *convex decomposition* (CD) of $\overline{A}$. Following as in the case in which $\overline{A}$ is convex, we may define C by $\bigcap_{i=1}^{N} H(p, S_i)$. This choice of C is correct (it is convex and $C \bigcap \overline{A} = \emptyset$). Further, just as in the simpler case in which $\overline{A}$ is convex, it is clear that the C thus constructed contains C_S, since C is equal to the intersection of a subset of the half-spaces whose intersection defines C_S (recall that C_S equals the intersection of all $H(p, r)$ for $r \in \overline{A}$, and the C defined using the convex decomposition of $\overline{A}$ is defined by intersecting only $H(p, S_i)$, a smaller set). We note that exactly the same construction can be applied if the convex decomposition contains an *infinite* number of sets (the case of join aggregates in Section 7 is such an example).

However, this construction can sometimes be improved. For example, assume that for some i, j, $H(p, S_i) \cap S_j = \emptyset$. Then $H(p, S_i)$ already separates S_j from p, and there is no need to intersect with $H(p, S_j)$; see Fig. 8. Note that this was also the case with our running example (Fig. 7 and discussion therein).

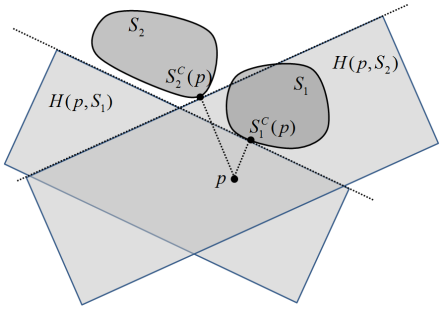

Figure 8: If $\overline{A}$ is the union of the two convex sets, S_1 and S_2, and $H(p, S_1)$ separates p from S_2, there is no need to intersect with $H(p, S_2)$, and C can be defined as $H(p, S_1)$.

To avoid cases as in Fig. 8, we propose the following definition:

DEFINITION 2. $\overline{A} = \bigcup_{i=1}^{N} S_i$ *is called* non-redundant *with respect to reference point p if, for all $1 \leq i \leq N$,*

$S_i \bigcap \left(\bigcap_{j \neq i} H(p, S_j) \right) \neq \emptyset$*; otherwise, it is called* redundant.

In other words, a cover of $\overline{A}$ by convex subsets S_i is non-redundant if no S_i is separated from p by the separating half-spaces of the other $S_j, j \neq i$. Note that this definition is with respect to a reference point p, and a cover which is non-redundant for one reference point can be redundant for a different reference point.

CD as a Generalization of CS. From Theorem 1, it immediately follows that CS is a special case of CD when the convex cover of

$\overline{A}$ is the trivial one, which consists of all the singletons comprising it; typically, this cover is highly redundant, which allows for a potentially dramatic improvement of CD over CS.

4. THE MONITORED FUNCTIONS

In this section, we discuss the specific monitoring functions involved in effectively tracking approximate norm and range-aggregate queries over distributed streams. (The more complex case of join aggregates is discussed in Section 7.) Given the global frequency distribution vector v, the function $f(v) = \langle v, v\rangle = ||v||^2 = \sum v_i^2$ denotes a (squared) L^2*-norm query* (also known as the "self-join size") of v – it can be used to derive important statistical information about the distribution, such as its variance and degree of skew. A *range query* $f(v) = \langle v, v_0\rangle$ counts the total appearances of some of the items by computing the inner-product of the global frequency vector v with a (constant) vector v_0 consisting of 1's at the locations corresponding to these items, and 0's elsewhere. Such inner-product operations (with constant vectors) can also be used to compute v's coefficients in some orthonormal basis, such as discrete cosine or wavelets.

The ability to express many important queries (e.g., heavy-hitters and data skew [2]) over the global frequency distribution by its inner-product with some fixed vector, or by its norm, has led to a great deal of research on memory and communication efficient algorithms for computing or approximating inner products and norms in a dynamic setting. A major breakthrough was the introduction of *sketches* [2], which constitute efficient data-summarization tools that allow to approximate the sought functions with pre-set bounds on the error magnitude and probability. A brief survey of the *AMS sketches* we apply here is provided in Section 4.1. Note that the monitoring is performed not on the original data, but on its sketch (i.e., in *sketch space*). Applying sketches reduces data volume. However, while range and norm functions are respectively linear and quadratic in the original data, their approximations in sketch space are quite more complicated. We next describe these approximations, and then show how to apply the CD method to monitor them.

4.1 Approximation by Sketches

The sketches applied here are defined by a $k \times l$ collection of four-wise independent random vectors, denoted r_{ij}, of the same length as v and v_0, and whose components are uniformly distributed in $\{-1, +1\}$. These vectors are generated on the fly, and need not be stored[6]. $||v||^2$ is then approximated as follows:

1. Construct a $k \times l$ matrix M (i.e. k rows, l columns), $M_{ij} = \langle v, r_{ij}\rangle^2$ (the inner products are also computed on the fly).
2. The approximate value of $||v||^2$ is defined as $\mathrm{med}_i\, ||M_i||^2$, where med is the median and M_i the i-th row of M.

In [2] it is proved that with probability $1 - \delta$, this estimate approximates $||v||^2$ by a relative error smaller than ϵ, where $k = O\left(\log\left(\frac{1}{\delta}\right)\right)$ and $l = O\left(\frac{1}{\epsilon^2}\right)$ (that is, with probability $1 - \delta$, the approximation to $||v||^2$ falls within $\epsilon||v||^2$ of the actual value).

Estimating the inner product with v_0 proceeds similarly, with $M_{ij} = \langle v_0, r_{ij}\rangle\langle v, r_{ij}\rangle$. Then, the inner product of v and v_0 is approximated by the median of the sums of M's rows. Similar approximation guarantees hold as for the case of norm queries, with the error relative to $||v|| \cdot ||v_0||$.

Since sketching is applied prior to computing the queried functions, the monitoring is applied not on the frequency distribution vectors, but on the sketch matrices M. We denote the functions in sketch space as

$$f_I(M) = \mathrm{med}_i \sum_j M_{ij} \quad \textbf{(range query function)} \tag{2}$$

$$f_N(M) = \mathrm{med}_i\, ||M_i||^2 \quad \textbf{(norm query function)} \tag{3}$$

with a slight abuse of notation, we will also apply f_I to a vector, defining it as the median of its coordinates.

