CLARO: Modeling and Processing of High-Volume Uncertain Data Streams

Thanh Tran, Liping Peng, Boduo Li, Yanlei Diao, Anna Liu[†]
Department of Computer Science [†]Department of Mathematics and Statistics
University of Massachusetts Amherst
{ttran, lppeng,boduo,yanlei}@cs.umass.edu [†] anna@math.cs.umass

ABSTRACT

Uncertain data streams, where data is *incomplete*, *imprecise*, and even *misleading*, have been observed in a variety of environments. Feeding uncertain data streams to existing stream systems can produce results of unknown quality, which is of paramount concern to monitoring applications. In this paper, we present the CLARO system that supports uncertain data stream processing for data that is naturally captured using *continuous random variables*. The CLARO system employs a unique data model that is flexible and allows efficient computation. Built on this model, we develop evaluation techniques for complex relational operators, including aggregates and joins, by exploring advanced statistical theory and approximation techniques. Our evaluation results show that our techniques can achieve high performance in stream processing while satisfying accuracy requirements, and these techniques significantly outperform a state-of-the-art sampling-based method. Furthermore, initial results of a case study show that our modeling and aggregation techniques can allow a tornado detection system to produce better quality results yet with lower execution time.

1. INTRODUCTION

Uncertain data streams, where data is *incomplete*, *imprecise*, and even *misleading*, have been observed in a variety of environments, such as sensor networks measuring temperature and light [8, 16], radio frequency identification (RFID) networks [18, 30], GPS systems [19], and weather radar networks [22]. As these data streams are collected by monitoring applications, they often undergo sophisticated query processing to derive useful high-level information. However, feeding uncertain data streams directly to existing stream systems can produce results of unknown quality. This issue is of paramount concern to monitoring applications that trigger actions based on the derived information.

Our work is particularly motivated by two emerging monitoring applications. The first is object tracking and monitoring using RFID readers [30]. Data streams from RFID readers are highly noisy due to the sensitivity of sensing to the orientation of reading and many environmental factors such as metal objects and interference. When such streams are used to support monitoring applications, for instance, to detect safety violations regarding flammable objects, the quality of the alerts raised is a critical issue to address.

The second application is severe weather monitoring [22], where meteorological data streams are collected from a weather radar network and processed in a real-time stream system to predict natural disasters such as tornados. Uncertainty in such data arises from environmental noise, device noise, and inaccuracies of various radar components. Such uncertainty can propagate all the way through the stream system, making the final detection results error-prone. Given the potential social impact, it is absolutely vital that the system capture the quality of the detection results.

In this paper, we address uncertain data stream processing for data that is naturally modeled using *continuous random variables*, such as many types of sensor data and financial data. Given such data, our work addresses relational query processing under uncertainty, with a focus on *aggregates* and *joins*. These complex operations are crucial to our target applications but have not been sufficiently addressed in prior work for continuous random variables. For each relational operator, our work aims to fully characterize the distribution of each result tuple produced from uncertain data. Such distributions, which we call *result tuple distributions*, can be used to return any statistics needed by the application, such as mean, variance, and confidence intervals. They also allow the stream system to feed these tuples as input to other operators and characterize the results of the further processing—it is evident that merely having statistics such as mean and variance for those tuples is not enough to do so.

1.1 Challenges

There are two fundamental challenges in uncertain data stream processing as described above: First, it is computationally difficult to obtain result distributions of aggregates and joins for input tuples modeled using continuous random attributes. Second, such computation must be performed efficiently for high-volume data streams. While approximation is a common approach to improving efficiency, the technique must be able to achieve a small bounded error while meeting performance requirements to be useful to monitoring applications.

Despite a large body of recent work on probabilistic data management, the above challenges have not been adequately addressed. Most probabilistic databases [1, 2, 4, 21, 25, 27, 31] and stream systems [5, 17, 20] model tuples using *discrete* random variables and evaluate queries using the possible worlds semantics. The continuous nature of our data, however, precludes the use of these techniques because the possible values of a continuous random variable are infinite and cannot be enumerated.

Recent work on uncertain data processing for continuous attributes has taken two approaches to handling aggregates: The integral-based approach [4] performs exact derivation of the result distribution, for instance, using a two-variable convolution algorithm for `sum`. While the result is accurate, the computation is too slow to be used in stream processing, as we shall show later in this paper. The sampling-based approach [14, 28] employs approximation by discretizing continuous distributions and sampling from the discretized distributions. However, for real-world data it is difficult, sometimes impossible, to find the right number of samples that guarantees both accuracy and efficiency, as we also show in performance evaluation of this paper. Joins of continuous random attributes have only been considered at the modeling level [28], lacking both implementation techniques and demonstrated performance for data streams.

1.2 Our Contributions

In this paper, we present the design, implementation, and evaluation of a probabilistic data stream system, which we call CLARO, that supports relational processing of uncertain data streams involving continuous random attributes. Our work has the following main contributions:

Data model. The foundation of CLARO is a flexible data model that allows efficient computation. Based on Gaussian Mixture distributions, this model subsumes the commonly used Gaussian distributions and can model arbitrarily complex real-world distributions. Moreover, it allows efficient computation based on powerful statistical techniques for continuous random variables and particular Gaussian properties. Our choice of data model stands in contrast to those based on histograms [14] and weighted particles [20] which indicate the use of samples in computation. Finally, our model has the potential to support a large set of relational operators. While our work focuses on two complex operators, aggregates and joins, an extension to other operators is sketched at the end of the paper.

Aggregates. Our data model empowers us to explore advanced statistical theory, such as characteristic functions, to obtain *exact result distributions* of aggregation at low computation cost. In particular, our algorithm for exact derivation completely eliminates the use of integrals, in contrast to the use of multiple integrals in [4]. However, the formulas for result distributions that the exact algorithm produces grow exponentially in the number of aggregated tuples. Hence, we provide two *approximation* methods to simplify the formulas for result distributions. These techniques, when combined, can satisfy arbitrary application-specified accuracy requirements while achieving high speed in stream processing.

Joins. Our data model also allows us to seek efficient and accurate techniques for joins. Depending on the application semantics, a join between a continuous random attribute and a deterministic attribute can often be modeled using an outer join over a probabilistic view. Our first technique supports such joins by using regression to efficiently construct the view and deriving a *closed-form solution* to join result distributions based on the view. Joins of continuous random attributes can also be modeled using the cross-product semantics. Our second technique supports such joins with not only result distributions but also an efficient filter that removes the join results with low existence probabilities.

Performance evaluation. We perform a thorough evaluation of our techniques for joins and aggregates, and compare them with sampling-based methods ([14] for aggregates and a home-grown method for joins). Results of this study demonstrate our advantages in both accuracy and speed over the sampling-based approach, due to the use of our data model and advanced techniques for continuous random variables.

We further perform a case study in the severe weather monitoring application, in which the CLARO system is used to process a raw trace collected from a real tornadic event. Our initial results show that fitting data to the CLARO model and using its aggregation techniques provides cleaner, more accurate data to the tornado detection algorithm, resulting in better detection results and faster computation.

The remainder of the paper is organized as follows. We describe our motivating applications more in §2. We present our data model in §3, and main techniques for aggregates and joins in §4 and §5. Evaluation results are described in §6. An extension of the CLARO system to a larger set of relational operators is outlined in §7. Finally, §8 covers related work and §9 concludes the paper.

2. MOTIVATING APPLICATIONS

In this section, we present two motivating applications.

