Efficient Threshold Monitoring for Distributed Probabilistic Data

Mingwang Tang, Feifei Li, Jeff M. Phillips, Jeffrey Jestes
School of Computing, University of Utah, Salt Lake City, USA
{tang, lifeifei, jeffp, jestes}@cs.utah.edu

***Abstract*— In distributed data management, a primary concern is monitoring the distributed data and generating an alarm when a user specified constraint is violated. A particular useful instance is the threshold based constraint, which is commonly known as the *distributed threshold monitoring* problem [4], [16], [19], [29]. This work extends this useful and fundamental study to distributed probabilistic data that emerge in a lot of applications, where uncertainty naturally exists when massive amounts of data are produced at multiple sources in distributed, networked locations. Examples include distributed observing stations, large sensor fields, geographically separate scientific institutes/units and many more. When dealing with probabilistic data, there are two thresholds involved, the score and the probability thresholds. One must monitor both simultaneously, as such, techniques developed for deterministic data are no longer directly applicable. This work presents a comprehensive study to this problem. Our algorithms have significantly outperformed the baseline method in terms of both the communication cost (number of messages and bytes) and the running time, as shown by an extensive experimental evaluation using several, real large datasets.**

I. Introduction

When massive amounts of data are generated, uncertainty is inherently introduced at the same time. For instance, data integration produces fuzzy matches [8], [31]; in measurements, e.g., sensor readings, data is inherently noisy, and is better represented by a probability distribution rather than a single deterministic value [2], [7], [10], [31]. In a lot of these applications, data are generated at multiple sources, and collected from distributed, networked locations. Examples include distributed observing stations, large sensor fields, geographically separate scientific institutes/units and many more [20], [34].

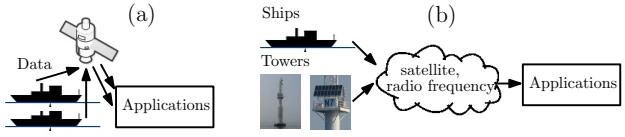


Fig. 1. Shipboard Automated Meteorological and Oceanographic System.

A concrete example is the Shipboard Automated Meteorological and Oceanographic System (SAMOS) project [27]. The goal of SAMOS is to provide effective access to marine meteorological and near-surface oceanographic observations from research vessels and voluntary ships (see Figure 1(a)). It has also been extended to include and integrate data from observation towers at sea in the northern Gulf of Mexico, as seen in Figure 1(b), in which we were involved. We have observed that in SAMOS: 1) data are naturally distributed: ships and towers are at geographically separated locations; 2) ambiguity, errors, imprecise readings, and uncertainty are present in the real-time data collected, due to hazardous conditions, coarse real-time measurement and multiple readings for the same observation; 3) large amounts of data needs to be processed; in fact, often times the ships and towers in SAMOS generate and report several hundreds of readings for a large number of attributes (e.g., wind speed, temperature, humidity, etc.) in less than a minute, continuously.

Clearly, it is useful to represent data in SAMOS (as well as other applications previously discussed) using *distributed probabilistic data*. For instance, due to the continuous influx of large amounts of data in very short periods, a common practice in SAMOS is for each ship/tower to buffer data for an interval (e.g., 5 minutes) and send one representative for data in an interval. Modeling data in a given interval using probabilistic data, such as a probability distribution function (pdf), is no doubt a viable and attractive solution (especially when we want to also account for the presence of uncertainty and errors in the raw readings). Meanwhile, as numerous studies in managing distributed data have shown, a primary concern is monitoring the distributed data and generating an alarm when a user-specified constraint is violated. A particular useful instance is the threshold based constraint, which we refer to as the *distributed threshold monitoring* (DTM) problem and has been extensively studied in distributed deterministic data [4], [16], [19], [29]. An application scenario is shown in Example 1.

Example 1 Suppose each distributed site continuously captures temperature readings (one per system-defined time instance), the goal is to monitor them continuously and raise an alarm at the coordinator site whenever the *average temperature* from all sites *exceeds* 80 degrees at any time instance.

Similar applications are required in managing distributed probabilistic data. And the notion of distributed threshold monitoring on probabilistic data is a critical problem, such as in the SAMOS system. The most natural and popular way of extending threshold queries to probabilistic data is probabilistic-threshold semantics [2], [5], [31], which introduces another threshold on the probability of the query answer in addition to the threshold on the score value of the results. Consider the following example that extends Example 1:

Example 2 Suppose readings in each site are now represented as probabilistic data (e.g., as we have just discussed for data in SAMOS), the goal is to monitor these readings continuously and raise an alarm at the coordinator site whenever *the*

probability of the *average temperature* from all sites *exceeding* 80 degrees is *above* 70% at any time instance.

We refer to them as the *distributed probabilistic threshold monitoring* (DPTM) problem. This variant is a robust alternative to DTM, more robust than the median, in that even if *all* sites report low-probability noise which skews their distributions, DPTM will only raise an alarm if a true threshold has been crossed, or what may have been noise occurs with high enough probability that it cannot be ignored. For the same reasons and motivations of its counterpart, the DTM problem, a paramount concern is to reduce the communication cost, measured by both the total number of messages and bytes communicated in the system. For example, on SAMOS, cutting down the communication cost would allow for the transmission of more accurate or diverse measurements. Due to the inherent difference in query processing between probabilistic and deterministic data, techniques developed from DTM are no longer directly applicable. This also brings up another challenge in DPTM, reducing the cpu cost, since query processing in probabilistic data is often computation-intensive which is even worse in distributed probabilistic data [20].

This work steps up to these challenges and presents a comprehensive study to the DPTM problem. Specifically:

- We formalize the DPTM problem in Section II.
- We propose baseline methods in Section III, which improve over the naive method of sending all tuples at each time instance.
- We design two efficient and effective monitoring methods in Section IV that leverage moment generating functions and adaptive filters to significantly reduce the costs.
- When an exact solution is not absolutely necessary, we introduce novel sampling-based methods in Section V to further reduce the communication and the cpu costs.
- We extensively evaluate all proposed methods in Section VII on large real data obtained from research vessels in the SAMOS project. The results have shown that our monitoring methods have significantly outperformed the baseline approach. They also indicate that our sampling method is very effective when it is acceptable to occasionally miss one or two alarms with very small probability.

We discuss some useful extensions in Section VI, survey the related work in Section VIII, and conclude in Section IX.

II. Problem Formulation

Sarma *et al.* [28] describe various models of uncertainty. We consider the attribute-level uncertain tuple that has been used frequently in the literature, and suits the applications for our problem well (e.g., data in SAMOS).

Each tuple has one or more uncertain attributes. Every uncertain attribute has a pdf for its value distribution. Correlation among attributes in one tuple can be represented by a joint pdf. This model has a lot of practical applications and is most suitable for measurements and readings [7], [18]. Without loss of generality, we assume that each tuple has only one uncertain attribute *score*. Let X_i be the random variable for the *score* of tuple d_i, where X_i can have either a discrete or a continuous pdf, with bounded size (see Figure 2). Since each pdf is bounded, we assume that for all X_i's, $|X_i| \leq n$ for some value n where $|X_i|$ is the size of the pdf for X_i, which is the number of discrete values X_i may take for a discrete pdf, or the number of parameters describing X_i and its domain for a continuous pdf.

tuples	attribute *score*
d_1	X_1
d_2	X_2
.	...
d_t	X_t

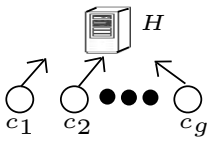


Fig. 2. Attribute-level uncertain tuple and the flat model.

Given g distributed clients $\{c_1, \ldots, c_g\}$, and a centralized server H. We consider the *flat model* for the organization of distributed sites as shown in Figure 2, e.g., SAMOS uses the flat model. At each time instance t, for $t = 1, \ldots, T$, client c_i reports a tuple $d_{i,t}$ with a score $X_{i,t}$. We assume that data from different sites, are independent. Similar assumptions were made in most distributed data monitoring or ranking studies [4], [12], [16], [19], [20], [23], [29]. Without loss of generality and for the ease of explanation, we assume that $X_{i,t} \in \mathbb{R}^+$. Our techniques can be easily extended to handle the case when $X_{i,t}$ may take negative values as well.

Definition 1 (DPTM) Given $\gamma \in \mathbb{R}^+$ and $\delta \in [0, 1)$, let $Y_t = \sum_{i=1}^{g} X_{i,t}$, for $t = 1, \ldots, T$. The goal is to raise an alarm at H, whenever for any $t \in [1, T]$ $\Pr[Y_t > \gamma] > \delta$.

In our definition, DPTM monitors the *sum* constraint. Monitoring the *average* constraint is equivalent to this case, as well as any other types of constraints that can be expressed as a linear combination of one or more *sum* constraints.

As argued in Section I, the goal is to minimize both the *overall* communication and computation costs, at the end of all time instances. We measure the communication cost using both the total number of bytes transmitted and the total number of messages sent. Lastly, when the context is clear, we omit the subscript t from Y_t and $X_{i,t}$.

III. Baseline Methods

At any time instance t, let $X_1, \ldots, X_g$ be the scores from c_1 to c_g and $Y = \sum_{i=1}^{g} X_i$. To monitor if $\Pr[Y > \gamma] > \delta$, the naive method is to ask each client c_i to send his score X_i to H, which is clearly very expensive.

A. Compute $\Pr[Y > \gamma]$ Exactly

The first challenge is to compute $\Pr[Y > \gamma]$ exactly at H. We differentiate two cases. When each X_i is represented by discrete pdf, clearly, we can compute $Y_{1,2} = X_1 + X_2$ in $O(n^2)$ time by a nested loop over the possible values they may take. Next, we can compute $Y_{1,2,3} = X_1 + X_2 + X_3 = Y_{1,2} + X_3$ using $Y_{1,2}$ and X_3 in $O(n^3)$ time, since in the worst case the size of $Y_{1,2}$ is $O(n^2)$. We can recursively apply this idea to compute $Y = Y_{1,\ldots,g}$ in $O(n^g)$ time, then check $\Pr[Y > \gamma]$ exactly. But note that in this approach, since we did not sort the values in the output (to reduce the cost), in each step the discrete values in the output pdf is no longer sorted.

A better idea is to compute $Y_{1,\ldots,g/2}$, and $Y_{g/2+1,\ldots,g}$ separately, which only takes $O(n^{g/2})$ time. Then, by using the cdf (cumulative distribution function) of $Y_{g/2+1,\ldots,g}$, we can compute $\Pr[Y > \gamma]$ as follows:

$$\Pr[Y > \gamma] = \sum_{\forall y \in Y_{1,\ldots,g/2}} \Pr[Y_{1,\ldots,g/2} = y] \cdot \Pr[Y_{g/2+1,\ldots,g} > \gamma - y].$$

Computing the cdf of $Y_{g/2+1,\ldots,g}$ takes $O(n^{g/2} \log n^{g/2})$ in the worst case: as discussed above, discrete values in $Y_{g/2+1,\ldots,g}$ are not sorted. After which, finding out $\Pr[Y_{g/2+1,\ldots,g} > \gamma - y]$ for any y takes only constant time. Hence, this step takes $O(n^{g/2})$ time only (the size of $Y_{1,\ldots,g/2}$ in the worst case). So the overall cost of computing $\Pr[Y > \gamma]$ exactly at H becomes $O(n^{g/2} \log n^{g/2})$.