The distributed setting for $f_I(), f_N()$ is identical to the one defined in Section 2; see Fig. 9. Since the sketches are linear, the

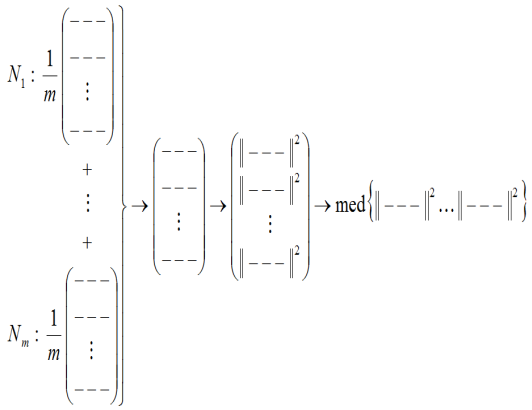

Figure 9: Computing f_N in the distributed setting. N_i are the distinct nodes, each of which holds a matrix. The matrices in the distinct nodes are first averaged, then the norm squared of each row in the average matrix is computed, and then a median is taken over the resulting values. For f_I the process is similar, with the norm squared operation replaced with an inner product with a fixed vector v_0.

global function is computed by averaging the local sketch matrices and applying a function on the average matrix. The introduction of sketches, while very useful for reducing data size, comes at a price: due to the application of the median operator, the functions f_I, f_N are quite more complicated than the inner product and norm functions – most notably, they are non-convex, hence the simple method which is applicable when A or $\overline{A}$ is convex (Section 3) cannot be used. Therefore, we apply the more general CD method, introduced in Section 3.1.

5. CD MONITORING IN SKETCH SPACE

Next we apply the CD method to f_I and f_N. Recall (Section 3.1) that, given a function f with an admissible region A, the first step is the representation of $\overline{A}$ as a (preferably non-redundant) union of convex sets. We now proceed to achieve this for our two sketch estimator functions.

5.1 f_I for vectors: median of components

Recall that for a matrix M, $f_I(M) = \mathrm{med}_i\, M_i$, where M_i equals the sum of entries of M's i-th row. We can therefore view $\{M_i\}_{i=1}^k$ as a vector of length k, where k is the number of matrix rows, and f_I as the median of its components. We proceed to first solve the problem for this vectorial version of f_I.

We assume, without loss of generality, that we monitor the condition $f_I(M) > 0$. If the condition is $f_I(M) < 0$, simply reverse all coordinates, solve for $f_I(M) > 0$, and reverse the coordinates again. To monitor $f_I(M) > b$ for some constant $b \neq 0$,

[6]Please see [2] for details; here we focus on the functional form of the sketches and the monitoring thereof.

switch to a new set of variables by subtracting b from the original ones, solve, and change back by adding b. Since the general case is somewhat technically involved, we start with $k = 3$. So, we are dealing with vectors (x_1, x_2, x_3), and the monitored query is $\text{med}\{x_1, x_2, x_3\} > 0$. (The general case is treated later in this section.)

We first note that A is non-convex. For example, it contains the vectors $(-11, 1, 1), (1, -11, 1), (1, 1, -11)$, but the average of these vectors is $(-3, -3, -3)$, which is not in A. Similarly, $\overline{A}$ is non-convex (note that $\overline{A} = \{-v | v \in A\}$). We next show that a non-redundant cover of $\overline{A}$ with three convex sets is provided by

$$\begin{aligned} \overline{A} &= S_1 \cup S_2 \cup S_3 \\ S_1 &= \{(x_1, x_2, x_3) | x_2, x_3 \leq 0\}, \\ S_2 &= \{(x_1, x_2, x_3) | x_1, x_3 \leq 0\}, \\ S_3 &= \{(x_1, x_2, x_3) | x_1, x_2 \leq 0\} \end{aligned} \tag{4}$$

Since for a point to be in $\overline{A}$ it has to have at least two negative coordinates, it is clear that S_1, S_2, S_3 indeed cover $\overline{A}$. Clearly S_1, S_2, S_3 are convex. We next compute the separating hyperplanes $H(p, S_i)$. Denote $p = (p_1, p_2, p_3)$ and assume for the while that all p_i's are > 0. It is trivial to verify that r_1, the point in S_1 closest to p, is $r_1 = (p_1, 0, 0)$. Therefore, the half-space $H(p, S_1)$ separating p from S_1 is defined by

$$\begin{aligned} &H(p, S_1) = \{x | \langle p - r_1, x - r_1 \rangle > 0\}, \\ &\text{or} \quad \{x | p_2 x_2 + p_3 x_3 > 0\}, \text{ for } x = (x_1, x_2, x_3) \end{aligned}$$

$H(p, S_2)$ and $H(p, S_3)$ are similarly defined.

LEMMA 1. *S_1, S_2, S_3 are a non-redundant cover of $\overline{A}$.*

PROOF. Let us show that $S_1 \cap H(p, S_2) \cap H(p, S_3) \neq \emptyset$ (the other cases follow similarly). Recall that $H(p, S_2) = \{x | p_1 x_1 + p_3 x_3 > 0\}, H(p, S_3) = \{x | p_1 x_1 + p_2 x_2 > 0\}$, and $S_1 = \{(x_1, x_2, x_3) | x_2, x_3 \leq 0\}$. Since we assumed $p_i > 0$, it follows that $(p_1, 0, 0) \in S_1 \cap H(p, S_2) \cap H(p, S_3)$, hence it is non-empty. □

Since the cover is non-redundant, we define C to be the intersection of $H(p, S_i), i = 1, 2, 3$, i.e.

$$\begin{aligned} C = \{(x_1, x_2, x_3) | &p_1 x_1 + p_2 x_2 > 0, \\ &p_1 x_1 + p_3 x_3 > 0, p_2 x_2 + p_3 x_3 > 0\} \end{aligned} \tag{5}$$

In order to visually demonstrate the advantage of the CD method over CS, Figure 10 depicts the sets C defined by both methods for dimension 3, with the reference point $p = (1, 0.2, 0.1)$. Both the three-dimensional sets and a cross-section are depicted. The CD method clearly yields larger sets.