2.1 Object Tracking and Monitoring

In the first application, radio frequency identification (RFID) readers are used to monitor a large area such as a warehouse, a retail store, or a library. RFID data is known to be highly noisy [18, 30]. The read rate of RFID readers is far less than 100% due to environment factors such as occluding metal objects, and interference. Moreover, mobile RFID readers may read objects from arbitrary angles, hence particularly susceptible to variable read rates. Our prior work [30] provides techniques to transform raw RFID readings into a stream of location tuples. Each location tuple contains (time, tag_id, x^p, y^p), where x^p and y^p denote the inferred location of the object which is probabilistic in nature.

Despite the data uncertainty, monitoring applications want to run queries over the location streams to derive high-level information. The first query illustrates the use in a fire monitoring application: "trigger an alert when a flammable object is exposed to a high temperature." This query takes two inputs: a location stream for flammable objects as described above, and a temperature stream from sensors placed in the same area (time, sensor_id, x, y, temp^p), where the temperature can be uncertain due to sensing noise. The query joins the location stream with the temperature stream based on the location. The query is written as if the location of an object and the temperature at a location were precise.

```
Q1:  Select Rstream(R.tag_id, R.x, R.y, T.temp)
     From   FlammableObject [Now] As R,
            Temp [Partition By sensor_id Rows 1] As T
     Where  T.temp > 60 ℃ and
            R.x = T.x and R.y = T.y
```

The second query detects violations of the shipping policy by the Food and Drug Administration: "foods with and without peanuts cannot be located closely in the supply chain." This query takes two location streams and checks for the proximity of two types of food. Again, this query is written as if the location of each object were certain.

```
Q2:  Select Rstream(R.tag_id, R.x, R.y, S.tag_id, S.x, S.y)
     From   PeanutFreeFood [Range 3 minutes] As R,
            PeanutFood [Range 3 minutes] As S
     Where  |R.x - S.x| < 3 ft and |R.y - S.y| < 3 ft
```

2.2 Hazardous Weather Monitoring

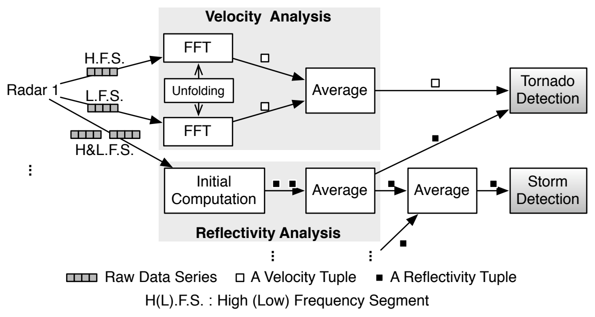


Figure 1: Simplified stream processing in CASA radar system

The CASA Research Center [3] is developing weather radar networks to detect hazardous weather events such as tornados and storms [22]. A four-radar testbed covering a 7,000 square km region has been deployed in southwestern Oklahoma, a region that receives an average of four tornado warnings and 53 thunderstorm warnings each year [22].

A CASA radar node rotates to scan. It sends around 2000 pulses per second, alternating between 47 high frequency pulses and 47 low frequency ones. For each pulse, the radar listens to 700 echoes reflected from the areas in increasing distance from the radar. Each echo consists of 4 32-bit raw floating numbers. Hence, the raw data is generated at a rate of 175Mb per second. In addition, the raw data is highly noisy due to electronic device noise, instability of transmit frequency, quality issues of the system clock, the positioner, and the antenna, and finally environmental noise.

The raw data streams from the radar nodes are fed to a real-time system for weather event detection. Such real-time stream processing is faced with two challenges: highly noisy data and high data volume, which can easily overwhelm the stream processing system. The current CASA system addressed both issues by means of taking average.

Figure 1 shows a simplified diagram of the current system. The top box in the figure depicts the generation of radial wind velocity from the raw data. The raw stream is partitioned into segments based on the frequency used. The velocity analysis transforms each segment into the frequency domain using Fast Fourier Transform (FFT) for signal processing. It then outputs a single velocity value for each segment and averages the values for high and low frequency segments close in time. The reflectivity analysis, in the lower box of the figure, generates reflectivity values also by averaging over adjacent stream segments. While such average operations can reduce data volume and gain a smoothing effect, the resulting data is still highly noisy, causing *low quality detection results* and *long running time*.

In our case study (detailed in §6.3), we explore the use of distributions, rather than simple average values, to separate useful data from noise while controlling the data volume. The output of our data analysis contains tuple streams with distributions, i.e., (`time, azimuth, distance`, `velocity`^{p}) and (`time, azimuth, distance`, `reflectivity`^{p}). We also study the transformation of these distributions through CASA operations, specially, the frequently used aggregation operations. By doing so, we expect to gain better tornado detection results yet with shorter execution time. Moreover, the distributions we provide also enable a potential extension of detection algorithms to use these distributions, and allow detailed post-facto analysis by the scientists.

3. THE CLARO DATA MODEL

The foundation of the CLARO system is a data model that can capture a variety of uncertainties for continuous attributes and further allow fast relational processing. Most notably, CLARO models continuous random attributes using *Gaussian Mixture Models* (GMMs) to leverage their flexibility and efficiency in computation. In this section, we introduce Gaussian Mixture Models, present techniques that transform real-world data into these models, and finally present the complete CLARO data model for relational processing.

3.1 Gaussian Mixture Models (GMMs)

Gaussian Mixture Models (GMMs) are statistical methods that are traditionally used for data clustering and density estimation. As an instance of probability mixture models, a GMM describes a probability distribution using a convex combination of Gaussian distributions.

Definition 1 *A Gaussian Mixture Model for a continuous random variable X is a mixture of m Gaussian variables $X_1, X_2, \cdots, X_m$. The probability density function (pdf) of X is:*

$$f_X(x) = \sum_{i=1}^{m} p_i f_{X_i}(x),$$

$$f_{X_i}(x) \sim N(u_i, \sigma_i^2) = \frac{1}{\sigma_i\sqrt{2\pi}} e^{-\frac{(x-u_i)^2}{2\sigma_i^2}},$$

where $0 < p_i < 1$, $\sum_{i=1}^{m} p_i = 1$.

It is apparent from this definition that the commonly used Gaussian distribution is a special case of GMM with the number of components equal to one.

Definition 2 *A multivariate Gaussian Mixture Model for a random vector $\mathbf{X}$ naturally follows from the definition of multivariate Gaussian distributions:*

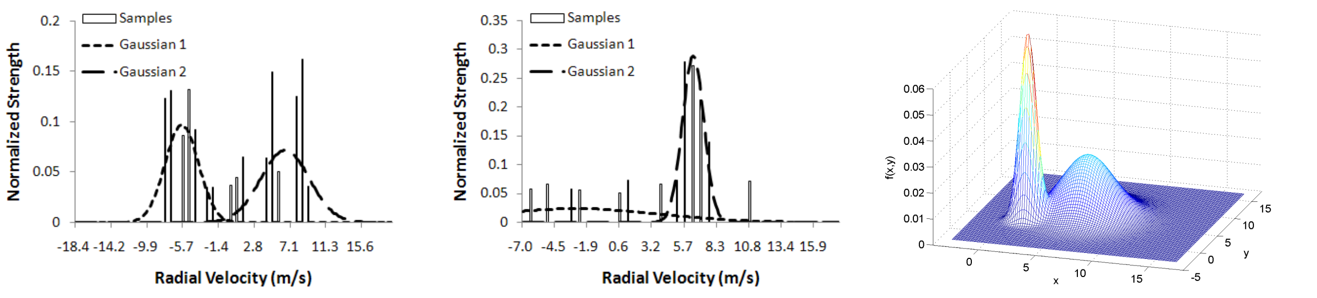
$$f_{\mathbf{X}}(\mathbf{x}) = \sum_{i=1}^{m} p_i f_{\mathbf{X}_i}(\mathbf{x}),$$

$$f_{\mathbf{X}_i}(\mathbf{x}) \sim N(\mathbf{u}_i, \Sigma_i) = \frac{1}{(2\pi)^{k/2}|\Sigma_i|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mathbf{u}_i)^T\Sigma_i^{-1}(\mathbf{x}-\mathbf{u}_i)},$$

where k is the size of random vector, and each mixture component is a k-variate Gaussian distribution with the mean $\mathbf{u}_i$ and the covariance matrix Σ_i.