When some X_i's are represented by continuous pdfs, the above process no longer works. In this case, we leverage on the characteristic functions of X_i's to compute Y exactly. The characteristic function [1] of a random variable X is:

$$\varphi_X(\beta) = \mathbf{E}(e^{i\beta X}) = \int_{-\infty}^{+\infty} e^{i\beta x} f_X(x) d(x), \forall \beta \in \mathbb{R},$$

where i is the imaginary unit and $f_X(x)$ is the pdf of X. Let $\varphi_i(\beta)$ and $\varphi(\beta)$ be the characteristic functions of X_i and Y respectively, a well-known result is that $\varphi(\beta) = \prod_{i=1}^{g} \varphi_i(\beta)$ [1]. Furthermore, by definition, $\varphi_i(\beta)$ and $\varphi(\beta)$ are the Fourier transform of the pdfs for X_i and Y respectively. Hence, an immediate algorithm for computing the pdf of Y is to compute the Fourier transforms for the pdfs of X_i's, multiply them together to get $\varphi(\beta)$, then do an inverse Fourier transform on $\varphi(\beta)$ to obtain the pdf of Y. After which, we can easily check if $\Pr[Y > \gamma] > \delta$. The cost of this algorithm depends on the cost of each Fourier transform, which is dependent on the types of distributions being processed. Note that using this approach when all pdfs are discrete does not result in less running time than the method above: since the size of Y in the worst case is $O(n^g)$ (the pdf describing Y), this algorithm takes at least $O(n^g)$ time in the worst case, even though we can leverage on fast Fourier transform in this situation.

We denote the above algorithms as EXACTD and EXACTC, for the discrete and continuous cases respectively.

B. Filtering by Markov Inequality

By the Markov inequality, we have $\Pr[Y > \gamma] \leq \frac{\mathbf{E}(Y)}{\gamma}$. Given that $\mathbf{E}(Y) = \sum_{i=1}^{g} \mathbf{E}(X_i)$, if each client X_i only sends $\mathbf{E}(X_i)$, H can check if $\frac{\mathbf{E}(Y)}{\gamma} < \delta$; if so, no alarm should be raised for sure; otherwise, we can then ask for X_i's, and apply the exact algorithm. We dub this approach the *Markov* method.

We can improve this further. Since $\mathbf{E}(Y) = \sum_{i=1}^{g} \mathbf{E}(X_i)$ and our goal is to monitor if $\mathbf{E}(Y) < \gamma\delta$ by the Markov inequality, we can leverage on the adaptive thresholds algorithm for the DTM problem in deterministic data [16] to monitor if $\sum_{i=1}^{g} \mathbf{E}(X_i) < \gamma\delta$ continuously, which installs local filters at clients and adaptively adjusts them. Specifically, $\gamma\delta$ is treated as the global constraint; at each time instance, client c_i can compute $\mathbf{E}(X_i)$ locally which becomes a "deterministic score". Thus, the algorithm from [16] is applicable. Whenever it cannot assert an alarm at a time instance t, clients transmit X_i's to H and the server applies the exact algorithm (only for that instance). This helps reduce the communication cost and we dub this improvement the *Madaptive* method.

IV. Improved Methods

We now improve on these baseline techniques. We replace the Markov Inequality through more complicated to apply, but more accurate, Chebyshev and Chernoff bounds (*Improved* Fig 3). Then, we redesign *Improved* to leverage adaptive monitoring techniques designed for DTM (*Iadaptive* Fig 4).

A. Improved Bounds on $\Pr[Y > \gamma]$

We first leverage on the general Chebyshev bound:

$$\Pr[|Y - \mathbf{E}(Y)| \geq a\sqrt{\mathrm{Var}(Y)}] \leq 1/a^2 \text{ for any } a \in \mathbb{R}^+,$$

which gives us the following one-sided forms:

$$\Pr[Y \geq \mathbf{E}(Y) + a] \leq \frac{\mathrm{Var}(Y)}{\mathrm{Var}(Y) + a^2}, \forall a \in \mathbb{R}^+ \quad (1)$$

$$\Pr[Y \leq \mathbf{E}(Y) - a] \leq \frac{\mathrm{Var}(Y)}{\mathrm{Var}(Y) + a^2}, \forall a \in \mathbb{R}^+. \quad (2)$$

When $\gamma > \mathbf{E}(Y)$, setting $a = \gamma - \mathbf{E}(Y)$ in (1) leads to:

$$\Pr[Y > \gamma] < \Pr[Y \geq \gamma] \leq \frac{\mathrm{Var}(Y)}{\mathrm{Var}(Y) + (\gamma - \mathbf{E}(Y))^2}. \quad (3)$$

As such, when $\gamma > E(Y)$, if $\frac{\mathrm{Var}(Y)}{\mathrm{Var}(Y)+(\gamma-\mathbf{E}(Y))^2} \leq \delta$, we definitely do not have to raise an alarm.

When $\gamma < \mathbf{E}(Y)$, we can set $a = \mathbf{E}(Y) - \gamma$ in (2) to get:

$$\Pr[Y \leq \gamma] \leq \frac{\mathrm{Var}(Y)}{\mathrm{Var}(Y) + (\mathbf{E}(Y) - \gamma)^2}. \quad (4)$$

This implies that,

$$\Pr[Y > \gamma] = 1 - \Pr[Y \leq \gamma] > 1 - \frac{\mathrm{Var}(Y)}{\mathrm{Var}(Y)+(\mathbf{E}(Y)-\gamma)^2}. \quad (5)$$

Hence, when $\gamma < \mathbf{E}(Y)$, as long as $1 - \frac{\mathrm{Var}(Y)}{\mathrm{Var}(Y)+(\mathbf{E}(Y)-\gamma)^2} \geq \delta$, we should surely raise an alarm.

Given these observations, in each time instance, clients send $\mathbf{E}(X_i)$'s and $\mathrm{Var}(X_i)$'s to H, which computes $\mathbf{E}(Y)$ and $\mathrm{Var}(Y)$ locally (given that X_i's are independent from each other, $\mathrm{Var}(Y) = \sum_{i=1}^{g} \mathrm{Var}(X_i)$). Depending whether $\mathbf{E}(Y) > \gamma$ or $\mathbf{E}(Y) < \gamma$, H uses (3) or (5) to decide to raise or not to raise an alarm for this time instance. Nevertheless, this approach may still incur expensive communication and computation when $\mathbf{E}(Y) = \gamma$, or (3) ((5), resp.) does not hold when $\mathbf{E}(Y) > \gamma$ ($\mathbf{E}(Y) < \gamma$, resp.). It is also limited in the fact that H can only check either to raise an alarm or not to raise an alarm, but not both simultaneously, as $\mathbf{E}(Y) > \gamma$ and $\mathbf{E}(Y) < \gamma$ cannot hold at the same time.

We remedy these problems using the general Chernoff bound and the moment-generating function [1]. For any random variable Y, suppose its moment generating function is given by $M(\beta) = \mathbf{E}(e^{\beta Y})$ for any $\beta \in \mathbb{R}$, then:

$$\Pr[Y \geq a] \leq e^{-\beta a} M(\beta) \text{ for all } \beta > 0, \forall a \in \mathbb{R} \quad (6)$$

$$\Pr[Y \leq a] \leq e^{-\beta a} M(\beta) \text{ for all } \beta < 0, \forall a \in \mathbb{R} \quad (7)$$

Here a can be any real value (positive or negative). Suppose the moment generating function of X_i and Y is $M_i(\beta)$ and $M(\beta)$ respectively, then $M(\beta) = \prod_{i=1}^{g} M_i(\beta)$ [1]. Hence, when the checking based on either (3) or (5) has failed, for any $\beta_1 > 0$ and $\beta_2 < 0$, the server requests c_i to calculate and send back $M_i(\beta_1)$ and $M_i(\beta_2)$. He computes $M(\beta_1)$ and $M(\beta_2)$, and by setting $a = \gamma$ in (6) and (7), he checks if:

$$\Pr[Y > \gamma] \leq \Pr[Y \geq \gamma] \leq e^{-\beta_1 \gamma} M(\beta_1) \leq \delta, \text{ and} \quad (8)$$

$$\Pr[Y > \gamma] = 1 - \Pr[Y \leq \gamma] > 1 - e^{-\beta_2 \gamma} M(\beta_2) \geq \delta. \quad (9)$$

When (8) holds, he does not raise an alarm; when (9) holds, he raises an alarm; only when both have failed, he requests X_i's for the exact computation.

Calculating $M_i(\beta)$ at a client c_i is easy. For a lot of parametric continuous pdfs, closed-form expressions exist for their moment generating functions, or, one can use numeric methods to compute $M_i(\beta)$ to arbitrary precision for other continuous pdfs. For discrete pdfs, $M_i(\beta) = \sum_{x \in X_i} e^{\beta x} \Pr[X_i = x]$.