Generalization to Higher Dimensionality. For higher dimensions, the partitioning of $\overline{A}$ to convex sets S_i, as well as the construction of $H(p, S_i)$, proceeds similarly as in dimension 3. Assume the dimension is $2N + 1$, and again, assume without loss of generality that we are monitoring the condition $f_I(v) > 0$. Assume, for the time being, that all p's coordinates are > 0; continuing as for dimension 3, it is easy to verify that the following defines a non-redundant cover of $\overline{A}$: there are $\binom{2N+1}{N+1}$ S_i's in the cover, each defined by setting one of the subsets of coordinates $\{1, 2, ...2N + 1\}$ of size $N + 1$ to be negative. If the subset of coordinates is $\{i_1, i_2 ... i_{N+1}\}$, the corresponding half-space separating p from S_i is defined by $\sum_{j=N+1}^{j=1} p_{i_j} x_{i_j} > 0$. C is then defined as the intersection of all these half-spaces. The non-redundancy of the cover follows exactly as in the case of three dimensions.

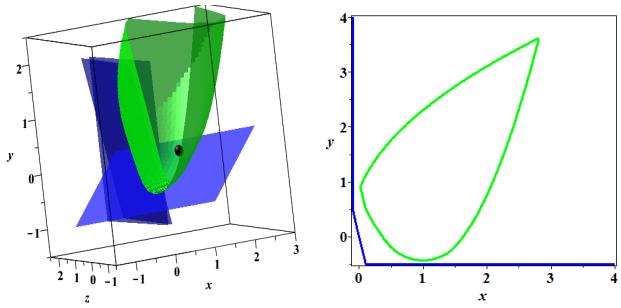

Figure 10: Comparison of C for CD vs. CS, for f_I in three dimensions, with $p = (1, 0.2, 0.1)$. Top left: the sets C in three dimensions. The CS set is bound by the green surface, and the CD one is bound by the three blue hyperplanes. p is the small dark sphere. Top right: cross-section at height $z = 1$ through both sets.

Recall that during the monitoring process, every node has to continually check whether $x = p + d \in C$, where p is the reference point and d the node's drift vector (Section 2). While the number of inequalities defining C is exponential in N, there is a simple way to reduce the complexity to $O(N)$: it suffices to sum the smallest $N + 1$ values from the $2N + 1$ values $p_i x_i$. Trivially, *all* the inequalities hold iff the sum of these values is > 0.

The Case of Negative p_i's. So far, we have assumed that all of p's coordinates are > 0. A slight modification is required when that is not the case. To avoid drowning in indices, we start with a special case, which suffices to explain the general one. Assume the dimension is 5, and we have $p_1 \leq 0, p_2, p_3, p_4, p_5 > 0$. Suppose we wish to separate $S = \{x_1, x_2, x_3 \leq 0\}$ from p. This proceeds almost exactly as before, but now the closest point to p in S is $(p_1, 0, 0, p_4, p_5)$ (note that the first coordinate is p_1, and not 0), and the equation of the respective half-space is not $p_1 x_1 + p_2 x_2 + p_3 x_3 > 0$, but $p_2 x_2 + p_3 x_3 > 0$; the same happens whenever x_1 is one of the variables defining S. If, for example, $S = \{x_2, x_3, x_4 \leq 0\}$, then the intersection of S with the half-space $p_2 x_2 + p_3 x_3 > 0$ is empty, hence the cover will be redundant. Therefore, *only* the S_i's whose definition involves x_1 should be considered.

The general case follows exactly along these lines: if l of p's coordinates are negative for some $l < N + 1$, an analysis similar to the one above yields that the set C is defined as before, for all inequalities ranging over the subsets of size $N + 1 - l$ of the set of indices with positive coordinates.

For example, if $N = 3$ and $p_1, p_2 \leq 0$, $p_3, p_4, p_5, p_6, p_7 > 0$ (recall we have $2N + 1$ coordinates), then $l = 2$ and the inequalities defining C range over all subsets of size $N + 1 - l = 2$ of $\{3, 4, 5, 6, 7\}$, i.e. $p_3 x_3 + p_4 x_4 > 0$ etc. Intuitively this is because, since we already have two negative coordinates, we can allow only one more amongst $\{p_3, p_4, p_5, p_6, p_7\}$; and since the inequalities range over all subsets of size 2, they indeed exclude the possibility of two negative coordinates.

5.2 Monitoring f_I for Matrices

So far we dealt with the vectorial version of f_I, and constructed a convex subset of vectors, denote it C_{vector}, such that $v \in C_{\text{vector}} \rightarrow f_I(v) > 0$, where f_I is defined as the median of the vector's components. Since the goal is to monitor the matrix version of f_I, which is defined as the median of the sums of the rows of the matrix, we need to define a convex subset C_{matrix} of matrices such that $M \in C_{\text{matrix}} \rightarrow f_I(M) > 0$. This set is defined as follows: de-

fine a "collapsing function" $g : \mathcal{R}^{k\times l} \to \mathcal{R}^k$ from $k \times l$ matrices to vectors of length k, by mapping each matrix M to the vector whose i-th component is the sum of M's i-th row. C_{matrix} is then defined as $g^{-1}(C_{\text{vector}})$. Practically, this just means that in order to test whether a matrix $M \in C_{\text{matrix}}$, it suffices to test whether $g(M) \in C_{\text{vector}}$ (as described earlier). It still remains to prove that C_{matrix} is convex: this follows from the fact that $g()$ is linear, and the inverse image of a convex set under a linear function is always convex. The case for f_N is more complicated; we proceed to study it next.