The CLARO system adopts Gaussian Mixture Models due to several key benefits of these models. First, GMMs are a natural extension to Gaussian distributions that are widely used in scientific sensing and financial applications. Hence, they can be easily accepted by end users such as the CASA scientists we are working with.

Second, GMMs are highly flexible in modeling arbitrary real-world data distributions. Figure 3 shows several data distributions from our target applications. Figure 2(a) shows a bimodal distribution of velocity at the boundary between a positive velocity area and a negative velocity area in a tornadic event. In contrast, Figure 2(b) shows a velocity distribution in a positive velocity area, in which one Gaussian component captures the peak and the other captures the noise spread across the entire spectrum. In the RFID application, Figure 2(c) shows the location distribution of a recently moved object produced by our inference algorithm [30]. Here, the bivariate, bimodal GMM represents the possibilities of the old and new locations using two mixture components;



(a) Velocity distribution after FFT in Area 430 Azimuth 281.9 degree in a tornadic event (b) Velocity distribution after FFT in Area 430 Azimuth 282.3 degree in a tornadic event (c) Location distribution of a recently moved object detected using RFID readers

Figure 2: Gaussian Mixture Models for real-world data collected from our target applications

each component is a bivariate Gaussian modeling the joint distribution of x and y locations.

The third benefit of GMMs is efficient computation based on Gaussian properties and advanced statistical theory. For instance, the mean and variance of GMMs can be computed directly from those of the mixture components:

$$E[X] = \sum_{i=1}^{m} p_i E[X_i] \tag{1}$$

$$Var[X] = E[X^2] - (E[X])^2 = \sum_{i=1}^{m} p_i E[X_i^2] - (E[X])^2$$
$$= \sum_{i=1}^{m} p_i (Var[X_i] + (E[X_i])^2) - (E[X])^2 \tag{2}$$

Further, the cumulative distribution function (cdf) of a GMM with a single variable can be written out in closed form:

$$F_X(x) = \int_{-\infty}^{x} f_X(x)\,\mathrm{d}x = \sum_{i=1}^{m} \frac{a_i}{2}[1 + erf(\frac{x-u_i}{\sigma_i\sqrt{2}})] \tag{3}$$

where $erf()$ is the known error function. Values of the error function can be approximated very accurately using numerical methods. In fact, these values are precomputed and stored in any numerical library. Hence, computing $F_X(x=a)$ or $\int_a^b f_X(x)\mathbf{d}x$ = $F_X(a)$-$F_X(b)$ incurs little cost.

Other computation benefits of GMMs, such as the characteristic functions, product distributions, and marginalization, are described in later sections when they are used.

3.2 Generating GMMs from Real-World Data

Gaussian Mixture Models can be generated from real-world data in a variety of ways.

Samples. Several recent studies [19, 30] have developed inference techniques that transform raw data to distributions and represent those distributions using weighted samples. Given these samples, GMMs can be generated using existing tools for density estimation or function fitting. If an application models data using other distributions, e.g., the Gamma distribution, it is easy to generate samples from this distribution and then fit a GMM as described above.

Correlated time series. Time series data is prevalent in many applications. Values in a time series are temporally correlated so cannot be viewed as samples to fit GMMs. Two techniques can be applied in this case:

Fast Fourier Transform (FFT) translates a correlated data sequence in the time domain to an uncorrelated sequence in the frequency domain. The latter is essentially a discrete distribution that can be used to fit a GMM. Although FFT has the $O(n \log n)$ complexity, where n is the sequence length, doing so for short subsequences of data does not incur high overhead. In fact, the CASA system has already applied FFT to the streams arriving at 175 Mb/sec. We will show the use of this technique in our case study in Section 6.3.

Another method is to use the autoregressive moving average (ARMA) model which restricts data correlations to the past n time steps. Our previous study applied ARMA fitting to the radar data streams with and without a tornado [10]. We discovered that in either case, the ARMA model with n=5 satisfies the statistical condition for fitting. Given such models, we can perform sampling at the frequency of once every n+1 values and feed the samples to fit GMMs.

The CLARO system offers all above methods for transforming raw data to tuples modeled using GMMs. Since the data provider has the best knowledge about the suitable method, in this work we assume that such transformation is performed on the stream provider side and the input to CLARO has already had uncertain data modeled using GMMs.

3.3 CLARO Data Model

We now present the complete data model that CLARO uses for relational processing. An uncertain data stream is an infinite sequence of tuples that conform to the schema $\mathcal{A}^d \cup \mathcal{A}^p$. The attributes in $\mathcal{A}^d$ are *deterministic attributes*, such as the tuple id and the fixed x-y location of a sensor. The attributes in $\mathcal{A}^d$ are real-valued *probabilistic attributes*, such as the temperature of a location and the velocity and reflectivity in an area. The set of probabilistic attributes is further partitioned into independent attributes $A_i^p \in \mathcal{A}^p$ and groups of correlated attributes $\mathbf{A}_j^p \subseteq \mathcal{A}^p$. An independent probabilistic attribute is modeled using a *continuous random variable* following a Gaussian Mixture distribution, denoted by $f(A_i^p)$. A set of correlated attributes $\mathbf{A}_j^p$ is modeled using a set of continuous random variables following a multivariate Gaussian Mixture distribution, denoted by $f(\mathbf{A}_j^p)$.

In CLARO, the *tuple distribution* is defined as:

$$f(\mathcal{A}^p) = \prod_i f(A_i^p) \prod_j f(\mathbf{A}_j^p),$$

which is a multivariate Gaussian Mixture distribution. Then, the *tuple existence probability* (P_t) is defined to be the integral of the tuple distribution over all attributes in $\mathcal{A}^p$.

$$P_t = \int \cdots \int_{\mathcal{A}^p} f([x_1 \cdots x_k] \in \mathcal{A}^p)\, \mathrm{d}x_1 \cdots \mathrm{d}x_k$$

Under normal conditions, the P_t of tuples is 1. In particular, all input tuples to CLARO are assumed to have P_t=1 (which is true in both of our target applications). However, the P_t of tuples can become less than 1 due to query processing. In this paper, we first assume the P_t of all tuples to be 1 to focus on the main techniques. We then relax this assumption in Section 7 and describe an extension of our techniques.

In general tuples in a stream can be correlated. Inter-tuple correlations can be modeled using a joint tuple distribution or lineage [2]. Our current data model does not include such correlations due to two reasons: First, while raw data is often correlated, our methods for transforming raw data to tuples, such as FFT and sampling based on the ARMA model, have already taken temporal correlations into account. Second, performance is the key to stream processing. For this reason, stream systems may sometimes have to sacrifice inter-tuple correlations to meet stringent performance requirements. For instance, the CASA system ignores spatial correlations in any processing prior to final tornado detection, and existing work on probabilistic stream processing [19, 24] ignores inter-tuple correlations, for the performance reason. A thorough treatment of tuple correlations in stream processing is left to our future work.

In the rest of the paper, we present evaluation strategies for aggregates and joins under the CLARO data model. An extension of our work to a more general data model and a larger set of relational operators is delayed until Section 7.