Another key issue is to figure out the optimal values for β_1 and β_2 in (8) and (9) to make the corresponding bound as tight as possible, which is to minimize $e^{-\beta_1 \gamma} M(\beta_1)$ and $e^{-\beta_2 \gamma} M(\beta_2)$ in (8) and (9) respectively. The *central limit theorem* states that the mean of a sufficiently large number of independent random variables will be approximately normally distributed, if each independent variable has finite mean and variance [1]. For a normal distribution with mean μ and variance σ^2, its moment generating function is $e^{\beta\mu + \frac{1}{2}\sigma^2\beta^2}$ for any $\beta \in \mathbb{R}$. Hence, let $Y' = \frac{1}{g}Y$, then Y' can be approximated by a normal distribution well, and we can approximate its moment generating function as:

$$M_{Y'}(\beta) \approx e^{\beta \mathbf{E}(Y') + \frac{1}{2}\mathrm{Var}(Y')\beta^2}, \forall \beta \in \mathbb{R}. \quad (10)$$

Note that $Y = gY'$, (8) and (10) imply that for any $\beta_1 > 0$:

$$\begin{aligned} \Pr[Y \geq \gamma] &= \Pr[Y' \geq \frac{\gamma}{g}] \leq e^{-\beta_1 \frac{\gamma}{g}} M_{Y'}(\beta_1) \\ &\approx e^{-\beta_1 \frac{\gamma}{g}} e^{\beta_1 \mathbf{E}(\frac{Y}{g}) + \frac{1}{2}\mathrm{Var}(\frac{Y}{g})\beta_1^2} \quad \text{by (10)} \\ &= e^{\frac{\beta_1}{g}(\mathbf{E}(Y) - \gamma) + \frac{1}{2g^2}\mathrm{Var}(Y)\beta_1^2} \quad (11) \end{aligned}$$

Hence, we can approximate the optimal β_1 value for (8) by finding the β_1 value that minimizes the RHS of (11). Let $f(\beta_1)$ be the RHS of (11) and take its derivative w.r.t. β_1:

$$f'(\beta_1) = e^{\frac{\beta_1}{g}(\mathbf{E}(Y) - \gamma) + \frac{1}{2g^2}\mathrm{Var}(Y)\beta_1^2} \left(\frac{\mathbf{E}(Y) - \gamma}{g} + \frac{\mathrm{Var}(Y)}{g^2}\beta_1\right).$$

Let $f'(\beta_1) = 0$, we get $\beta_1 = \frac{g(\gamma - \mathbf{E}(Y))}{\mathrm{Var}(Y)}$. Furthermore, we can show that the second order derivative of $f(\beta_1)$,$f''(\beta_1)$, is always greater than 0 (we omit the details for brevity). That said, $f(\beta_1)$ (hence, the RHS of (11)) takes its minimal value when $\beta_1 = \frac{g(\gamma - \mathbf{E}(Y))}{\mathrm{Var}(Y)}$. Using a similar analysis, we can derive the optimal β_2 value. However, a constraint is that $\beta_1 > 0$ and $\beta_2 < 0$. That said, also with the observation that $f(\beta_1)$ (the corresponding function for β_2) is monotonically increasing when $\beta_1 > \frac{g(\gamma - \mathbf{E}(Y))}{\mathrm{Var}(Y)}$ ($\beta_2 < \frac{g(\gamma - \mathbf{E}(Y))}{\mathrm{Var}(Y)}$ respectively), let

Algorithm Improved($c_1, \ldots, c_g, H$)
1. for $t = 1, \ldots, T$
2. let $X_i = X_{i,t}$ and $Y = Y_t = \sum_{i=1}^{g} X_i$;
3. each c_i computes $\mathbf{E}(X_i)$ and $\mathrm{Var}(X_i)$ locally, and sends them to H;
4. H sets $\mathbf{E}(Y) = \sum \mathbf{E}(X_i)$, $\mathrm{Var}(Y) = \sum \mathrm{Var}(X_i)$;
5. if ($\gamma > \mathbf{E}(Y)$ and $\frac{\mathrm{Var}(Y)}{\mathrm{Var}(Y) + (\gamma - \mathbf{E}(Y))^2} \leq \delta$)
6. raise no alarm; **continue** to next time instance;
7. if ($\gamma < \mathbf{E}(Y)$ and $1 - \frac{\mathrm{Var}(Y)}{\mathrm{Var}(Y) + (\mathbf{E}(Y) - \gamma)^2} \geq \delta$)
8. raise an alarm; **continue** to next time instance;
9. H sets β_1 and β_2 according to (12);
10. H broadcasts β_1, β_2 to all clients, and asks them to compute and send back $M_i(\beta_1)'s$ and $M_i(\beta_2)'s$;
11. H sets $M(\beta_1) = \prod_i M_i(\beta_1)$, $M(\beta_2) = \prod_i M_i(\beta_2)$;
12. if ($e^{-\beta_1 \gamma} M(\beta_1) \leq \delta$)
13. raise no alarm; **continue** to next time instance;
14. if ($1 - e^{-\beta_2 \gamma} M(\beta_2) \geq \delta$)
15. raise an alarm; **continue** to next time instance;
16. H asks for X_i's, applies EXACTD or EXACTC;

Fig. 3. The *Improved* method.

$\theta > 0$ be some small value,

$$\begin{cases} \beta_1 = \frac{g(\gamma - \mathbf{E}(Y))}{\mathrm{Var}(Y)}, \beta_2 = -\theta & \text{if } \gamma > \sum_{i=1}^{g} \mathbf{E}(X_i), \\ \beta_1 = \theta, \beta_2 = \frac{g(\gamma - \mathbf{E}(Y))}{\mathrm{Var}(Y)} & \text{if } \gamma < \sum_{i=1}^{g} \mathbf{E}(X_i), \\ \beta_1 = \theta, \beta_2 = -\theta & \text{otherwise,} \end{cases} \quad (12)$$

will help achieve tight bounds in (8) and (9).

This yields the *Improved* method, shown in Figure 3.

B. Improved Adaptive Threshold Monitoring

The *Improved* method needs at least g messages per time instance, to reduce this, we again leverage on the adaptive thresholds algorithm developed for work on DTM [16].

Consider (8) and (9), when we can continuously monitor if:

$$e^{-\beta_1 \gamma} \prod_{i=1}^{g} M(\beta_1) \leq \delta, \text{ or } 1 - e^{-\beta_2 \gamma} \prod_{i=1}^{g} M(\beta_2) \geq \delta \quad (13)$$

efficiently, whenever the first inequality in (13) holds at a time instance t, H knows for sure that $\Pr[Y > \gamma] \leq \delta$ at t and no alarm should be raised at this time instance; whenever the second inequality in (13) holds at t, H knows for sure that $\Pr[Y > \gamma] > \delta$ at t and an alarm should be raised. Monitoring the first inequality in (13) is the same as monitoring if

$$\sum_{i=1}^{g} \ln M_i(\beta_1) \leq \ln \delta + \beta_1 \gamma. \quad (14)$$

We can treat $(\ln \delta + \beta_1 \gamma)$ as the global constraint, and at time t, let $V_i = \ln M_i(\beta_1)$ be the local deterministic score at client c_i; this becomes the exactly same formulation for the DTM problem. We now apply the adaptive thresholds algorithm for constraint monitoring from [16] to monitor (14). We denote this monitoring instance as J_1. At any time t, if J_1 raises no alarm, H knows that no alarm should be raised at t, since by implication (14) holds, and hence $\Pr[Y > \gamma] \leq \delta$.

Monitoring the 2nd inequality in (13) is to monitor if

$$\sum_{i=1}^{g} \ln M_i(\beta_2) \leq \ln(1-\delta) + \beta_2\gamma. \quad (15)$$

By treating $(\ln(1-\delta)+\beta_2\gamma)$ as the global constraint, at time t let $W_i = \ln M_i(\beta_2)$ be the local deterministic score at client c_i; then we again apply [16] to monitor (15). Denote this monitoring instance as J_2. Constrasting J_1 to J_2, when J_2 does *not* report an alarm at t, it means that (15) holds, which implies that $\Pr[Y > \gamma] > \delta$, so H needs to raise an alarm.

One choice is to let H run both J_1 and J_2. However, when $\Pr[Y > \gamma]$ deviates from δ considerably, one of them will almost always raise alarms, which results in a global poll and adjusting the local filters [16]. So the total communication cost will actually be higher than running just one. A critical challenge is deciding which instance to run. A simple and effective method is to make this decision periodically using recent observations of $\Pr[Y > \gamma]$ and δ.

Suppose we set the period to k, and the current time instance is t. For any $i \in [t-k, t)$, let $e_i = 1$ if $\Pr[Y_i > \gamma] > \delta$ and 0 otherwise; and $e = \sum_{i=t-k}^{t-1} e_i$. If $e \geq k/2$, then in majority of recent instances $\Pr[Y_i > \gamma] > \delta$, hence (15) is more likely to hold and J_2 is most likely not going to raise alarms and more efficient to run. If $e < k/2$, in majority of recent instances $\Pr[Y_i > \gamma] < \delta$, (14) is more likely to hold and J_1 is most likely not going to raise alarms and more efficient to run.

Another question is how to set the β_1 and β_2 values in (14) and (15). Since they are derived directly from (13), which are originated from (8) and (9), the same way of setting them as shown in (12) will likely result in tight bounds, thus, less violations to (14) and (15), making J_1 and J_2 efficient to run respectively. However, this does require H to ask for $\mathbf{E}(X_i)$'s and $\mathrm{Var}(X_i)$'s in every time instance, defeating the purpose of using the adaptive thresholds algorithm to reduce the number of messages. To remedy, we let H reset the optimal β_1 and β_2 values for the two adaptive thresholds instances periodically in every k time instances, for a system parameter k.

The complete algorithm, *Iadaptive*, is shown in Figure 4.

V. Sampling Methods to Estimate The Threshold

In either of the previous methods, when the algorithm fails to definitively indicate that an alarm should be raised or not, then likely $\Pr[Y > \gamma]$ is close to δ. If H needs to be sure that the (γ, δ) threshold is crossed, all of X_i have to be retrieved, and the exact algorithms in Section III-A are applied. But in a lot of situations, this is expensive and impractical, due to both the communication and computation costs involved. Since uncertainties naturally exist in probabilistic data, it is very likely that users are willing to approximate the conditions under which the server raises the alarm, if approximation guarantees can be provided.

A natural choice and standard approximation is to leverage random sampling. Technical details of adapting the most general random sampling technique to our problem are presented in Appendix A, and designated MRS. It approximates $\Pr[Y > \gamma]$ within ε with at least $(1-\phi)$ probability, using $O(g/\varepsilon^2 \ln(1/\phi))$ bytes, for any $\varepsilon, \phi \in (0,1)$.

Algorithm Iadaptive($c_1, \ldots, c_g, H, k$)
1. initialize (without starting) two adaptive thresholds instances J_1, J_2 [16]: J_1 monitors $\sum_i V_i \leq \ln\delta + \beta_1\gamma$, and J_2 monitors if $\sum_i W_i \leq \ln(1-\delta) + \beta_2\gamma$;
2. H sets β_1 to a small positive value, $e = 0$, starts J_1;
3. for $t = 1, \ldots, T$
4. let $X_i = X_{i,t}$, $Y = Y_t = \sum X_i$;
5. c_i computes $V_i = \ln M_i(\beta_1)$, or $W_i = \ln M_i(\beta_2)$;
6. if (J_1 is running and raises no alarm)
7. H raises no alarm; **continue** to line 11;
8. if (J_2 is running and raises no alarm)
9. H raises an alarm; $e = e + 1$; **continue** line 11;
10. H asks for X_i's, applies EXACTD or EXACTC, sets $e = e + 1$ if an alarm is raised;
11. if ($t \mod k == 0$)
12. stop the currently running instance J_x;
13. each c_i sends $\mathbf{E}(X_i)$ and $\mathrm{Var}(X_i)$ to H;
14. reset β_1 in J_1 and β_2 in J_2 according to (12);
15. if ($e \geq k/2$) set $x = 2$ else set $x = 1$;
16. H sets $e = 0$, starts J_x, broadcasts setup information of J_x, and new β_1 and β_2 values;

Fig. 4. The *Iadaptive* method.