5.3 Monitoring f_N for Matrices

Recall that f_N is computed for a matrix M by constructing a vector whose i-th coordinate is the norm squared of M's i-th row, and then computing the median of this vector's coordinates. That is, we are no longer dealing with the median of *linear* functions, but *quadratic* ones. To emphasize the difference between the two cases, let us look first at an auxiliary function which is similar but somewhat simpler – the vectorial version of f_N, defined as the median of the *squared* coordinates of an input vector (we still denote it f_N): $f_N(x_1, ...x_n) = \text{med}\{x_1^2, ...x_n^2\}$. Let us start with the upper bound case: following the definition of the safe zone by linear inequalities, which was developed in Section 5.1, the safe zone for $f_N()$ is defined by the intersection of sets defined by inequalities of the form $\sum_i p_i x_i^2 \leq T$, where the p_i's are ≥ 0 and T a positive threshold. Note that each such set is convex (it is just an ellipsoid), and since the intersection of any family of convex sets is convex, the safe zone thus defined is also convex.

However, the case for the *lower bound* is different: while sets defined by linear inequalities are always convex, a set of the form $\sum_i p_i x_i^2 \geq T$ is *not* convex (it is the complement of an ellipsoid). Note that this is the same problem discussed in Section 3: the admissible region, being the complement of a convex quadratic set, is non-convex. However, this problem can be remedied exactly as in Section 3, by separating the convex quadratic from the reference point with a hyperplane.

Let us now return to the matrix version of $f_N(M)$ (Eq. 3). Suppose we wish to monitor the upper bound condition, $f_N() < b$ for some constant b. As for the vectorial version of $f_N()$, directly substituting the norms squared of the rows into the inequalities defining C_{vector} (Section 5.1), yields inequalities of the form $\sum_{s\in\mathcal{S}} p_s ||M_s||^2 < b$ for some subset of row indices $\mathcal{S}$, and where the p_s are the norms squared of the rows of the reference point (matrix), hence are positive. Exactly as for the vectorial version of $f_N()$, the set satisfying all these inequalities is convex (recall that the norm squared of a row is just a sum of the squares of the row's elements, hence the individual inequalities still define ellipsoids in matrix space).

To handle the lower bound case, we proceed as follows: the admissible region A is defined as the set of matrices M such that $\text{med}_i\, ||M_i||^2 > b$, where M_i is M's i-th row. As for the case of vectors, $\overline{A}$ can be described as the following union of convex sets: if there are $2N+1$ rows, let $\{i_1, i_2...i_{N+1}\}$ range over all subsets of indices of size $N+1$, and for each such subset define the set $S(i_1, i_2...i_{N+1}) \triangleq \{M|\ ||M_{i_1}||^2, ||M_{i_2}||^2 ... ||M_{i_{N+1}}||^2 \leq b\}$. These sets are convex and obviously their union covers $\overline{A}$. It remains to construct the hyperplanes separating p from each such set; this requires finding the closest matrix to p in the set. This matrix, denoted as before $S(p)$, is constructed as follows: if $S(p)_j$ (p_j) denotes the j-th row of $S(p)$ (respectively, p), then

- If $j \notin \{i_1, i_2...i_{N+1}\}$, $S(p)_j = p_j$.
- If $j \in \{i_1, i_2...i_{N+1}\}$ and $||p_j||^2 \leq b$, $S(p)_j = p_j$.
- If $j \in \{i_1, i_2...i_{N+1}\}$ and $||p_j||^2 > b$, $S(p)_j = \sqrt{b}\frac{p_j}{||p_j||}$.

This follows since if $||v||^2 > B$, the vector whose norm squared is B and is closest to v is $\sqrt{B}\frac{v}{||v||}$.

As before, $S(i_1, i_2...i_{N+1})$ can be separated from p by a hyperplane perpendicular to the segment $\overline{p, S(p)}$, and the intersection of the respective half-spaces defines a convex subset of matrices M for which $f_I(M) > b$ (again, this can be viewed as a generalization of monitoring the condition $||v||^2 > b$ for a single vector, discussed in the beginning of Section 3).

Finally, we note that the same method used to expedite the testing of the condition $v \in C_{\text{vector}}$, for a vector v (Section 5.1) can be applied to the test whether a matrix belongs to C_{matrix}.

6. EXPERIMENTAL STUDY

The goal of our experiments is to quantify the effect of the improved safe zones obtained by the CD method on distributed stream monitoring performance. We tested the CD method on two real data sets for self-join queries (i.e., computing the size of the self-join of the stream, which is simply the squared norm of the frequency vector), and compared it to the CS method. We also compared CD to the method of *error vectors*, as applied in [11] which, to the best of our knowledge, presents the state-of-the-art results in monitoring self-join queries over distributed streams. We examined the *sliding window* scenario, in which one is interested in the records received within a certain period; for example, one may wish to monitor only over records that are less than 5 minutes old. Every record then has an expiration time, and whenever it expires the corresponding frequency count is *decreased* by one. Hence, the value of the norm in this scenario is fluctuating over time; see Fig. 11, which compares the value of the norm squared for sliding window with that of the *append only* stream (in which records are only added to the stream, hence the norm function increases monotonically). We note that the sliding window case corresponds to the turnstile model, in which the data vector's entries can both increase and decrease; in [11] only windows with a fixed starting point were used, which corresponds to the cash-register case (entries can only increase). We show that CD greatly outperforms the method in [11] for the sliding window case.

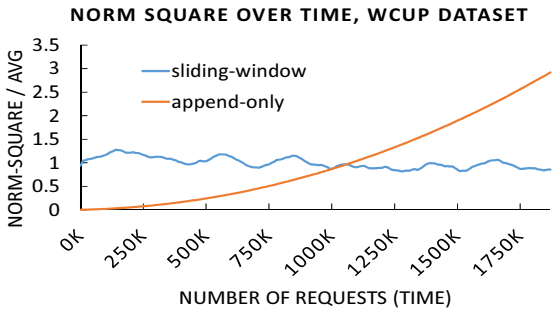

Figure 11: Norm-squared value over time for sliding and append-only windows, normalized by the average value for both cases. The advantage of CD over previous work ([11, 8]) is especially noticeable for the more general, and difficult, sliding window scenario.

Performance Metrics. The metrics used to evaluate CS and CD were the overall communication cost, measured as the total number of bytes comprising the data sent over the network, as well as the number of messages sent. The rationale for using two metrics is that typically there is a constant cost associated with every communication operation, which is independent of its net size (e.g. the size of a header, or the cost of opening a channel); hence, for example, sending 10 messages 5000 bytes each is cheaper than sending 100 messages 500 bytes each. Since the exact cost is usually some network-specific combination of the overall communication size and the number of messages, we provide both. We also present results for the combined cost for some typical parameters.