4. AGGREGATION OF UNCERTAIN TUPLES

We first address aggregation of uncertain tuples. In this work, we focus on aggregation using `sum` and `avg` because they are crucial to our target applications but have not been sufficiently addressed in prior work.[1] Much of existing work on probabilistic databases [1, 6] focuses on discrete probabilistic attributes and employs possible world semantics for query processing. For continuous attributes, there have been two main approaches to supporting aggregates:

The *integral-based approach* directly manipulates continuous random variables and derives exact result distributions. Consider the sum of n tuples modeled by n continuous random variables. A state-of-the-art solution [4] integrates two variables at a time, resulting in the use of n-1 integrals for aggregating these tuples. As we shall show in Section 4.1, even if we use a hand-optimized numerical method for integration, computing a single integral for each sum operation is still too slow for data streams. This implies that the approach using n-1 integrals for each sum operation is infeasible for stream processing.

The *sampling-based approach* [14, 28] generates a fixed number of samples from the distribution of each input tuple, computes a set of answers using the samples, and constructs the result distribution from these answers. Despite its generality, its approach has two main drawbacks: First, it is unknown how many samples are needed a priori. When a small number of samples is used, the computation is fast but with potential loss of accuracy. To accurately characterize complex distributions, the number of samples needed can be large, hence penalizing performance. Further, the sampling-based approach has no means to acquire the knowledge about the true result distribution. Hence, it can not adapt to varying data characteristics and accuracy requirements.

Our work departs from existing approaches by exploring advanced statistical theory to obtain *exact result distributions* while completely eliminating the use of integrals. However, the formulas for result distributions that the exact algorithm produces grow exponentially in the number of aggregated tuples. Hence, we provide two *approximation* techniques to simplify the formulas for result distributions. These techniques can satisfy varying accuracy requirements while achieving high speed in stream processing. It is important to note that such accuracy cannot be guaranteed without the knowledge of the true result distribution!

In the following, we present our techniques for evaluating `sum`(A) and `avg`(A) over a tuple stream, where A is a continuous random attribute. For succinctness, our discussion focuses on `sum`. The application to `avg` is straightforward.

4.1 Using Characteristic Functions

In this section, we introduce *characteristic functions* and describe how they can be used to derive the result distribution for `sum` of a set of tuples.

In probability theory, characteristic functions, as their name suggests, are used to "characterize" distributions. Specifically, the charactertistic function of a random variable U completely defines its distribution as follows (chapter 2, [12]).

$$\Phi_U(t) = Ee^{iUt},$$

where E denotes the expected value and i is the complex number $\sqrt{-1}$. The pdf of U can be obtained by the inverse transformation of the characteristic function:

$$f_U(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} e^{-itx} \Phi_U(t) dt$$

Now let us consider `sum`(A), with the attribute A in n tuples being modeled using X_1, ..., X_n . Let $U = X_1 + X_2 + ... + X_n$. The characteristic function of U is:

$$\begin{aligned} \Phi_U(t) &= Ee^{iUt} = Ee^{i(X_1+X_2+...+X_n)t} \\ &= Ee^{iX_1t}e^{iX_2t}...e^{iX_nt} \\ &= \Phi_{X_1(t)}\Phi_{X_2(t)}...\Phi_{X_n(t)} \end{aligned}$$

The step 2 above holds due to the independence assumption among the X_i's. For a GMM, its characteristic function can be expressed in closed form (i.e., without the integral to compute the expectation). For example, for a Gaussian mixture of two components:

$$f(x) = p_1 \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} + p_2 \frac{1}{\sigma_2\sqrt{2\pi}} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}},$$

its characteristic function is:

$$\Phi_X(t) = p_1 e^{i\mu_1 t - \frac{1}{2}\sigma_1^2 t^2} + p_2 e^{i\mu_2 t - \frac{1}{2}\sigma_2^2 t^2}$$

Therefore, the characteristic function for `sum`(A) can be directly written out as the product of characteristic functions of the input tuples. The above discussion leads to an algorithm for `sum` with three steps.

[1]Our system also supports `min` and `max`. The techniques we develop are similar to those in [4] and hence omitted in this paper.

Algorithm 1 Direct approach to `sum`

```
Input: Independent tuples, $X_1$, ..., $X_n$; a given value, $x$.
Output: $f_U(x)$, with $U = X_1 + ... + X_n$
1: Get the characteristic function of each tuple $X_i$.
2: Take the product of these characteristic functions.
3: Apply the inverse transformation at $x$.
```

Given our choice of GMMs, step 1 does not involve any integration. The computation thus involves a *single integral* for the inverse transformation. This gives a boost in performance compared to the two-variable convolution method, which requires $n-1$ integrals [4].

The main limitation of this approach is that the result distribution is **symbolic** because it involves an unresolved integral in the pdf formula. To get sufficient knowledge of the result distribution (e.g., calculating its mean and variance), one needs to repeat the inverse transformation for a large number of points. To understand the cost of such repeated integration in our initial study, we used a **numerical solution** called adaptive quadrature [26] to compute integrals. The task is to average over 10 tuples and compute the pdf values for 20 points. We applied optimizations such as restricting the range of integration and limiting the maximum number of iterations. The throughput obtained is less than 200 tuples/second. This indicates that this technique is inefficient for our data stream applications. Besides, it is unknown if the result distribution is a GMM.

4.2 Exact Derivation

The discussion in the previous section motivated us to seek a solution without using numerical integration. For GMMs, it turns out that we can obtain the **closed-form** solution to the inverse transformation. In addition, when input tuples are Gaussian mixtures and independent, the result of `sum` over those tuples is also a Gaussian mixture that can be directly obtained from the input tuples.

Theorem 4.1 *Let each $X_i, (i = 1..n)$ be a mixture of i_m components identified by the parameters $(p_{ij}, \mu_{ij}, \sigma_{ij}), (j = 1..i_m)$. The result distribution for $U = \sum_{i=1}^{n} X_i$ is a Gaussian mixture of $\prod_{i=1}^{n} i_m$ components, each of which corresponds to a unique combination that takes one component from each input Gaussian mixture $\{i_j\}, (i = 1..n, j \in \{1..i_m\})$ and is identified by (p_k, μ_k, σ_k):*

$$p_k = \prod_{i=1}^{n} p_{i_j};\ \mu_k = \sum_{i=1}^{n} \mu_{i_j};\ \sigma_k = \sqrt{\sum_{i=1}^{n} \sigma_{i_j}^2}.$$

Proof: See Appendix.

To the best of our knowledge, we are not aware of any state-of-the-art book on mixture models showing this result [13, 11, 23] .

This technique gives an exact solution so the accuracy is guaranteed. The only computation involved is to enumerate and compute all components of the result Gaussian mixture. The number of components in the result distribution, however, is exponential in number of input tuples. Thus, this technique cannot scale to a large number of tuples. Our goal is to settle for simpler distributions that are still good approximations of the true ones. We next describe our approximation techniques to achieve this goal.

4.3 Approximation of Closed-Form Solution

In this section, we seek an approximation technique to simplify the formula of the result distribution while achieving a bounded error. The nature of this task is similar to approximate lineage [25], in which the complex lineage formula is replaced with an approximate and simpler one. The approximate distributions in our work are user-friendly and allow lower cost in subsequent processing.

For the cases that the result Gaussian mixtures have too many components, we propose to simplify them by reducing the number of components while achieving the accuracy requirements. The idea is to group adjacent Gaussian peaks together and approximate them with a single component. We search for the right number of components, K, by starting with a small number and increasing it if the accuracy is not satisfied. Since we know the property of the result distribution, we can quantify our approximation on the fly.