A. Random Distributed ε-Sample

Instead of using the standard random sampling approach as shown in Appendix A, we can leverage on a more powerful machinery in our analysis to derive a new algorithm with the same guarantee w.r.t. a fixed pair of thresholds (γ, δ), but it is simpler to implement and works better in practice. Later, in Section VI, we also show that it can handle multiple pairs of thresholds simultaneously without incurring additional costs.

We can approximate the probabilities of raising an alarm by a Monte Carlo approach where H asks each c_i for a sample x_i from X_i. He then computes a value $y = \sum_{i=1}^{g} x_i$; this is a sample estimate from the distribution over Y, so $\Pr[Y > \gamma] = \Pr[y > \gamma]$. Repeating this to amplify success is the *r*andom *d*istributed ε-*s*ample (RDεS) algorithm in Figure 5.

Algorithm RDεS ($c_1, \ldots, c_g, H, t, \varepsilon, \phi$)
1. $X_i = X_{i,t}$, $Y = \sum_{i=1}^{g} X_i$, $S_i = \emptyset$, $v = 0$, $\kappa = \frac{1}{\varepsilon^2}\ln\frac{1}{\phi}$;
2. for $i = 1, \ldots, g$
3. for $j = 1, \ldots, \kappa$
4. c_i selects some value $x_{i,j}$ from X_i, into S_i, at random according to its underlying distribution;
5. c_i sends S_i to H;
6. for $j = 1, \ldots, \kappa$
7. if ($y_j = \sum_{i=1}^{g} x_{i,j} > \gamma$) $v = v + 1$;
8. if ($v/\kappa > \delta$) H raises an alarm;
9. else H raises no alarm;

Fig. 5. The RDεS method.

Theorem 1 RDεS *gives* $\mathbf{E}(v/\kappa) = \Pr[Y > \gamma]$ *and* $\Pr[|v/\kappa - \Pr[Y > \gamma]| \leq \varepsilon] \geq 1 - \phi$, *using* $O(\frac{g}{\varepsilon^2}\ln\frac{1}{\phi})$ *bytes.*

Proof: First, it is clear that in line 7 for any $j \in [1,\kappa]$, $y_j = \sum_{i=1}^{g} x_{i,j}$ is a random sample drawn from the distribution of Y. Hence, $\mathbf{E}(v) = \kappa \cdot \Pr[Y > \gamma]$.

We next leverage on the concept of *VC-dimension* [33]. Let P be a set of points, or more generally a distribution. Let $\mathcal{I}$ be a family of subsets of P. Let P have domain $\mathbb{R}$ and let $\mathcal{I}$ consist of ranges of the form of one-sided intervals (x,∞) for any $x \in \mathbb{R}$. The pair $(P,\mathcal{I})$ is a *range space* and we say a subset $X \subset P$ *shatters* a range space $(P,\mathcal{I})$ if every subset $X_s \subseteq X$ can be defined as $I \cap X$ for some $I \in \mathcal{I}$. The size of the largest subset X that shatters $(P,\mathcal{I})$ is the VC-dimension of $(P,\mathcal{I})$. For one-sided intervals $\mathcal{I}$, the VC-dimension for a range space $(P,\mathcal{I})$ using any set P is $\nu = 1$.

An *ε-sample* for a range space $(P,\mathcal{I})$ is a subset $Q \subset P$ that approximates the density of P such that:

$$\max_{I \in \mathcal{I}} \left| \frac{|I \cap P|}{|P|} - \frac{|I \cap Q|}{|Q|} \right| \leq \varepsilon. \tag{16}$$

A classic result of Vapnik and Chervonenkis [33] shows that if $(P,\mathcal{I})$ has VC-dimension ν and if Q is a random sample from P of size $O((\nu/\varepsilon^2)\log(1/\phi))$, then Q is an ε-sample of $(P,\mathcal{I})$ with probability at least $1-\phi$.

Every y_j in line 7 can be viewed as a random point in P, the distribution of values for Y. The ranges we estimate are one-sided intervals ((γ,∞) for any $\gamma \in \mathbb{R}$ and they have VC-dimension $\nu = 1$). If we let $\kappa = O((1/\varepsilon^2)\ln(1/\phi))$, DTS gets exactly an ε-sample and guarantees that $|v/\kappa - \Pr[Y > \gamma]| \leq \varepsilon$ with probability at least $1-\phi$. ∎

B. Deterministic Distributed ε-Sample

The sizes of samples in RDεS could be large, especially for small ε and ϕ values, which drive up the communication cost (measured in bytes). We introduce another sampling algorithm, the *d*eterministic *d*istributed ε-*s*ample (DDεS) method, to address this problem, which is shown in Figure 6.

Algorithm DDεS $(c_1,\ldots,c_g,H,t,\varepsilon,\phi)$
1. $X_i = X_{i,t}$, $Y = \sum_{i=1}^{g} X_i$, $S_i = \emptyset$, $v = 0$;
2. $\varepsilon' = \varepsilon/g$, $\kappa = 1/\varepsilon'$;
3. for $i = 1,\ldots,g$
4. $\quad c_i$ selects κ evenly-spaced $x_{i,j}$'s from X_i into S_i, s.t. $S_i = \{x_{i,1},\ldots,x_{i,\kappa}\}$, and $\int_{x=x_{i,j}}^{x_{i,j+1}} Pr[X_i = x]dx = \varepsilon'$;
5. $\quad c_i$ sends S_i to H;
6. let $(1,\ldots,\kappa)^g$ define a g-dimensional space where each dimension takes values $\{1,\ldots,\kappa\}$;
7. for each $u \in (1,\ldots,\kappa)^g$ //u is a vector of g elements
8. $\quad$ if $(\sum_{i=1}^{g} x_{i,u_i} > \gamma)$ $v = v+1$;
9. if $(v/\kappa^g > \delta)$ H raises an alarm;
10. else H raises no alarm;

Fig. 6. The DDεS method.

Let $\tilde{X}_i$ represent S_i in the DDεS algorithm. Clearly, $\tilde{X}_i$ approximates X_i. Let $\tilde{Y} = \sum_{i=1}^{g} \tilde{X}_i$, i.e., for any $u \in (1,\ldots,\kappa)^g$ (as in lines 6-8) insert $\sum_{i=1}^{g} x_{i,u_i}$ into $\tilde{Y}$, by the construction of the DDεS, it is easy to see that:

$$\Pr[\tilde{Y} > \gamma] = v/\kappa^g. \tag{17}$$

To analyze its error, consider the distribution $Y_{\neq j} = \sum_{i=1, i\neq j}^{g} X_i$. Note that $Y = Y_{\neq j} + X_j$. We can claim the following about the random variable $\tilde{Y}_j = Y_{\neq j} + \tilde{X}_j$:

Lemma 1 *If $\tilde{X}_j$ is an ε-sample of $(X_j,\mathcal{I})$ then $|\Pr[\tilde{Y}_j > \gamma] - \Pr[Y > \gamma]| \leq \varepsilon$ with probability 1.*

Proof: The distribution of the random variable $\tilde{Y}_j$ has two components $Y_{\neq j}$ and $\tilde{X}_j$. The first has no error, thus,

$$\Pr[\tilde{Y}_j > \gamma] = \frac{1}{|\tilde{X}_j|} \sum_{x \in \tilde{X}_j} \Pr[x + Y_{\neq j} > \gamma]$$

Each $x \in \tilde{X}_j$ shifts the distribution of the random variable $Y_{\neq j}$, so part of that distribution that is greater than γ for $x_i \in \tilde{X}_j$ will also be greater than γ for $x_{i+1} \in \tilde{X}_j$ (since $x_{i+1} > x_i$ by definition). Let $y_i = \gamma - x_i$ denote the location in the distribution for $Y_{\neq j}$ where x_i causes $y \in Y_{\neq j}$ to have $\tilde{Y}_j > \gamma$. Now for $y \in [y_i, y_{i+1}]$ has $y + x_l \leq \gamma$ if $l < i$ and $y + x_l > \gamma$ if $l \geq i$. So $y \in [y_i, y_{i+1}]$ only has error in $\Pr[y + x > \gamma]$ (where x is either drawn from X_j or $\tilde{X}_j$) for $x \in [x_i, x_{i+1}]$. Otherwise, for $x \in [x_l, x_{l+1}]$, for $l < i$ has $\Pr[y + x > \gamma] = 0$ and for $x \in [x_l, x_{l+1}]$, for $l > i$ has $\Pr[y + x > \gamma] = 1$. Since for any i $\int_{x=x_i}^{x_{i+1}} \Pr[X_j = x] \leq \varepsilon$ (because $\tilde{X}_j$ is an ε-sample of $(X_j,\mathcal{I})$), we observe that:

$$\int_{y=y_i}^{y_{i+1}} \Pr[Y_{\neq j} = y] \tfrac{1}{|\tilde{X}_j|} \textstyle\sum_{x \in \tilde{X}_j} |\Pr[y + x > \gamma] - \Pr[y + X_j > \gamma]| \, dy$$

$$\leq \varepsilon \int_{y=y_i}^{y_{i+1}} \Pr[Y_{\neq j} = y] dy.$$ Thus we use that

$$\Pr[\tilde{Y}_j > \gamma] = \int_y \Pr[Y_{\neq j} = y] \frac{1}{|\tilde{X}_j|} \sum_{x \in \tilde{X}_j} \Pr[y + x > \gamma] dy$$

to conclude that

$$\left| \Pr[Y > \gamma] - \Pr[\tilde{Y}_j > \gamma] \right| \leq \textstyle\sum_{i=0}^{|\tilde{X}_j|} \varepsilon \int_{y=y_i}^{y_{i+1}} \Pr[Y_{\neq j} = y] dy \leq \varepsilon.$$

∎

This bounds the error on Y with $\tilde{Y}_j$ where a single X_j is replaced with $\tilde{X}_j$. We can now define $(\tilde{Y}_j)_l = \tilde{Y}_j - X_l + \tilde{X}_l = \sum_{i=1 \neq j,l}^{g} X_i + \tilde{X}_j + \tilde{X}_l$. And then apply Lemma 1 to show that if $\tilde{X}_l$ is an ε-sample of $(X_l,\mathcal{I})$ then

$$|\Pr[(\tilde{Y}_j)_l > \gamma] - \Pr[\tilde{Y}_j > \gamma]| \leq \varepsilon.$$

We can apply this lemma g times, always replacing one X_i with $\tilde{X}_i$ in the approximation to Y. Then the sum of error is at most εg. This implies the following theorem.

Theorem 2 *If for each c_i constructs $\tilde{X}_i$ as an (ε/g)-sample for $(X_i,\mathcal{I})$ then for any γ $|\Pr[\tilde{Y} > \gamma] - \Pr[Y > \gamma]| \leq \varepsilon$ with probability* 1.