Violation Recovery. Whenever a local violation occurs (i.e. a local vector wanders outside its safe-zone), the corresponding node notifies the coordinator. The coordinator then attempts to resolve the local violation by searching for a subset of nodes (which contain the violating node), whose local vectors "balance" each other, that is, $p + d_{\text{avg}} \in C$, where d_{avg} is the average of the subset's drift vectors. Following [23], we applied the recovery scheme in which the coordinator gradually gathers local vectors until it manages to balance the violating ones.

Experimental Setting and Results. We implemented and compared the CD and CS methods on two data sets, which were also used in [11, 8]: "Wcup", which contains access logs to the soccer World Cup 1998 website[7], and "Cdad", which contains SNMP requests of network users, such as number of packets and bytes from/to each user's machine.[8] The data were collected from a corporate research center (IBM Watson) over several weeks. The parameters defining each run were the number of nodes (4 to 20), and ϵ and δ, which determine the accuracy and probabilistic guarantee level of the sketch, as well as its size (Section 4.1). The monitored function was f_N, and a threshold crossing is defined by a relative deviation of no more than θ from the current value. The frequency vectors were calculated for the feature "objectID" in Wcup and "shortRet" in Cdad.

From each dataset, we simulated a distributed streaming setup for between 4 and 20 nodes, by hashing the site field from the original data (the original Wcup data had 26 sites and Cdad had 174). We fixed sketch accuracy to $\delta = 2^{-11}$ and $\epsilon = 0.05$, and ran tests for different values of node numbers and θ.

Figs. 12,13 summarize the results. Performance is measured relative to the CS method. The results indicate a substantial improvement over CS, especially for the number of messages metric and for small values of θ, which correspond to a "tighter" monitoring problem, in which the monitored value is restricted to a narrower range.

Fig. 14 depicts a combined measure of both communication size and number of messages. We used the following typical assumptions: the protocol is TCP over ethernet, where packet payload is 1460 bytes and the headers (TCP+ETH) total 40 bytes. We also assume that the coordinator applies a broadcast channel. For these assumptions, the combined cost was dominated by the per-message overhead and not by the data size, and the advantage of CD over CS is clear.

Absolute Traffic and Scalability. In addition to the relative performance of CD vs. CS, we include next the absolute traffic. In Fig. 15, the ratio of the total traffic vs. the total stream size is provided, for 20 to 100 nodes. We differentiate between total and "upstream" traffic, the latter including only messages from the nodes to the coordinator, since often (e.g. in sensor networks) the "upstream" traffic is crucial to minimize, (e.g., it exhausts a sensor's power supply). It is evident that CD scales much better with the number of nodes, for both measures. These results for CS are consistent with those in [11], in which it was noted that, for over 10 nodes, the overall traffic for CS increases above the stream size[9].

[7]available at http://tinyurl.com/23ftxrl

[8]available at http://tinyurl.com/phdezz7

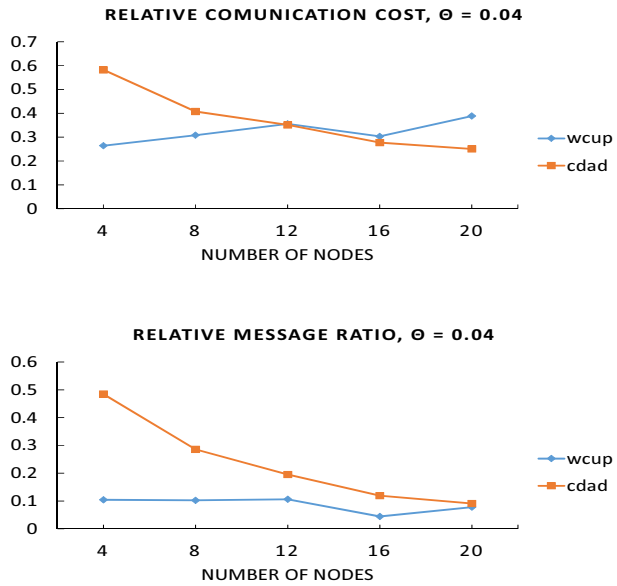

Figure 12: Scaled message and data cost, θ = 0.04, varying number of nodes.

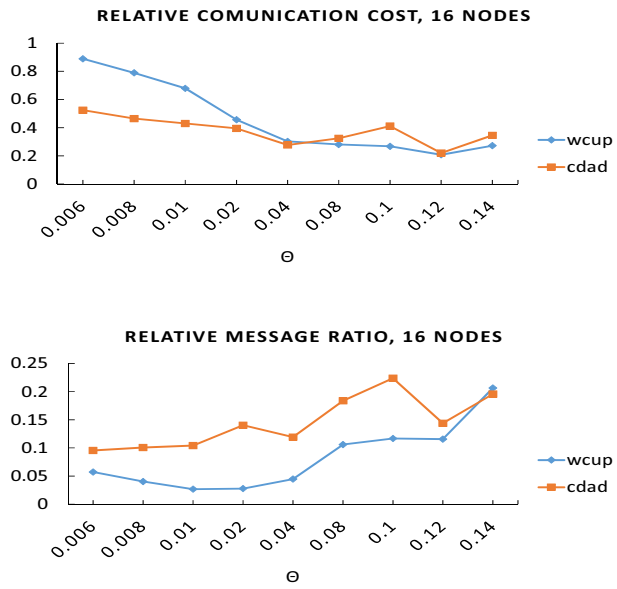

Figure 13: Scaled message and data cost, 16 nodes, varying θ.

6.1 CPU Cost

[9]In [11] only the overall traffic is provided; here we use the combined measure, which adds overhead for every message.

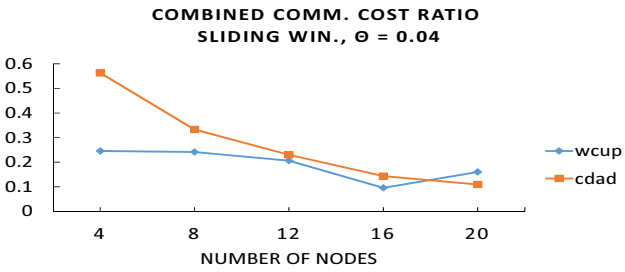

Figure 14: Combined cost, θ = 0.04, varying number of nodes.