The algorithm, referred thereafter as **sort-group**, has the following main steps:

Algorithm 2 Approximation with **sort-group**

```
1: Start with the number of components $K = 1$.
2: Enumerate all $N$ components of the result distribution
   based on Theorem 4.1. Each component is a Gaussian
   $N(\mu_i, \sigma_i)$ with a coefficient $p_i$.
3: Sort the components in increasing order of $\mu_i$.
4: Group the components into $K$ result Gaussian com-
   ponents. Each of them replaces $\lfloor \frac{N}{K} \rfloor$ original compo-
   nents; except for the last one which replaces the last
   $(N - (K-1)\lfloor \frac{N}{K} \rfloor)$ original components.
5: Calculate the approximate point-based variation distance
   (VD) to see if it is satisfied the given accuracy constraint.
   If so, return the mixture of $K$ Gaussian components; oth-
   erwise, increase $K$ and go to step 4.
```

In step 4, we group each m components and replace them with a single Gaussian. Formally, this can be written as: $p_1 N(\mu_1, \sigma_1) + \cdots + p_m N(\mu_m, \sigma_m) =$
$\sum_{i=1}^{m} p_i \left[\frac{p_1}{\sum_{i=1}^{m} p_i} N(\mu_1, \sigma_1) + \cdots + \frac{p_m}{\sum_{i=1}^{m} p_i} N(\mu_m, \sigma_m) \right]$.
The quantity inside the bracket represents a Gaussian mixture and is approximated by a single Gaussian with mean and variance equal to those of the Gaussian mixture.

In step 5, the point-based VD is a metric we use to measure the distance between distributions. For two continuous distributions $D_1(x)$ and $D_2(x)$, it is defined as:

$$VD = \frac{1}{2} \sum_x |D_1(x) - D_2(x)| \Delta x \tag{4}$$

The values of x are evenly spaced and taken from the range where the two distributions have most of their density mass lies. Δx is the distance between two consecutive x points. The constant $\frac{1}{2}$ ensures that VD is in [0,1]. This idea is similar to the metric used in [14] in the sense of discretizing continuous distributions except that we use *equi-width* histograms, instead of *equi-depth* ones.

Instead of using a large number of points (i.e., 1000) to calculate VD as used for accuracy measurement, we use a smaller number to reduce this computation overhead. It is experimentally shown that we can achieve a reasonable approximation of the true VD using 30 points. Also, as we know the true distribution, the points for the true distribution can be calculated once and stored to be used later.

The search for the number of components K can be optimized by using information from previous segments of the

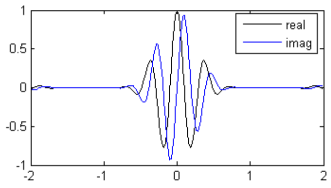


Figure 3: Example of Characteristic Function for avg of 10 tuples.

stream to determine the initial K. A simple heuristic is to use the average of previous K's. This is applied after the "warm-up" phase (i.e., when the system is stabilized) so that we have a sufficient number of segments to take average from.

4.4 Approximation using Fitting

The previous approximation still enumerates all mixture components of the result formula and becomes very inefficient when the number of input tuples is large. We next propose to approximate the result distributions by performing function fitting in the characteristic function space. This is based on the property that the characteristic function of sum can be succinctly represented as a product of n individual characteristic functions (instead of an exponential number of components). The goal is to find some Gaussian mixture whose characteristic function best fits this product function.

We devise an approximation algorithm, named **Characteristic Function (CF) fitting**. It works as follows:

Algorithm 3 Approximation with **CF fitting**

```
1: Obtain the expression of the CF of the sum, $\Phi_{\texttt{sum}}(t) = \prod_{i=1}^{n} \Phi_{X_i}(t)$. This is a complex function.
2: Take $M$ points $\{t_i\}, (i = 1..M)$ from a range $[C_1, C_2]$, , and compute $\{\Phi_{\texttt{sum}}(t_i)\}, (i = 1..M)$.
3: Start with $K = 1$. Consider a Gaussian mixture of $K$ components. The corresponding CF is $\overline{\Phi}(t)$.
4: Evaluate the function using the same $M$ points to get $\{\overline{\Phi}(t_i)\}(i = 1..M)$.
5: Run least squares fitting to minimize:
   $\sum_{i=1}^{M} \left[(Re(\overline{\Phi}(t_i) - \Phi_{\texttt{sum}}(t_i)))^2 + (Im(\overline{\Phi}(t_i) - \Phi_{\texttt{sum}}(t_i)))^2\right]$.
6: Get the fitting residual. If this is smaller than a threshold $\epsilon$, return the fitted Gaussian mixture. Otherwise, increase $K$ and go back to step 3.
```

Note that the objective function for fitting contains both *real* and *imaginary* parts since the characteristic functions are complex and both parts contribute to the pdf via inverse transformation.

We further propose several optimizations based on statistical theory to improve performance and accuracy.

Range of CFs: Since the characteristic function $\Phi(t)$ approaches 0 fast as the magnitude of t increases, the range $[C_1, C_2]$ needs to be small and centered around 0. See Figure 3 for an example of CF for avg of 10 tuples. Both real and imaginary parts of the function are shown. Hence, we use $[-1, 1]$ in our algorithm as the range can capture the shape of the CFs in most cases.

Inital guesses: Due to the oscillating behavior of the characteristic functions, the fitting results are quite sensitive to the initial values and can get stuck in local optima. Finding good initial guesses for fitting can be as hard as fitting itself. Theorem 4.1 provides insights into choosing these initial guesses. Specifically, we precompute a fixed number of result components (e.g., 40) and use them to derive the guesses. These components are chosen so that their means are separated to capture different regions of the result distribution. At each iteration, they are grouped into K components that are used as initial guesses.

Choice of fitting residual: The fitting residual ϵ is chosen to guarantee the given VD requirement. We have performed an approximation analysis and derived an upper bound for ϵ. The proof is based on the application of Cauchy-Schwartz inequality to characteristic functions and the property that these functions approach 0 quickly for points far from the origin to restrict the integration bounds. Given our choice of other paramaters, we can show that $\epsilon \leq 6 * 10^{-5} VD^2$ can meet the accuracy requirement VD.

Relation to the Central Limit Theorem. The Central Limit Theorem (CLT) is a special case of our algorithm. It states that the sum of a sufficiently large number of independent random variables is normally distributed [12]. This gives an asymptotic result but our algorithm dynamically determines when this result applies with a bounded small error. Our experiments show that the fitted result distributions are smooth single Gaussians when the number of tuples is large enough (e.g., ≥ 20). However, when this number is smaller, our algorithm will fit using a Gaussian mixture to better capture more complex distributions.

The above aggregation techniques can be directly applied to stream systems using tumbling windows such as the CASA system and XStream system [15]. When sliding windows are used instead, they will repeat computation for each window.

5. JOINS OF UNCERTAIN TUPLES

In this section we consider efficient evaluation of joins under the CLARO data model. Consider an example $R \bowtie_{R.a=S.a} S$. The evaluation strategies of the join vary significantly based on the nature of the join attributes. Recent research on probabilistic databases [1, 2, 20, 31] has mostly focused on the case that at least one of the join attributes is probabilistic and the uncertainty is captured using a *discrete* random variable. Since it is possible to enumerate the values of a discrete random variable, existing work supports such joins based on the possible worlds semantics (PWS): in each possible world, each random variable takes a specific value so a join can proceed just as in a traditional database.