Finally, by the definition of ε-samples on one-sided intervals (refer to (16) and the fact that in our case $\mathcal{I}$ consists of (γ,∞)'s), it is easy to see that:

Lemma 2 *Using g/ε evenly spaced points, each S_i in* DDεS *gives $\tilde{X}_i$ that is an ε/g-sample of $(X_i,\mathcal{I})$.*

Combining with (17), we have:

Corrolary 1 DDεS *gives* $|\Pr[\tilde{Y} > \gamma] - \Pr[Y > \gamma]| \leq \varepsilon$ *with probability 1 in* g^2/ε *bytes.*

A randomized improvement. We can improve the analysis slightly by randomizing the construction of the α-samples for each X_i. We choose $x_{i,1} \in \tilde{X}_i$ (the smallest point) to be at random so that $\Pr[x_{i,1} = x] = \frac{1}{\alpha}\Pr[X_i = x \mid x \leq x_\alpha]$ where x_α is defined so $\int_{x=-\infty}^{x_\alpha} \Pr[X_i = x]dx = \alpha$. Then each $x_{i,j}$ still satisfies that $\int_{x \in x_{i,j}}^{x_{i,j+1}} \Pr[X_i = x]dx = \alpha$. This keeps the points evenly spaced, but randomly shifts them.

Now we can improve Theorem 2 by modifying the result of Lemma 1. We can instead state that the error caused by $\tilde{X}_i$

$$H_i = (\Pr[\tilde{Y}_j > \gamma] - \Pr[Y > \gamma]) \in [-\alpha, \alpha].$$

Because of the random shift of $\tilde{X}_i$ places each $x_{i,j} \in \tilde{X}_i$ with equal probability as each point it represents in X_i, then for $I \in \mathcal{I}$ we have that

$$\mathbf{E}\left[\frac{|I \cap \tilde{X}_i|}{|\tilde{X}_i|}\right] = \mathbf{E}\left[\frac{|I \cap X_i|}{|X_i|}\right]$$

and hence for any γ $\mathbf{E}[\Pr[\tilde{Y}_j > \gamma]] = \mathbf{E}[\Pr[Y > \gamma]]$. Thus $\mathbf{E}[H_i] = 0$ and for all i $\Delta = \max\{H_i\} - \min\{H_i\} \leq 2\alpha$. Since the H_i are independent, we can apply a Chernoff-Hoeffding bound to the error on $\tilde{Y}$. So,

$$\Pr[|\Pr[\tilde{Y} > \gamma] - \Pr[Y > \gamma]| \geq \varepsilon] = \Pr[|\sum_{i=1}^{g} H_i| \geq \varepsilon]$$
$$\leq 2\exp(-2\varepsilon^2/(g\Delta^2)) \leq 2\exp(-\varepsilon^2/(2g\alpha^2)) \leq \phi,$$

when $\alpha \leq \varepsilon/\sqrt{2g\ln(2/\phi)}$. This implies that:

Theorem 3 *If each* $\tilde{X}_i$ *is of size* $(1/\varepsilon)\sqrt{2g\ln(2/\phi)}$ *and is randomly shifted, for any* γ

$$\Pr[|\Pr[\tilde{Y} > \gamma] - \Pr[Y > \gamma]| < \varepsilon] > 1 - \phi.$$

This gives a better bound when the acceptable failure probability ϕ satisfies $2\ln(2/\phi) < g$. We can modify DDεS according to Theorem 3 to get the αDDεS method:

Corrolary 2 αDDεS *guarantees* $\Pr[|\Pr[\tilde{Y} > \gamma] - \Pr[Y > \gamma]| < \varepsilon] > 1 - \phi$ *for any* $\varepsilon, \phi, \gamma$ *in* $(g/\varepsilon)\sqrt{2g\ln(2/\phi)}$ *bytes.*

C. Practical Improvements

Whenever a sample is required at any time t, for both RDεS and DDεS algorithms when the local sample size $|S_i|$ at t has exceeded the size required to represent the distribution X_i, client c_i simply forwards X_i to the server and the server can generate the sample for X_i himself. This is a simple optimization that will minimize the communication cost.

For the DDεS algorithm (in both its basic version and the random-shift version), a drawback is that its computation cost might become expensive for larger sample size or a large number of clients. In particular, executing its lines 7-10 requires the calculation of κ^g sums. In practice, however, we have observed that the DDεS algorithm can still give accurate estimation if we test only a small, randomly selected subset of possible combinations of local samples, instead of testing all κ^g combinations, i.e., in line 7, we randomly select $m < \kappa^g$ such u's and in line 9 we test v/m instead.

VI. Extension

A. Weighted Constraint

Suppose the user is interested at monitoring $Y = \sum_{i=1}^{g} a_i X_i$, for some weights $\{a_1, \ldots, a_g\}$, $\forall a_i \in \mathbb{R}^+$. All of our results can be easily extended to work for this case. The *Improved* and *Iadaptive* methods can be adapted based on the observations that: 1) $\mathbf{E}(Y) = \sum_{i=1}^{g} a_i \mathbf{E}(X_i)$ and $\mathrm{Var}(Y) = \sum_{i=1}^{g} a_i^2 \mathrm{Var}(X_i)$; 2) $M(\beta) = \prod_{i=1}^{g} M_i(a_i\beta)$. The RD$\varepsilon$S and DD$\varepsilon$S algorithms can also be easily adapted. For any sample j, instead of checking if $\sum_{i=1}^{g} x_{i,j} > \gamma$, they check if $\sum_{i=1}^{g} a_i x_{i,j} > \gamma$, in line 7 and 8 of Figures 5 and 6 respectively. The exact methods can also be extended easily. The discrete case is trivial, and the continuous case leverages on the observation that $\varphi(\beta) = \prod_{i=1}^{g} \varphi(a_i\beta)$.

B. Handling Multiple (γ, δ) Thresholds

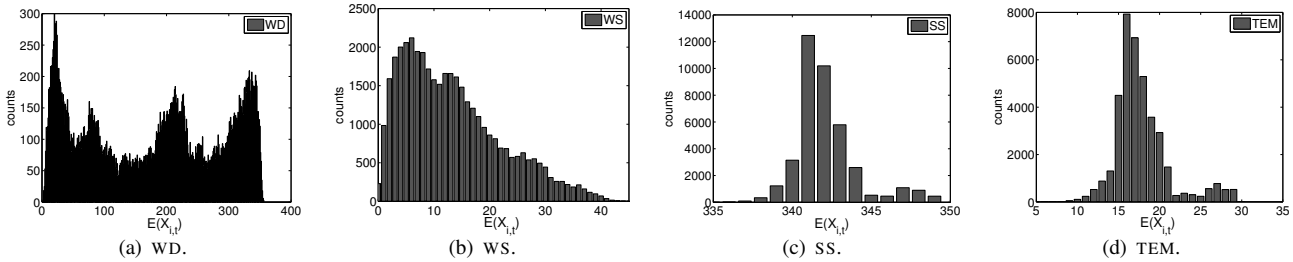
The other nice aspect of RDεS and DDεS is that after the server has gathered the samples S_i's from all clients and he wants to check another threshold pair (γ', δ'), he already has sufficient information. H re-executes lines 6-9 of RDεS or lines 6-10 of DDεS, with the new threshold pair (γ', δ'). The estimation of $\Pr[Y > \gamma']$ is again within ε of δ' with at least probability $1 - \phi$ and 1 for RDεS and DDεS respectively, i.e., the same error ε and the failure probability ϕ (or 0) cover all possible pairs (γ, δ) simultaneously in RDεS (or DDεS). This is especially useful if there was a continuous set of threshold pairs $\Gamma \times \Delta$ such that any violation of $(\gamma, \delta) \in \Gamma \times \Delta$ should raise the alarm. Then RDεS and DDεS are sufficient to check all of them, and are correct within ε with probability at least $(1 - \phi)$ and 1, respectively, without additional costs.

This also means that RDεS delivers stronger guarantee than the basic random sampling method in Appendix A. For the basic random sampling method approach, a second pair of thresholds (γ', δ') is a separate, but dependent problem. We can also estimate $\Pr[Y > \gamma'] > \delta'$ with ε-error with failure probability ϕ using the same sample as we used for estimating $\Pr[Y > \gamma] > \delta$. But now the probability that either of the thresholds has more than ε error is greater than ϕ. Using union bound, we need a sample size of about $O(\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon\phi})$ from each client to monitor $\frac{1}{\varepsilon}$ pairs of thresholds simultaneously, which is more than the sample size $O(\frac{1}{\varepsilon^2}\log\frac{1}{\phi})$ required by RDεS.

Small additional samples are also required for αDDεS to monitor multiple pairs of thresholds simultaneously.

VII. Experiments

All algorithms were implemented in C++. We used the GMP library when necessary in calculating the moment generating function $M_i(\beta)$. We simulated the distributed clients and the server, and executed all experiments in a Linux machine with an Intel Xeon E5506 cpu at 2.13GHz and 6GB memory. Since the flat model is used, server-to-client communication is *broadcast* and client-to-server communication is *unicast*. The server-to-client broadcast counts as one message, regardless the number of clients. Every client-to-server transmission is one separate message, which may contain multiple values or a pdf. Score and probability values are both 4 bytes.



(a) WD. (b) WS. (c) SS. (d) TEM.

Fig. 7. Distributions of $\mathbf{E}(X_{i,t})$ for WD, WS, SS, and TEM, where $i \in [1, g]$ and $t \in [1, T]$.

Datasets and setup. We used real datasets from the SAMOS project [27]. Raw readings from the research vessel *Wecoma* were obtained which consists of approximately 11.8 million records observed during a 9 month interval in 2010, from March to November. Each record consists of the current time and date, and the wind direction (WD), wind speed (WS), sound speed (SS), and temperature (TEM) measurements which are observed roughly every second (sometimes in less than a second). The wind direction measures the directional degree of the wind. The wind speed and sound speed are measured in meters per second and the temperature is in degrees Celsius. We observed that some measurements were erroneous or missing, e.g., a temperature of 999 or -999 degrees Celsius. Currently in SAMOS, to reduce communication and processing costs, records are grouped every τ consecutive seconds (the *grouping interval*), then replaced by one record taking the average readings of these records on each measurement respectively, which obviously loses a lot of useful information.

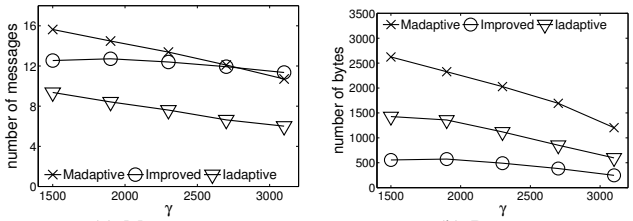
Instead, we derive pdfs (one per measurement) for records in one grouping interval and assign these pdfs to an attribute-level probabilistic tuple. There are different ways in how to derive a pdf for a measurement attribute, for example, [6], [7], [18], which is not the focus of this work. Without loss of generality and to ease the presentation, we simply generate a discrete pdf based on the frequencies of distinct values for a given measurement attribute: the probability of a distinct value is proportional to its frequency over the total number of records in the current grouping interval.