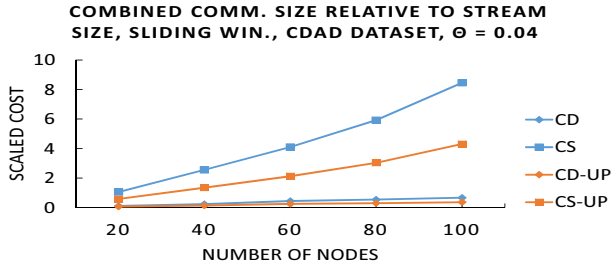

Figure 15: Combined cost, θ = 0.04, varying number of nodes, for CD and CS. Both total and "up" traffic are presented.

The CPU time required to process a given stream of length L, for both CD and CS, depends on three factors:

- The time is takes a single node to test the local condition for safe-zone containment (denoted T_s).
- The number of violations (N_v).
- The average size of the subset of nodes required to resolve a violation (V_r).

The overall processing time, O_t, is then $LT_s + N_v V_r T_s$. In our experiments, CD improved over CS in all these parameters; results for 12 nodes are provided in Table 1.

Improvement ratio (CS to CD)	T_s	V_r	O_t
	3.1	3.7	5.1

Table 1: average CPU cost, CS vs. CD.

The absolute running times were as follows: for a stream of length 2M, and sketch size 1600×11, overall time was about 2.1K seconds for CD (15K violating rounds) and 9.8K (74K violating rounds) for CS.

In Fig. 16 the average improvement factor in running times for violation recovery (which equals $V_r T_s$) is plotted for an increasing number of nodes.

Dependence on the sketch parameters. Recall (Section 4.1) that the sketch parameters ϵ (respectively δ) determine the number of sketch columns (respectively rows). CPU cost was proportional to the sketch size (rows times columns)[10]. While the improvement factor (in communication reduction) of CD over CS did not depend on the number of sketch columns, it generally modestly increased with the number of sketch rows (Fig. 17).

[10]For CD vs. CS CPU cost, please see first part of this sub-section.

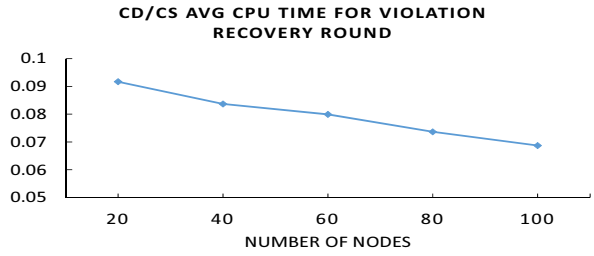

Figure 16: Comparison of CPU time, CS vs. CD.

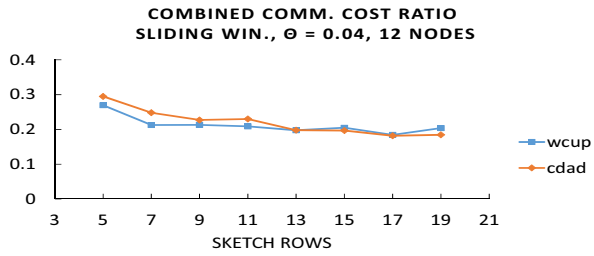

Figure 17: Dependence on the number of sketch rows.

6.2 Comparison to the Error Vectors method

Another approach for lowering the communication cost of monitoring function $f_N()$ (Section 4) was proposed in [11], which we refer to as the *error vectors* method. The idea is to reduce communication (when a local violation occurs at node i) by sending not the node's drift matrix d_i, but only the (much smaller) vector consisting of the norms of d_i's rows. If M is the reference matrix, then it follows from the triangle inequality that $f_N(M + \sum_i d_i) \leq \text{med}_k(||M_k|| + \sum_i ||(d_i)_k||)^2$ (where M_k resp. $(d_i)_k$ denotes the k-th row of M resp. d_i). Thus monitoring an upper bound for $f_N()$ can be replaced by monitoring $\text{med}_k(||M_k|| + \sum_i ||(d_i)_k||)^2$.

The *error vector* technique can easily be combined with the much improved safe zones derived by CD. However, since the error vectors method approximates $||\sum_i (d_i)_k||$ by its upper bound $\sum_i ||(d_i)_k||$, the tightness of this approximation is crucial to its success.

We remind that in [11] only the *cash register* model was assessed experimentally, where distributed streams are append-only. Fig. 18 contrasts three methods on append-only streams: CD, EV-CS (the technique of [11]) and EV-CD (combining error vectors with our CD safe zones). As can be seen, for the *cash register* workload, both CD and EV-CD consistently outperform EV-CS (and CS), albeit by a modest factor.

Next, we experimentally evaluate our methods on streams with a sliding window, which follow the so-called *turnstile* model, where streams may have both insertions and deletions. The communication costs, both in terms of messages and in terms of data size, are depicted in Fig. 19. Here it can be seen that the error vector technique increases the number of messages of the monitoring task by up to two orders of magnitude, for sliding window streams. Thus, despite these messages being much smaller for the error vector case, the amount of data transmitted increases significantly; The pure CD method clearly our-performs all other methods, both in terms of bytes sent and in terms of number of messages.

To explain this massive increase in the number of messages, we empirically assess the quality of the approximation of $||\sum_i (d_i)_k||$

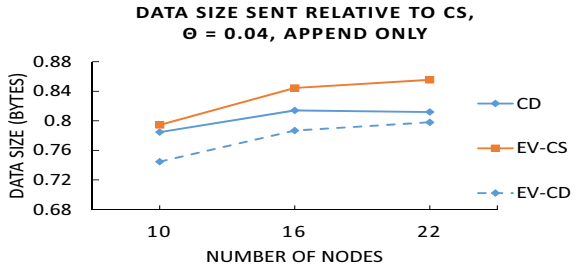

Figure 18: Total data sent in the cash register case: comparison of CS to CD, with the error vectors approach ("EV") and without. The results are w.r.t the basic CS algorithms (without EV).