In a more complex scenario, on which our work focuses, at least one of the join attributes is probabilistic and the uncertainty is captured using a *continuous* random variable. The approach based on PWS does not apply because we can no longer enumerate the possible values of a continuous random variable. The recent work [28] defines the result tuple distributions of a join to be the result tuple distributions of a selection following a cross-product. This definition, however, is useful only in limited situations and does not consider the performance issue related to the cross product.

Below, we consider joins of continuous random attributes based on two different types of semantics and propose evaluation and optimization techniques for these joins.

5.1 Joins using Probabilistic Views

We first consider a join between a continuous random attribute and a deterministic attribute. Query Q1 in Section 2.1 is such an example: the join attributes x and y in the object location stream are probabilistic, but x and y in the temperature stream are simply fixed sensor locations. This type of join is inherently difficult to support because it is not possible to enumerate the values of the continuous random attributes x^p and y^p. Furthermore, each value of x^p and y^p has probability 0. Hence, the join results based on the individual values of x^p and y^p also have probability 0.

In order to support this type of join, we introduce the notion of a **probabilistic view.** In the above example, a probabilistic view on the temperature stream is defined to be the conditional probability $p(temp|x,y)$, that is, the distribution of temperature given a particular (x,y) value. Once this view is defined, for any possible value of x^p and y^p, we can retrieve the temperature distribution based on this value. The process of iterating all possible values of x^p and y^p and retrieving the corresponding temperature distributions can be compactly represented by $f(x^p, y^p)p(temp|x,y)$, yielding a joint distribution $f(x^p, y^p, temp^p)$.

Formally, let the left join input R be $\{A_1^p, \cdots, A_k^p, \bar{R}\}$, where $A_1^p, \cdots, A_k^p$ are the join attributes and $\bar{R}$ denotes the rest of R. Let the right input S be $\{A_1, \cdots, A_k, B_1, \cdots, B_l, \bar{S}\}$, where $A_1, \cdots, A_k$ are the join attributes, $B_1, \cdots, B_l$ are the attributes that can be functionally determined by the join attributes in a deterministic world, and $\bar{S}$ is for the rest of S. In our example of Q1, the functional dependency in the temperature stream is (x, y) $\rightarrow$ temp. Then a probabilistic view can be defined for the attributes $B_1, \cdots, B_l$ conditioned on the attributes $A_1, \cdots, A_k$, denoted by $\mathbf{V}_{B_1^p,\cdots,B_l^p|A_1,\cdots,A_k}$, which is essentially characterized by $p(B_1, \cdots, B_l|A_1, \cdots, A_k)$. We now give the definition of the join between R and S based on the probabilistic view semantics.

Definition 3 *Given $R = \{A_1^p, \cdots, A_k^p, \bar{R}\}$, $S = \{A_1, \cdots, A_k, B_1, \cdots, B_l, \bar{S}\}$ with FD: $A_1, \cdots, A_k \rightarrow B_1, \cdots, B_l$, a join of R and S based on the probabilistic view semantics ($\bowtie^{\mathbf{V}}$) is a left outer join of R and the probabilistic view $\mathbf{V}_{B_1^p,\cdots,B_l^p|A_1,\cdots,A_k}(S)$.*

$$R \underset{R.A_1^p=S.A_1,\cdots,R.A_k^p=S.A_k}{\bowtie^{\mathbf{V}}} S \equiv R \underset{R.A_1^p,\cdots,R.A_k^p}{⟕} \mathbf{V}_{B_1^p,\cdots,B_l^p|A_1,\cdots,A_k}(S)$$

In this definition, the left outer join preserves each tuple in R and extends it with the attributes $B_1^p, \cdots, B_l^p$ from S. The attributes in S that are neither the join attributes nor functionally determined by the join attributes are not included in the output as the probabilistic view cannot capture them.

Closed-form result distributions. Given the join semantics based on the probabilistic view, we seek a solution to the tuple distribution of each outer join result. Recall that the Claro data model describes the join attributes in the left input using Gaussian Mixture Models (GMMs) or a multivariate GMM if they are correlated. Next, we propose a special model for the probabilistic view, which we call *order-1 linear regression*, that allows us to obtain closed-form join result distributions also in the form of GMM. While the assumption of order-1 linearity may sound restrictive, it can be actually applied to the view at either a global or local scale, hence amenable to accurate and efficient implementation.

The following two theorems directly offer closed-form join result distributions. They differ based on the nature of the attributes used to the construct the probabilistic view.

Theorem 5.1 *Given $R = \{A_1^p, \cdots, A_k^p, \bar{R}\}$, $S = \{A_1, \cdots, A_k, B_1, \cdots, B_l, \bar{S}\}$, assume the following order-1 linear regression:*

$$\tilde{B} = \tilde{A}\boldsymbol{\beta} + \mathbf{E} \tag{5}$$

where $\tilde{B} = (B_1, \cdots, B_l)$, $\tilde{A} = (A_1, \cdots, A_k)$, $\boldsymbol{\beta}$ is a parameter matrix of $k \times l$, and $\mathbf{E}$ is an error vector of length l, which is normally distribution with mean $\mathbf{0}$ and covariance matrix Σ_E. If $\tilde{A}$ follows a GMM, then the outer join result distribution $(\tilde{A}, \tilde{B})$ follows a multivariate GMM.

Proof sketch: According to (5), we have

$$\tilde{B}|\tilde{A} \sim N(\tilde{A}\boldsymbol{\beta}, \Sigma_E). \tag{6}$$

Given $\tilde{A} \sim$ GMM, there exists a random variable C that captures the mixing distribution of $\tilde{A}$ such that

$$C \sim Multinomial(m, p_1, \cdots, p_m) \tag{7}$$

$$\tilde{A}|C = c \sim N(\boldsymbol{\mu}_{\tilde{A},c}, \Sigma_{\tilde{A},c}) \tag{8}$$

Based on (6) and (8), matrix manipulation can show that

$$(\tilde{A}, \tilde{B})|C = c \sim N\left((\boldsymbol{\mu}_{\tilde{A},c}, \boldsymbol{\mu}_{\tilde{A},c}\boldsymbol{\beta}), \begin{pmatrix} \Sigma_{\tilde{A},c} & \Sigma_{\tilde{A},c}\boldsymbol{\beta} \\ \boldsymbol{\beta}^T\Sigma_{\tilde{A},c} & \Sigma_E + \boldsymbol{\beta}^T\Sigma_{\tilde{A},c}\boldsymbol{\beta} \end{pmatrix}\right).$$

Together with (7), we have $(\tilde{A}, \tilde{B})$ follows GMM. $\square$

The above proof sketch fully characterizes the join result distribution, including all its parameters. In practice, $\boldsymbol{\beta}$ and Σ_E are unknown. They can be estimated using regression over the tuples in the S input, denoted by S_i=$\{\tilde{A}_i, \tilde{B}_i\}$, $i = 1, \cdots, n$. The least square estimates of these parameters are:

$$\boldsymbol{\beta} = (\tilde{\mathbf{A}}^T\tilde{\mathbf{A}})^{-1}\tilde{\mathbf{A}}^T\tilde{\mathbf{B}} \tag{9}$$

$$\Sigma_E = \tilde{\mathbf{B}}^T(\mathbf{I}_n - \tilde{\mathbf{A}}(\tilde{\mathbf{A}}^T\tilde{\mathbf{A}})^{-1}\tilde{\mathbf{A}}^T)\tilde{\mathbf{B}}/(n-k), \tag{10}$$

where $\tilde{\mathbf{A}} = (\tilde{A}_1^T, \cdots, \tilde{A}_n^T)^T$ and $\tilde{\mathbf{B}} = (\tilde{B}_1^T, \cdots, \tilde{B}_n^T)^T$.