Four measurements lead to four datasets WD, WS, SS, and TEM, each with one probabilistic attribute. We were unable to obtain additional datasets of large raw readings from other research vessels, since in most cases they did not keep them after reporting the average readings per grouping interval. As a result, we simulate the effect of having multiple distributed vessels by assigning to each vessel tuples from a given dataset. Tuples are assigned in a round robin fashion to ensure and preserve the temporal locality of observed measurements.

The default values of key parameters are: $\tau = 300$, $g = 10$, $\delta = 0.7$, and γ is set to a value for a given dataset such that over all T instances, there should be approximately 30% alarms raised by an exact algorithm. The domains (in $\mathbb{R}$) of WD, WS, SS, and TEM are $[0, 359]$, $[0, 58.58]$, $[335.25, 355.9]$, and $[5.88, 41.3]$ respectively. These datasets also give us quite different distributions, allowing us to investigate different algorithms thoroughly. To illustrate this, we plot the distributions of $\mathbf{E}(X_{i,t})$ where $i = [1, g]$ and $t = [1, T]$ in the default setup in Figure 7. $\mathbf{E}(X_{i,t})$ also presents interesting (but quite different) temporal patterns and significant temporal changes in 4 datasets, which is also quite natural given that they precisely represent the large, real raw readings of different measurements at sea for a long period. Due to the space constraint, we omit these figures. That said, the default γ value is $230g$, $17g$, $343g$, and $19g$ for WD, WS, SS, and TEM. $X_{i,t}$ also has quite different sizes in 4 datasets. Under the default setup, the average size of $X_{i,t}$ is 41.15, 204.84, 20.5, and 20.98 for WD, WS, SS, and TEM respectively (they also change when we vary τ, obviously). Under the default setup, $T = 3932$.

For each experiment, we vary one of the key parameters while keeping the others fixed at their default values. For any sampling method, the default *sample size per client* is $\kappa = 30$. In the *Iadaptive* method, $k = 0.3T$ by default. For communication costs and running time, since T may vary, we report *the average cost of one time instance* which is obtained by dividing the corresponding total cost by T. Note that, we calculate the total running time by counting the server's running time plus the maximum running time of one client at each time instance. This ensures that the average running time reflects the expected *response time* at each round (since clients are running in parallel at distributed sites).

When most $X_{i,t}$ have large variances, sampling methods have the worst approximations. In our datasets, $\mathrm{Var}(X_{i,t})$ in WD are consistently large (much larger than other datasets) which least favors our methods. WD also has a medium average distribution size and a wide range of values (which makes it the most interesting for a monitoring problem). Thus, we use WD as the default dataset. For our problem, the naive solution is to run EXACTD every time instance, which is clearly much worse than the two baseline methods, *Madaptive* and *Markov*. Between the two, *Madaptive* is always better. Hence, we only show the results from *Madaptive* as the competing baseline.



(a) Messages. (b) Bytes.

Fig. 8. Communication: vary γ.

Effect of γ. Figure 8 shows the communication costs of *Madaptive*, *Improved*, and *Iadaptive* when we vary γ from 1500 to 3100. Both the number of messages and bytes reduce

for all algorithms while γ increases, since probabilistic tail bounds become tighter for larger γ values. Nevertheless, Figure 8(a) indicates that *Iadaptive* communicates the least number of messages, and Figure 8(b) shows that *Improved* sends the least number of bytes. *Improved* employs the most sophisticated combination of various lower and upper bounds (on both sides of $\mathbf{E}(Y)$), thus it has the largest number of "certain" instances where retrieving $X_{i,t}$'s can be avoided, which explains its best communication cost in bytes. Furthermore, it maintains low bytes for all γ values (a wide range we have tested), meaning that its pruning is effective on both sides of $\mathbf{E}(Y)$. However, *Improved* does require at least one, to a few, message(s) per client at every time instance, as shown in Figure 8(a). When reducing the number of messages is the top priority, *Iadaptive* remedies this problem. Figure 8(a) shows in most cases, it uses only half to one-third number of messages compared to *Madaptive* and *Improved*. In fact, it sends less than one message per client per time instances in most cases.

Figure 9 shows the response time of these methods when γ varies. Clearly, all methods take less time as γ increases, since there are less number of instances where they need to call the EXACTD method (which is costly). *Improved* and *Iadaptive* are much more efficient than *Madaptive*. The dominant cost in *Madaptive* and *Improved* is the calls to EXACTD, while the dominant cost in *Iadaptive* is the calculation of the moment generating function at the client. This explains why the response time of both *Madaptive* and *Improved* improves at a faster pace than that in *Iadaptive* when γ increases, since this mainly reduces the number of calls to EXACTD, but *Iadaptive* still needs to calculate moment generating functions. Nevertheless, *Iadaptive* is still more efficient than *Madaptive* in all cases. When $\gamma = 3100$, *Iadaptive* takes less than 0.001 second, and *Improved* takes close to 0.0003 second.

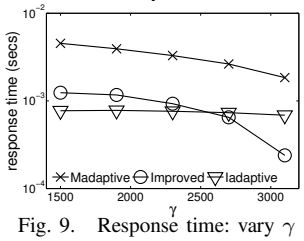


Fig. 9. Response time: vary γ

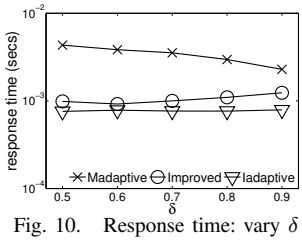
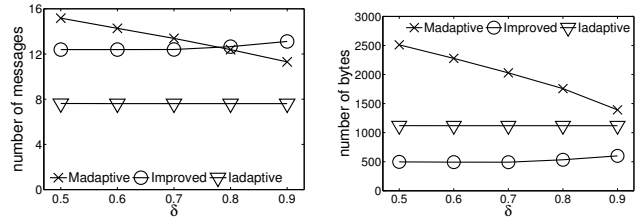


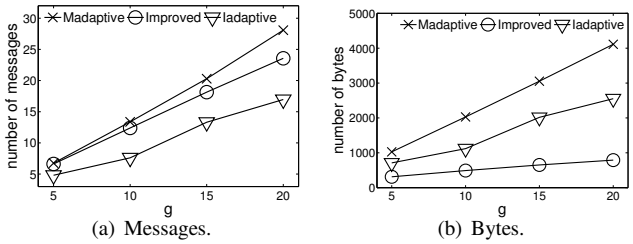
Fig. 10. Response time: vary δ



(a) Messages. (b) Bytes.

Fig. 11. Communication: vary δ.

Effect of δ. When δ changes from 0.5 to 0.9 in Figure 11, *Madaptive* benefits the most where both its messages and bytes are decreasing, since its global constraint is linearly dependent on δ, leading to a linearly increasing global constraint. Nevertheless, *Iadaptive* still uses much fewer messages and bytes than *Madaptive*, and *Improved* uses the least number of bytes, in all cases. In terms of the response time, Figure 10 shows that their trends are similar to what we have observed in Figure 9: *Improved* and *Iadaptive* are more efficient than *Madaptive*.



(a) Messages. (b) Bytes.

Fig. 12. Communication: vary g.

Effect of g. We next investigate the impact of the number of clients; Figure 12 shows the results on communication. Not surprisingly, we see a linear correlation between the number of messages and g in Figure 12(a) where *Iadaptive* consistently performs the best. Figure 12(b) shows that all methods send more bytes as g increases, nevertheless, both *Improved* and *Iadaptive* send many fewer bytes than *Madaptive*.

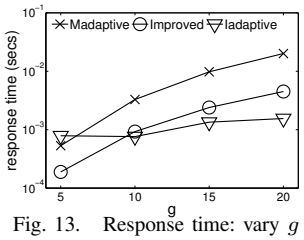


Fig. 13. Response time: vary g

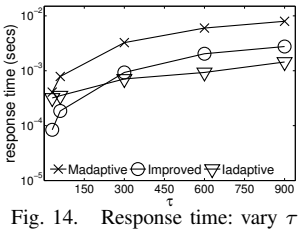
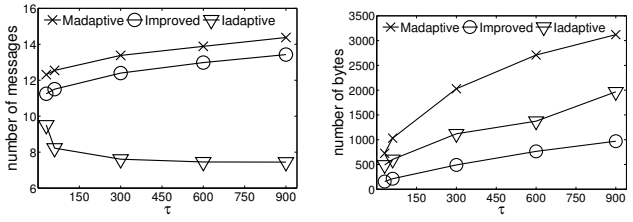


Fig. 14. Response time: vary τ

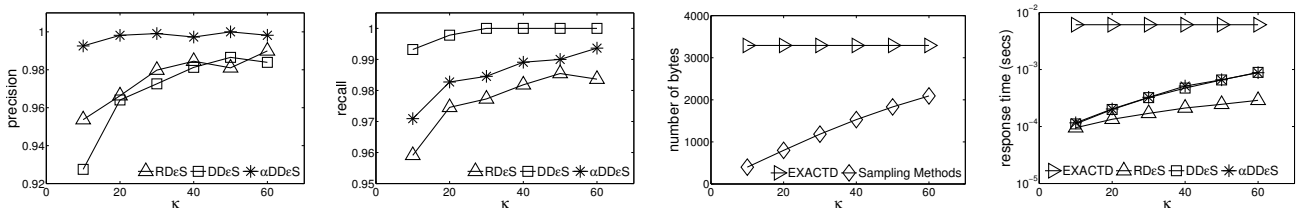
All methods take longer to respond on average in Figure 13 for larger g values, due to the increasing cost in executing EXACTD. However, the cost of *Madaptive* increases at a faster pace than other methods, since it makes many more calls to EXACTD. On the other hand, both *Improved* and *Iadaptive* are highly efficient, even though EXACTD becomes quite expensive for large g values, since they avoid calling EXACTD in most cases. Even when $g = 20$, both of them only take less than 0.005 seconds to respond.



(a) Messages. (b) Bytes.