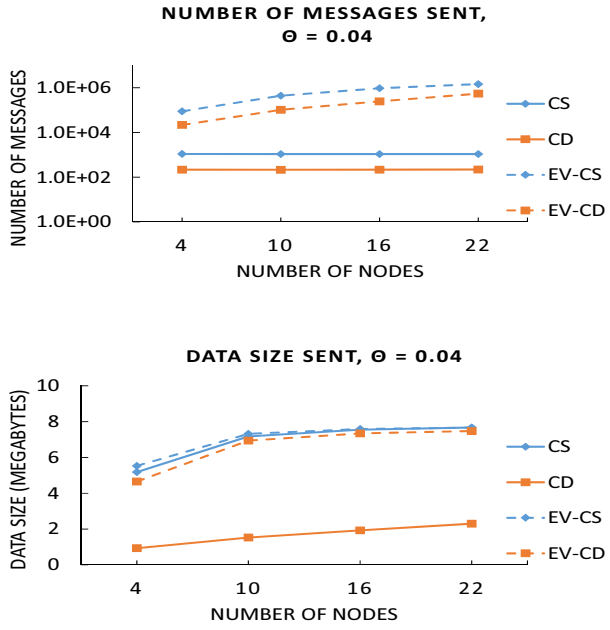

Figure 19: Number of messages (top) and total data sent (bottom) in the turnstile case: comparison of CS to CD, with the error vectors approach ("EV") and without. Note that top figure is in logarithmic scale. The advantage of CD in this case is very clear.

by $\sum_i ||(d_i)_k||$, both for the cash register and the turnstile model. The results are depicted in Fig. 20. It can be seen that, for the cash register model, the approximation applied by the error vectors method is reasonably tight (larger by about 15% than the actual value on average); by contrast, for the turnstile model, it is larger than the actual value by a factor close to 350%. Thus in this case, the bound applied in the error vectors method provides a poor approximation to the original drift matrices, and this causes numerous false alarms to be sent, resulting in a large communication overhead.

Note that our experimental results are backed by a theoretical argument (which is too lengthy to include here, but will appear in the full paper), predicting that, asymptotically, the ratio of the error vectors bound to the actual value is on average equal to $\sqrt{4/3}$ for

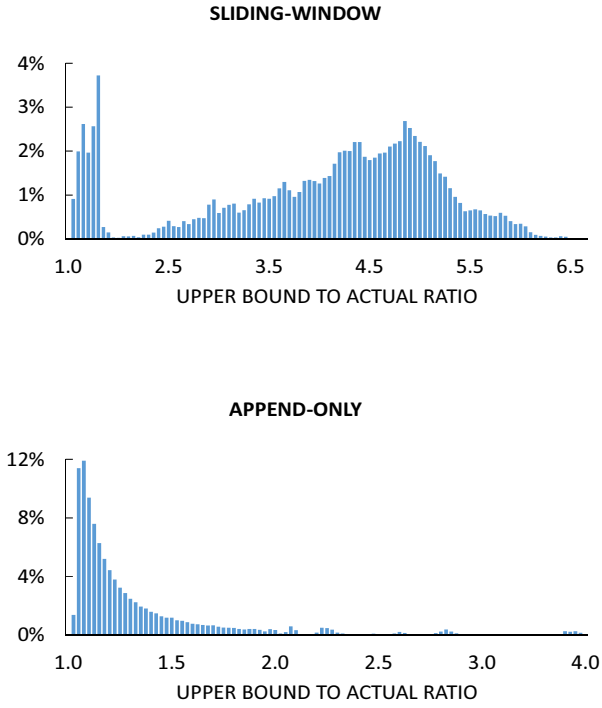

Figure 20: The empirical distribution of the ratio of the upper bound used in the "error vectors" approach to the actual value, turnstile model (top) and cash register model (bottom). Data is from Cdad (Section 6), 20 nodes.

the cash-register case, and to $\sqrt{m}$ for the turnstile model, where m is the number of nodes. This reaffirms that the error vectors method will not perform as well as CD for the turnstile model.

7. EXTENSIONS: JOIN AGGREGATES

A more general problem than monitoring norms and range queries is that of monitoring the inner product of two dynamic vectors (often referred to as the *join*). The join can also be approximated with AMS sketches. Typical applications include estimating the join size [24] and the estimation of the correlation between two vectorial quantities, e.g., frequency distributions; for example, one may be interested in the correlation between the purchase data in two different retail stores.

In order to define a convex decomposition (CD) for the join operation, note that $\langle x, y\rangle$, the inner product of two vectors x, y, is equal to $\frac{\|x+y\|^2-\|x-y\|^2}{4}$ (this is an extension of the identity $ab = \frac{(a+b)^2-(a-b)^2}{4}$ for scalars a, b.) By applying a rotation to align the axes with $x-y, x+y$, then (since rotation does not change the norm) it follows that monitoring $\langle x, y\rangle$ is equivalent to monitoring $\|x\|^2-\|y\|^2$. Let us look at the condition $\|x\|^2-\|y\|^2 \leq T$, and assume $T > 0$ (other cases are similarly treated). The set of all x, y satisfying this inequality is called a *hyperboloid* (see Fig. 21). A convex decomposition of the hyperboloid can be defined as follows: for a fixed y, note that the set $\{x|\ \|x\|^2 \leq \|y\|^2 + T\}$ is convex, since it is just a sphere with radius $\sqrt{\|y\|^2+T}$. Thus ranging over y defines a CD, each set of which is a "slice" through the hyperboloid (we note here, without proof, that it is impossible to find a *finite* CD of the hyperboloid). In Fig. 21, a hyperboloid and a "slice" through it are depicted in three dimensions.

It remains to show how to quickly determine safe zone containment. Due to lack of space, this technical discussion cannot be included here; we refer to an extended version of this paper, with an appendix which details the solution [1].