Theorem 5.2 *Given $R = \{A_1^p, \cdots, A_k^p, \bar{R}\}$, $S = \{A_1, \cdots, A_k, B_1^p, \cdots, B_l^p, \bar{S}\}$, let $\tilde{B}^p = (B_1^p, \cdots, B_l^p)$ and assume it follows a GMM with r components, each with mean $\boldsymbol{\mu}_{\tilde{B},j}$ and covariance $\Sigma_{\tilde{B},j}$. Further assume the order-1 linear regression:*

$$\boldsymbol{\mu}_{\tilde{B},j} = \tilde{A}\boldsymbol{\beta}_j^p + \mathbf{E}_j^p, \quad j = 1, \cdots, r \tag{11}$$

where $\boldsymbol{\beta}_j^p$ and $\mathbf{E}_j^p$ are similarly defined as $\boldsymbol{\beta}$ and $\mathbf{E}$, and we denote the covariance matrix of $\mathbf{E}_j^p$ as $\Sigma_{E,j}^p$. Then if $\tilde{A}^p$ follows a GMM, we conclude that $(\tilde{A}^p, \tilde{B}^p)$ follows a multivariate GMM as well.

Proof sketch: Given $\tilde{B}^p \sim GMM$, there exists a random variable $C_{\tilde{B}}$ that capures the mixing distribution of $\tilde{B}^p$ such that

$$C_{\tilde{B}} \sim Multinomial(r, q_1, \cdots, q_r) \tag{12}$$

$$\tilde{B}^p|C_{\tilde{B}} = j, \boldsymbol{\mu}_{\tilde{B},j} \sim N(\boldsymbol{\mu}_{\tilde{B},j}, \Sigma_{\tilde{B},j}) \tag{13}$$

According to assumption (11), we have

$$\boldsymbol{\mu}_{\tilde{B},j}|\tilde{A} \sim N(\tilde{A}\boldsymbol{\beta}_j^p, \Sigma_{E,j}^p) \tag{14}$$

Combining (13) and (14), we obtain

$$\tilde{B}^p|C_{\tilde{B}} = j, \tilde{A} \sim N(\tilde{A}\boldsymbol{\beta}_j^p, \Sigma_{E,j}^p + \Sigma_{\tilde{B},j}). \tag{15}$$

Given $\tilde{A}^p \sim$ GMM, there exists a random variable $C_{\tilde{A}}$ that captures the mixing distribution of $\tilde{A}^p$ such that

$$C_{\tilde{A}} \sim Multinomial(m, p_1, \cdots, p_m) \tag{16}$$

$$\tilde{A}^p|C_{\tilde{A}} = i \sim N(\boldsymbol{\mu}_{\tilde{A},i}, \Sigma_{\tilde{A},i}) \tag{17}$$

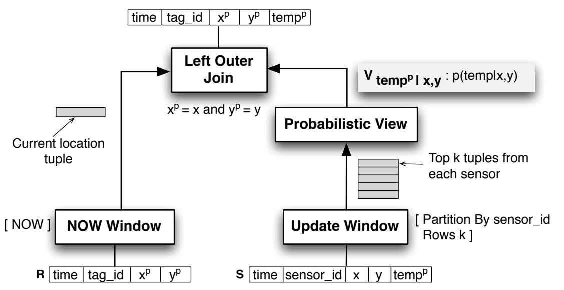

time | tag_id | x^p | y^p | temp^p
Left Outer Join
x^p = x and y^p = y
V temp^p | x,y : p(temp|x,y)
Probabilistic View
Current location tuple
Top k tuples from each sensor
[NOW]
NOW Window
Update Window
[Partition By sensor_id Rows k]
R | time | tag_id | x^p | y^p
S | time | sensor_id | x | y | temp^p

Figure 4: Query plan for the join using a probabilistic view (Q1).

Combining (15) and (17), matrix manipulation can show that

$$(\tilde{A}^p, \tilde{B}^p)|C_{\tilde{A}} = i, C_{\tilde{B}} = j$$
$$\sim N\left((\boldsymbol{\mu}_{\tilde{A},i}, \boldsymbol{\mu}_{\tilde{A},i}\boldsymbol{\beta}_j^p), \begin{pmatrix} \Sigma_{\tilde{A},i} & \Sigma_{\tilde{A},i}\boldsymbol{\beta}_j^p \\ \boldsymbol{\beta}_j^{pT}\Sigma_{\tilde{A},i} & \Sigma_{E,j}^p + \Sigma_{\tilde{B},j} + \boldsymbol{\beta}_j^{pT}\Sigma_{\tilde{A},i}\boldsymbol{\beta}_j^p \end{pmatrix}\right).$$

Define $C = (C_{\tilde{A}}, C_{\tilde{B}})$, then based on (12) and (16), we have

$$C \sim Multinomial(mr, p_i q_j, i = 1, \cdots, m, \;\; j = 1, \cdots, r).$$

The above two equations show that $(\tilde{A}^p, \tilde{B}^p)$ follows a GMM. □

In practice, $\boldsymbol{\beta}_j^p$ and $\Sigma_{E,j}^p$ are unknown. They can be estimated using regression over the tuples in the S input, denoted by S_i=$\{\tilde{A}_i, \tilde{B}_i^p\}$, $i = 1, \cdots, n$. Also, the distribution of $\tilde{B}_i^p$ is given, which is a GMM with mean $\boldsymbol{\mu}_{\tilde{B},j}$ and covariance $\Sigma_{\tilde{B},j}$, $j = 1, \cdots, r$. The least square estimates of these parameters are:

$$\boldsymbol{\beta}_j^p = (\tilde{\mathbf{A}}^T\tilde{\mathbf{A}})^{-1}\tilde{\mathbf{A}}^T\tilde{\boldsymbol{\mu}} \qquad (18)$$

$$\Sigma_{E,j}^p = \tilde{\boldsymbol{\mu}}^T(\mathbf{I}_n - \tilde{\mathbf{A}}(\tilde{\mathbf{A}}^T\tilde{\mathbf{A}})^{-1}\tilde{\mathbf{A}}^T)\tilde{\boldsymbol{\mu}}/(n-k), \qquad (19)$$

where $\tilde{\mathbf{A}} = (\tilde{A}_1^T, \cdots, \tilde{A}_n^T)^T$ and $\tilde{\boldsymbol{\mu}} = (\boldsymbol{\mu}_{\tilde{B},1}^T, \cdots, \boldsymbol{\mu}_{\tilde{B}_n}^T)^T$.

Evaluation Techniques. The query plan for a join using a probabilistic view is illustrated in Figure 4 for the example query Q1. For stream processing, this plan first applies the windows specified in the query to the two inputs: A Now window is applied to the location stream, which feeds each arriving location tuple as the left input to the left outer join. An update window, [Partition By sensor_id Rows k], selects the most recent k temperature tuples from each sensor into the window. The probabilistic view is maintained over the update window and returns the view $\mathbf{V}_{temp^p|x,y}$ as the right input to the left outer join. Finally, the join extends each location tuple from the left with the attributes presented by the view, and returns a joint distribution of these attributes.

Given the closed form join result distribution, the main implementation issue is the construction of the probabilistic view using regression. While recent research has applied regression to build models and views over sensor data [16, 9], our work differs by exploring the tradeoffs of applying order-1 linear regression at a global versus local scale.