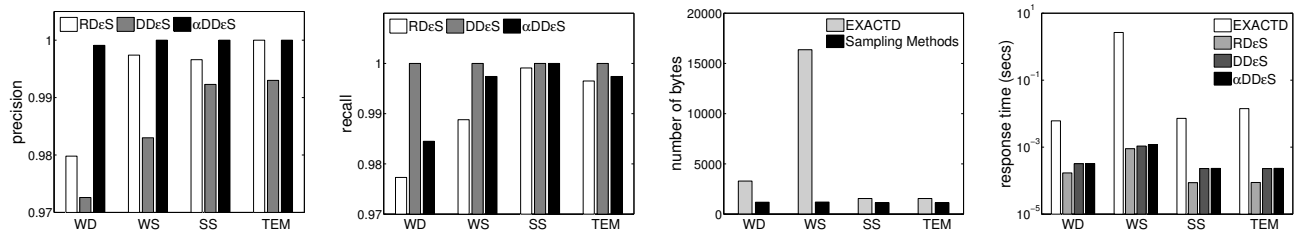
Fig. 15. Communication: vary τ.

Effect of τ. When τ changes, Figure 15 shows the communication of various methods. Figure 15(a) shows that *Iadaptive* reduces messages when τ increases, while the other two methods sends more messages. Larger τ values lead to larger pdfs, i.e., more values in $X_{i,t}$ but each taking smaller probability value, which make the bounds based on the moment generating functions tighter. But other bounds become looser, since $X_{i,t}$ becomes relatively more uniform for larger pdfs. Hence, *Iadaptive*, relying only the moment generating function bounds, is performing better for larger τ values, while others degrade slowly, in terms of number of messages. In terms of number of bytes, all methods send more bytes for larger τ values, which is easy to explain: whenever a call to EXACTD is



(a) Precision. (b) Recall. (c) Communication: bytes. (d) Response time.

Fig. 16. Performance of the sampling methods: vary κ (sample size per client).



(a) Precision. (b) Recall. (c) Communication: bytes. (d) Response time.

Fig. 17. Performance of the sampling methods: vary datasets.

necessary, $X_{i,t}$'s need to be communicated and they become larger for larger τ values. Nevertheless, both *Iadaptive* and *Improved* are still much more effective than *Madaptive*, e.g., even when $\tau = 900$ (15 minutes grouping interval), *Improved* only sends about 1000 bytes per time instance. Figure 14 shows that all methods take longer time to respond, since EXACTD becomes more expensive due to the increase in the pdf size. *Improved* and *Iadaptive* are clearly faster than *Madaptive*. When $\tau = 900$, both of them still only take less than 0.005 second to respond.

Sampling methods. The RDεS method offers similar (and even stronger, see Section VI-B) theoretical guarantee than the basic random sampling method in Appendix A. Its performance in practice is also better. Thus, we focus on studying RDεS, DDεS and its randomized improvement, denoted as αDDεS. Note that we have incorporated the practical improvements introduced in Section V-C; $m = 2$ for both DDεS and αDDεS (which has achieved sufficient accuracy for both methods).

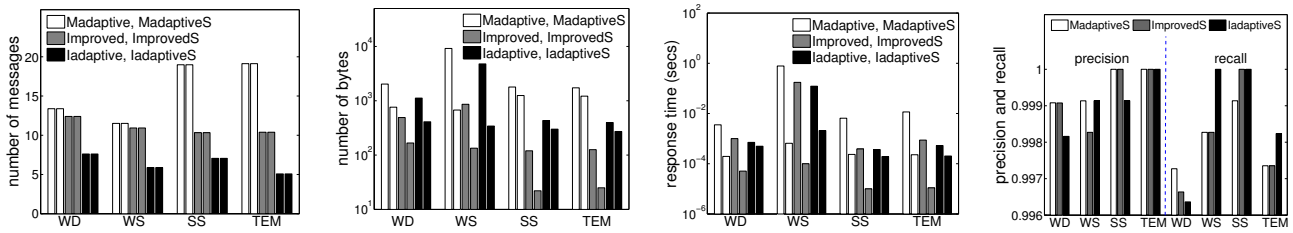
In this set of experiments, we compare sampling methods against the EXACTD method by running them over all T time instances. We use the *precision* and *recall* metrics to measure the approximation quality of sampling methods. Here, *precision* and *recall* are calculated w.r.t. the set of true alarms among the T instances, i.e., suppose there are a set A of 300 true alarms over $T = 1000$ time instances; an approximate method may raise a set B of 295 alarms out of the 1000 instances, with 5 false positives and 10 false negatives. Then, its precision is $290/295$ and its recall is $290/300$.

Figures 16(a) and 16(b) show that all sampling methods improve their precisions and recalls when the sample size per client κ increases. Theoretically, both αDDεS and DDεS should always have better precisions and recalls than RDεS given the same sample size. However, since we have incorporated the practical improvement to αDDεS and DDεS to cut down their computation cost, RDεS might perform better in some cases. Nevertheless, Figures 16(a) and 16(b) show that in practice, given the same sample size, αDDεS achieves the best precision while DDεS has the best recall; and αDDεS always outperforms RDεS. When $\kappa = 30$, they have achieved a precision and recall close to or higher than 0.98. The sample size required in practice to achieve good accuracy for all sampling methods is clearly much less than what our theoretical analysis has suggested. This is not surprising, since theoretical analysis caters for some worst cases that rarely exist in real datasets. In all remaining experiments, we use $\kappa = 30$ by default.

Figures 16(c) and 16(d) show that sampling methods result in clear savings in communication (bytes) and computation costs. They are especially useful in saving response time, which is 1-2 orders magnitude faster than EXACTD and the gap expects to be even larger for larger pdfs or more clients. Note that all sampling methods have the same communication cost given the same sample size (hence we only show one line for all of them in Figure 16(c)). Also, they result in the same number of messages as EXACTD.

We have also tested the sampling methods using all 4 datasets under the default setup, and the results are shown in Figure 17; the trends are clearly similar to what we have observed in Figure 16. Note that WS has quite large pdfs, thus, EXACTD becomes very expensive on this dataset in terms of both bytes communicated and running time, making sampling methods more valuable under these situations (several orders of magnitude more efficient than EXACTD).

Integrated methods. Lastly, we integrate our sampling methods with *Madaptive*, *Improved*, and *Iadaptive* to derive the *MadaptiveS*, *ImprovedS*, and *IadaptiveS* methods, where in any time instance a call to EXACTD is replaced with a call to a sampling method. In particular, we use αDDεS as the sampling method since it achieves the best trade-off between efficiency and accuracy as shown in last set of experiments. We tested these methods, along with their exact versions, on all datasets using the default setup. The results are shown in Figure 18. The trends are clear: 1) The approximate versions have outperformed the corresponding exact versions in both communication and response time consistently; 2) Our methods have outperformed the baseline methods, *Madaptive* and *MadaptiveS* in all cases, by significant margins; 3)



(a) Communication: messages. (b) Communication: bytes. (c) Response time. (d) Precision and recall.

Fig. 18. Performance of all methods: vary datasets.

Iadaptive and *IadaptiveS* are the best exact and approximate methods in saving the number of messages, and *Improved* and *ImprovedS* are the best methods in saving the number of bytes. For example, *Iadaptive* and *IadaptiveS* use less than one message per client per time instance on all datasets; *Improved* and *ImprovedS* use less than 1000 and 100 bytes per time instance respectively on WS that has an average pdf size of 204.84; 4) *Iadaptive*, *IadaptiveS*, *Improved* and *ImprovedS* are efficient to run. In particular, *IadaptiveS* and *ImprovedS* are extremely fast, e.g., Figure 18(c) shows that they take less than 10^{-3} and 10^{-4} seconds to respond, respectively, in all datasets. 5) αDDεS is highly effective. Figure 18(d) shows that *MadaptiveS*, *ImprovedS*, and *IadaptiveS* have almost perfect precisions and recalls on all datasets (more than 0.996 in all cases). Note that their precisions and recalls are clearly better than using sampling methods on every time instance, since many alarms will already be caught certainly by *Madaptive*, *Improved*, and *Iadaptive*, only a tiny fraction of undecided cases will be then decided by the sampling methods.

VIII. Related Work

To our knowledge, aggregate constraint monitoring on distributed data with uncertainty has not been explored before.

That said, ranking and frequent items queries were studied on distributed probabilistic data in [20], [34]. Monitoring centralized uncertain data for top-k and similarity queries were studied in [11], [17], [35]. On the other hand, due to their importance and numerous applications, constraint and function monitoring with thresholds on deterministic distributed data were examined extensively, e.g., [4], [12], [16], [19], [23], [29]. In our study, we have leveraged on the adaptive thresholds algorithm for the deterministic (sum) constraint monitoring from [16]. This choice is independent from the design of our adaptive algorithms for the DPTM problem: any adaptive algorithms for the (sum) constraint monitoring in deterministic data can be used in our *Iadaptive* method.

Our study is also related to aggregation queries in probabilistic data, e.g., [14], [15], [22], [26], [30], [32]. However, monitoring both score and probability thresholds on aggregate constraints continuously over distributed probabilistic data is clearly different from these studies. Probabilistic threshold queries in uncertain data are also relevant [3], [5], [24], [25], as they are also concerned with the probability thresholds on the query results, but they mostly focus on one-shot query processing over centralized, offline probabilistic data.

Lastly, the basic sampling method MRS in Appendix A can be viewed as a standard extension of the random sampling technique [21], [33]. The RDεS and DDεS methods are related to VC-dimensions and ε-samples [33] as we already pointed out. The design principle behind the RDεS method, i.e., using a Monte Carlo approach, has also been used for general query processing in probabilistic data (e.g., [9], [13], [24] and more in [31]). The DDεS and αDDεS are based on several intriguing insights to the distinct properties of our problem.

IX. Conclusion

This paper presents a comprehensive study on the threshold monitoring problem over distributed probabilistic data. We focus on the sum constraint and explore a number of novel methods that have effectively and efficiently reduced both the communication and computation costs in monitoring the user-specified constraint continuously. Extensive experiments demonstrate the excellent performance and significant savings achieved by our methods, compared to the baseline algorithms. Many interesting directions are open for future work. Examples include but not limit to how to extend our study to the hierarchical model that is often used in a sensor network, how to monitor more sophisticated constraints (beyond sum and linear combinations of sum constraints) continuously, and how to handle the case when data from different sites are correlated.

X. Acknowledgment

Mingwang Tang, Feifei Li and Jeffrey Jestes were partially supported by NSF grants IIS-0916488 and IIS-1053979. We are grateful to Shawn R. Smith from the SAMOS project for providing the datasets and many useful feedback.

References

[1] P. Billingsley. *Probability and measure*. Wiley-Interscience, 1995.
[2] R. Cheng, D. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *SIGMOD*, 2003.
[3] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J. S. Vitter. Efficient indexing methods for probabilistic threshold queries over uncertain data. In *VLDB*, 2004.
[4] G. Cormode, S. Muthukrishnan, and K. Yi. Algorithms for distributed functional monitoring. In *SODA*, 2008.
[5] N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, 2004.
[6] A. Deshpande, C. Guestrin, and S. Madden. Using probabilistic models for data management in acquisitional environments. In *CIDR*, 2005.
[7] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *VLDB*, 2004.
[8] X. Dong, A. Y. Halevy, and C. Yu. Data integration with uncertainty. In *VLDB*, 2007.
[9] T. Ge, D. Grabiner, and S. B. Zdonik. Monte carlo query processing of uncertain multidimensional array data. In *ICDE*, 2011.
[10] L. Gruenwald, H. Chok, and M. Aboukhamis. Using data mining to estimate missing sensor data. In *ICDMW*, 2007.
[11] M. Hua and J. Pei. Continuously monitoring top-k uncertain data streams: a probabilistic threshold method. *DPD*, 26(1):29–65, 2009.