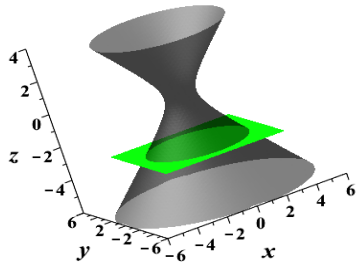

Figure 21: The hyperboloid defined by $x^2 + y^2 - z^2 = 1$, and a "slice" parallel to the $X-Y$ plane. Note that, while the hyperboloid is not convex, every such slice is convex; this holds in any dimension. The CD is defined by the union of all such slices.

In Fig. 22, the communication overhead for monitoring the join with the CS and CD methods is compared. As in [11], two streams were created by splitting the records (the Wcup dataset was split on the clientID attribute, and Cdad was split on the site attribute). While the improvement factor of CD over CS is larger for the Wcup data, it increases for both data sets as the number of nodes increases.

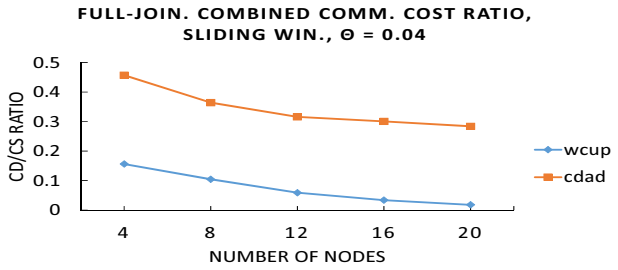

Figure 22: Combined message cost for monitoring the join, CS vs. CD.

8. CONCLUSIONS

We presented a monitoring algorithm for a distributed stream system, which is based on a convex decomposition (CD) of a subset of the queried function's domain, and proved that it always improves on the covering spheres (CS) method. To apply CD, a non-redundant convex decomposition should be first constructed. We showed how to achieve this for an important family of queries, namely range and norm queries over distributed streams, for the general case in which both negative and positive updates of the frequency counts are allowed. Experiments yielded substantial improvement over the state-of-the-art. Future research will concentrate on applying CD to other functions and scenarios.

Acknowledgments. This work was supported by the European Commission under ICT-FP7-FERARI-619491 (Flexible Event Processing for big Data Architectures).

9. REFERENCES

[1] http://tinyurl.com/oh3xvp6. Technical report.
[2] N. Alon, P. B. Gibbons, Y. Matias, and M. Szegedy. Tracking join and self-join sizes in limited storage. In *PODS*, 1999.
[3] C. Arackaparambil, J. Brody, and A. Chakrabarti. Functional monitoring without monotonicity. In *ICALP (1)*, 2009.
[4] B. Babcock and C. Olston. Distributed top-k monitoring. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, New York, NY, USA, 2003. ACM.
[5] A. Bar-Or, D. Keren, A. Schuster, and R. Wolff. Hierarchical decision tree induction in distributed genomic databases. *IEEE Trans. Knowl. Data Eng.*, 17(8):1138–1151, 2005.
[6] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
[7] S. Burdakis and A. Deligiannakis. Detecting outliers in sensor networks using the geometric approach. In *ICDE*, 2012.
[8] G. Cormode and M. N. Garofalakis. Approximate continuous querying over distributed streams. *ACM Trans. Database Syst.*, 33(2), 2008.
[9] G. Cormode, S. Muthukrishnan, and K. Yi. Algorithms for distributed functional monitoring. In *SODA*, 2008.
[10] M. Gabel, A. Schuster, and D. Keren. Communication-efficient distributed variance monitoring and outlier detection for multivariate time series. In *IEEE IPDPS*, pages 37–47, 2014.
[11] M. N. Garofalakis, D. Keren, and V. Samoladas. Sketch-based geometric monitoring of distributed stream queries. *PVLDB*, 2013.
[12] N. Giatrakos, A. Deligiannakis, M. N. Garofalakis, I. Sharfman, and A. Schuster. Prediction-based geometric monitoring over distributed data streams. In *SIGMOD*, 2012.
[13] L. Golab and M. T. Özsu. Issues in data stream management. *SIGMOD Record*, 32(2):5–14, 2003.
[14] R. Gupta, K. Ramamritham, and M. K. Mohania. Ratio threshold queries over distributed data sources. In *ICDE*, 2010.
[15] L. Huang, X. Nguyen, M. N. Garofalakis, J. M. Hellerstein, M. I. Jordan, A. D. Joseph, and N. Taft. Communication-efficient online detection of network-wide anomalies. In *INFOCOM*, 2007.
[16] S. R. Kashyap, J. Ramamirtham, R. Rastogi, and P. Shukla. Efficient constraint monitoring using adaptive thresholds. In *ICDE*, pages 526–535, 2008.
[17] R. Keralapura, G. Cormode, and J. Ramamirtham. Communication-efficient distributed monitoring of thresholded counts. In *SIGMOD*, 2006.
[18] D. Keren, I. Sharfman, A. Schuster, and A. Livne. Shape sensitive geometric monitoring. *IEEE Trans. Knowl. Data Eng.*, 24(8), 2012.
[19] S. Meng, T. Wang, and L. Liu. Monitoring continuous state violation in datacenters: Exploring the time dimension. In *ICDE*, pages 968–979, 2010.
[20] S. Michel, P. Triantafillou, and G. Weikum. Klee: a framework for distributed top-k query algorithms. In *VLDB '05*. VLDB Endowment, 2005.
[21] S. Shah and K. Ramamritham. Handling non-linear polynomial queries over dynamic data. In *ICDE*, 2008.
[22] I. Sharfman, A. Schuster, and D. Keren. A geometric approach to monitoring threshold functions over distributed data streams. In *SIGMOD*, 2006.
[23] I. Sharfman, A. Schuster, and D. Keren. A geometric approach to monitoring threshold functions over distributed data streams. *ACM Trans. Database Syst.*, 32(4), 2007.
[24] Z. Wei and K. Yi. Beyond simple aggregates: indexing for summary queries. In *PODS*, pages 117–128, 2011.
[25] R. Wolff, K. Bhaduri, and H. Kargupta. A generic local algorithm for mining data streams in large distributed systems. *IEEE Trans. on Knowl. and Data Eng.*, 21(4), 2009.
[26] K. Yi and Q. Zhang. Optimal tracking of distributed heavy hitters and quantiles. In *PODS*, 2009.