[12] L. Huang, M. Garofalakis, A. D. Joseph, and N. Taft. Communication-efficient tracking of distributed cumulative triggers. In *ICDCS*, 2007.
[13] R. Jampani, F. Xu, M. Wu, L. L. Perez, C. M. Jermaine, and P. J. Haas. MCDB: a monte carlo approach to managing uncertain data. In *SIGMOD*, 2008.
[14] T. S. Jayram, S. Kale, and E. Vee. Efficient aggregation algorithms for probabilistic data. In *SODA*, 2007.
[15] T. S. Jayram, A. McGregor, S. Muthukrishnan, and E. Vee. Estimating statistical aggregates on probabilistic data streams. In *PODS*, 2007.
[16] S. Jeyashanker, S. Kashyap, R. Rastogi, and P. Shukla. Efficient constraint monitoring using adaptive thresholds. In *ICDE*, 2008.
[17] C. Jin, K. Yi, L. Chen, J. X. Yu, and X. Lin. Sliding-window top-k queries on uncertain streams. In *VLDB*, 2008.
[18] B. Kanagal and A. Deshpande. Online filtering, smoothing and probabilistic modeling of streaming data. In *ICDE*, 2008.
[19] R. Keralapura, G. Cormode, and J. Ramamirtham. Communication efficient distributed monitoring of thresholded count. In *SIGMOD*, 2006.
[20] F. Li, K. Yi, and J. Jestes. Ranking distributed probabilistic data. In *SIGMOD*, 2009.
[21] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
[22] R. Murthy, R. Ikeda, and J. Widom. Making aggregation work in uncertain and probabilistic databases. *TKDE*, 23(8):1261–1273, 2011.
[23] C. Olston, J. Jiang, and J. Widom. Adaptive filters for continuous queries over distributed data streams. In *SIGMOD*, 2003.
[24] L. Perez, S. Arumugam, and C. Jermaine. Evaluation of probabilistic threshold queries in MCDB. In *SIGMOD*, 2010.
[25] Y. Qi, R. Jain, S. Singh, and S. Prabhakar. Threshold query optimization for uncertain data. In *SIGMOD*, 2010.
[26] R. Ross, V. S. Subrahmanian, and J. Grant. Aggregate operators in probabilistic databases. *J. ACM*, 52(1):54–101, 2005.
[27] SAMOS. Shipboard Automated Meteorological and Oceanographic System. http://samos.coaps.fsu.edu.
[28] A. D. Sarma, O. Benjelloun, A. Halevy, S. Nabar, and J. Widom. Representing uncertain data: models, properties, and algorithms. *The VLDB Journal*, 18(5):989–1019, 2009.
[29] I. Sharfman, A. Schuster, and D. Keren. A geometric approach to monitoring threshold functions over distributed data streams. In *SIGMOD*, 2006.
[30] M. A. Soliman, I. F. Ilyas, and K. C.-C. Chang. Probabilistic top-k and ranking-aggregate queries. *TODS*, 33(3):1–54, 2008.
[31] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.
[32] T. T. L. Tran, A. McGregor, Y. Diao, L. Peng, and A. Liu. Conditioning and aggregating uncertain data streams: Going beyond expectations. *PVLDB*, 3(1):1302–1313, 2010.
[33] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *The. of Prob. App.*, 16:264–280, 1971.
[34] S. Wang, G. Wang, and J. Chen. Distributed frequent items detection on uncertain data. In *ADMA*, 2010.
[35] H. Woo and A. K. Mok. Real-time monitoring of uncertain data streams using probabilistic similarity. In *RTSS*, 2007.

APPENDIX

A. The Random Sampling Approach

We first introduce the RS algorithm in Figure 19.

Algorithm RS $(c_1, \ldots, c_g, t, H, \varepsilon)$
1. let $X_i = X_{i,t}$, $Y = Y_t = \sum_{i=1}^{g} X_i$, $S = \emptyset$, $\kappa = 4/\varepsilon^2$;
2. for $i = 1, \ldots, g$
3. send *random sample* $S_i = \{x_{i,1}, \ldots, x_{i,\kappa}\}$ of X_i to H;
4. For any $j \in [1, \kappa]$, H inserts $\sum_{i=1}^{g} x_{i,j}$ into S;
5. let $s(\gamma)$ be the number of elements in S greater than γ;
6. return $\widehat{p}(\gamma) = s(\gamma) \cdot \frac{\varepsilon^2}{4}$;

Fig. 19. The RS estimator

Lemma 3 *The* RS *estimator satisfies* $\mathbf{E}(\widehat{p}(\gamma)) = \Pr[Y > \gamma]$, *and* $\Pr[|\widehat{p}(\gamma) - \Pr[Y > \gamma]| < \varepsilon] > \frac{3}{4}$.

Proof: Let $\varepsilon' = \varepsilon/2$, then κ in line 1 is $1/\varepsilon'^2$. Clearly, by lines 2-5, S is a random sample of Y with size $1/\varepsilon'^2$. Suppose Y's distribution is represented by a *multi-set* P of elements $P = \{y_1, \ldots, y_N\}$ for some imaginary, sufficiently large value $N \in \mathbb{Z}^+$. Let $r(\gamma)$ be the number of elements in P that is larger than γ, then $\Pr[Y > \gamma] = r(\gamma)/N$.

Let $p = 1/(\varepsilon'^2 N)$, we then define N i.i.d. random variables $Z_1, \ldots, Z_N$, such that $\Pr[Z_i = 1] = p$ and $\Pr[Z_i = 0] = 1-p$. We associate Z_i with $y_i \in P$. Then, S can be viewed as being created by the following process: for each $i \in [1, N]$, insert y_i into S if $Z_i = 1$. For any γ, $s(\gamma)$ in line 6 is a random variable determined by the number of elements in P larger than γ (each sampled with probability p) in S. There are precisely $r(\gamma)$ such elements in P, and we denote them as $\{y_{\ell_1}, \ldots, y_{\ell_{r(\gamma)}}\}$, where $y_{\ell_i} \in P$. This means that: $s(\gamma) = \sum_{i=1}^{r(\gamma)} Z_{\ell_i}$. Since each Z_i is a Bernoulli trial, $s(\gamma)$ is a Binomial distribution $B(r(\gamma), p)$. Immediately, $\mathbf{E}(s(\gamma)) = p \cdot r(\gamma)$. Hence, $\mathbf{E}(\widehat{p}(\gamma)) = \mathbf{E}(\varepsilon'^2 N \frac{s(\gamma)}{N}) = \frac{1}{p}\frac{p \cdot r(\gamma)}{N} = \Pr[Y > \gamma]$, and

$$\begin{aligned} \mathrm{Var}(\frac{s(\gamma)}{p}) &= \frac{1}{p^2}\mathrm{Var}(s(\gamma)) = \frac{1}{p^2} r(\gamma) p(1-p) \\ &< \frac{r(\gamma)}{p} = r(\gamma)\varepsilon'^2 N \le (\varepsilon' N)^2. \end{aligned}$$

Also, $\mathbf{E}(s(\gamma)/p) = r(\gamma)$. By Chebyshev's inequality: $\Pr[\left|\frac{s(\gamma)}{p} - r(\gamma)\right| \ge 2\varepsilon' N] \le \frac{1}{4}$, which implies that: $\Pr[\frac{1}{N}\left|\frac{s(\gamma)}{p} - r(\gamma)\right| \ge 2\varepsilon'] \le \frac{1}{4}$. Given $\varepsilon = 2\varepsilon'$ and $p = 1/(\varepsilon'^2 N)$, $\frac{s(\gamma)}{pN} = \frac{s(\gamma)\varepsilon^2}{4}$, we have $\Pr[\left|\frac{s(\gamma)\varepsilon^2}{4} - \Pr[Y > \gamma]\right| \ge \varepsilon] \le \frac{1}{4}$. Immediately, $\Pr[|\widehat{p}(\gamma) - \Pr[Y > \gamma]| < \varepsilon] > \frac{3}{4}$. ∎

We can boost up $\Pr[|\widehat{p}(\gamma) - \Pr[Y > \gamma]| < \varepsilon]$ to be arbitrarily close to 1 by the MRS (*m*edian RS) Algorithm in Figure 20.

Algorithm MRS $(c_1, \ldots, c_g, t, H, \varepsilon, \phi)$
1. run $8 \ln \frac{1}{\phi}$ independent instances RS $(c_1, \ldots, c_g, t, H, \varepsilon)$;
2. let $\widehat{p}_i(\gamma)$ be the ith RS's output for $i \in [1, 8 \ln \frac{1}{\phi}]$;
3. set I_i be 1 if $|\widehat{p}_i(\gamma) - \Pr[Y > \gamma]| < \varepsilon$, and 0 otherwise;
4. let I_j be the median of $I = \{I_1, \ldots, I_{8 \ln \frac{1}{\phi}}\}$;
5. return $\widehat{p}_j(\gamma)$;

Fig. 20. The MRS estimator

Theorem 4 MRS *returns* $\widehat{p}_j(\gamma)$ *s.t.* $\Pr[|\widehat{p}_j(\gamma) - \Pr[Y > \gamma]| < \varepsilon] > 1 - \phi$, *for any* $\varepsilon, \phi \in (0, 1)$*; it uses* $32\frac{g}{\varepsilon^2}\ln\frac{1}{\phi}$ *bytes.*

Proof: By Lemma 3, each I_i outputs 1 with probability at least $\frac{3}{4}$ in line 3 in Figure 20. Let $h = 8 \ln \frac{1}{\phi}$, by the common form of the Chernoff Bound [21], $\Pr[\sum_{i=1}^{h} I_i < \frac{h}{2}] < e^{-2h(\frac{3}{4} - \frac{1}{2})^2} = \phi$. $\Pr[\sum_{i=1}^{h} I_i < \frac{h}{2}]$ is exactly the probability that less than half of I_i's being 0. Since I_j is the median in I (line 4), there is at least $(1 - \phi)$ probability that $I_j = 1$. By line 3, in this case, we must have $|\widehat{p}_j(\gamma) - \Pr[Y > \gamma]| < \varepsilon$. The communication in bytes is straightforward. ∎

Lastly, if $\widehat{p}(\gamma)$ returned by MRS is greater than δ, H raises an alarm at t; otherwise no alarm is raised.