Global regression: Take all S tuples in the current update window. Global regression is straightforward: simply apply regression equations (e.g., Eq. 9 and Eq. 10) to these tuples to construct the view defined by the regression (e.g., Eq. 6). As new S tuples arrive, some old tuples in the update window are replaced by the new tuples. The view, however, can be maintained incrementally by updating intermediate matrix operation results for $\boldsymbol{\beta}$, e.g., $\tilde{\mathbf{A}}^T\tilde{\mathbf{A}}$ and $\tilde{\mathbf{A}}^T\tilde{\mathbf{B}}$ in Eq. 9. In our work, we directly apply the incremental technique proposed in [16]. Then, when an R tuple arrives, the view is refreshed by completing the matrix operations for $\boldsymbol{\beta}$ and those for Σ_E.

A fundamental limitation of global regression is that the order-1 linear assumption may not hold over the entire view. For our example query Q1, the temperature may not be a linear function of the location but, for instance, a quadratic function instead. Hence, global regression may result in severe error when its assumption fails to match the reality.

Local regression: Our second technique is motivated by recent statistical theory that a smooth function can be well approximated by a low degree polynomial, often a linear function, in the neighborhood of any point. Hence, we design a local regression technique as follows: Given each R tuple that follows a GMM, use the mean and the variance of the GMM to dynamically define a *local regression region* (LR region). For example, the LR region for an R tuple with the distribution $N(\mu, \sigma)$ can be $[\mu + m\sigma, \; \mu - m\sigma]$ with $m \geq 2$. Then, retrieve the subset of S tuples that reside in the LR region and apply regression to these tuples.

A key advantage of this technique is that it does not require the assumption of global linearity, hence allowing more accurate view construction. However, when given a very small set of tuples in the LR region, it may not have enough data points to achieve the accuracy (which is a data problem, not a model problem). When this problem occurs, we can collect more data points by adjusting the LR region appropriately, and if needed, also enlarging the update window, that is, secretly keeping more old tuples for the purpose of regression. Computation-wise, regression is applied to a small set of tuples, hence with a low cost. In addition, we can cache the regression results of two LR regions that significantly overlap, similar in spirit to the materialized views.

5.2 Joins using the Cross-Product

Depending on the application, joins of continuous random attributes can also be modeled using the cross-product semantics. Query Q2 in Section 2.1 is such an example: it compares every pair of objects for their proximity in location. Our second join technique supports such joins with both result distributions and an efficient method to prune intermediate results of the cross-product.

Before defining join result distributions, we note that if a join between two continuous random attributes uses equality predicates, the probability of each join result is always 0. The solution is to transform the equality comparison into inequality comparison using a small bound δ. Then, we define the join result distributions as follows:

Definition 4 *Given* R = $\{A_1^p, \cdots, A_k^p, \bar{R}\}$, S = $\{A_1^p, \cdots, A_k^p, \bar{S}\}$, *let* $\tilde{A} = (A_1^p, \cdots, A_k^p)$. *The join* $R \bowtie_\theta S$, *where* θ= $|R.A_i^p - S.A_i^p| < \delta$ *(i=1,* $\cdots$*, k), produces tuples that each has a joint probability* $f(R.\tilde{A}, S.\tilde{A}) = f_{R.\tilde{A}}(R.\tilde{A})f_{S.\tilde{A}}(S.\tilde{A})$, *and existence probability* $\int_{|R.A_i^p - S.A_i^p| < \delta, i=1,\cdots,k} f(R.\tilde{A}, S.\tilde{A})\, \mathrm{d}R.\tilde{A}\, \mathrm{d}S.\tilde{A}$.

A similar definition can be given to a join between continuous random attributes and deterministic attributes based on the cross-product semantics.

While our definition for joins based on the cross-product is the same as in [28], our work further considers its efficient implementation. A cross-product can produce a large number of tuples, so we must filter most of them early using the join condition, more precisely, to filter those tuples with low existence probability. However, the evaluation of the join condition requires an integration, which can be an expensive operation. In our work, we devise *linear transformation* for GMMs to reduce the dimensionality of integration and further perform the integration using fast numerical methods.

Case 1: $\tilde{A}$ has a single attribute A_1^p; $R.A_1^p$ is probabilistic following a GMM and $S.A_1^p$ is deterministic which then is reduced as $S.A_1$. Then the existence probability of the join, $\int_{|R.A_1^p - S.A_1|<\delta} f_{R.A_1^p}(R.A_1^p) dR.A_1^p$, can be evaluated with 1-d Gaussian probabilities.

Case 2: $\tilde{A}$ has a single attribute A_1^p; Both $R.A_1^p$ and $S.A_1^p$ are probabilistic following GMMs. For the sake of simplicity, suppose the distribution of each attribute value is a mixture of two Gaussian, that is, $R.A_1^p \sim a_1 N(\mu_{R,1}, \sigma_{R,1}) + a_2 N(\mu_{R,2}, \sigma_{R,2})$ and $S.A_1^p \sim b_1 N(\mu_{S,1}, \sigma_{S,1}) + b_2 N(\mu_{S,2}, \sigma_{S,2})$. Then the cross-product follows $\sum_{i,j=1,2} a_i b_j N(\mu_{R,i}, \sigma_{R,i}) N(\mu_{S,j}, \sigma_{S,j})$. Therefore the existence probability of the join,

$$\int_{|R.A_1^p - S.A_1^p|<\delta} f_{R.A_1^p}(R.A_1^p) f_{S.A_1^p}(S.A_1^p) dR.A_1^p dS.A_1^p,$$

can be easily carried out, as the integrand is the summation of the product of two 1-d Gaussian which actually is a 2-d Gaussian and then a linear transformation can be applied to reduce the dimensionality of integration. Therefore, the integral is transformed to a univariate one on the variable $R.A_1^p - S.A_1^p$, which can be calculated with 1-d Gaussian probabilities.

Case 3: $\tilde{A}$ has multiple attributes; Both $\tilde{A}_R$ and $\tilde{A}_S$ are probabilistic following GMMs. For the sake of simplicity, suppose $\tilde{A} = (A_1^p, A_2^p)$. According to the linear transformation properties of Gaussian distributions, we have $R.A_1^p - S.A_1^p$ and $R.A_2^p - S.A_2^p$ follow a bivariate Gaussian distribution in the integral $\int_{|R.A_i^p - S.A_i^p|<\delta, i=1,\cdots,k} f(R.\tilde{A}, S.\tilde{A})\, \mathrm{d}R.\tilde{A}\, \mathrm{d}S.\tilde{A}$. Therefore, the 4-d integral is reduced to a 2-d one. Numerical integration methods can then be applied to calculate the integral with the reduced dimensionality (See for example, [32]).

6. PERFORMANCE EVALUATION

In this section, we evaluate our techniques for joins and aggregates, and compare them with sampling-based methods ([14] for aggregates and a home-grown method for joins) to demonstrate our performance benefits. We further perform a case study in the real-world application of tornado detection, and show that our techniques can improve the existing system with better detection results and faster computation.

6.1 Evaluation of Aggregation

We implemented a simulator to generate a tuple stream with one continuous random attribute. Each tuple is modeled by a mixture of two Gaussian components. The mean of the first component is uniformly sampled from [0, 5] and the mean of the second is from [5, 50]. This way, we can model a variety of real-world distributions, including bi-modal when the two means are far apart, asymmetric when they are somewhat close, and almost Gaussian when they are very close. The standard deviation of each Gaussian component is within [0.5, 1] by default. The coefficient of each component is uniform from [0, 1].

We evaluated `avg` over the above tuple stream using a tumbling window. The window size is measured by the number of tuples N. We measured both accuracy and throughput. The accuracy metric is the Variation Distance (VD) defined in Section 4.3. The default accuracy requirement is $VD \leq 0.1$ for each algorithm. All experiments were run on a 3Ghz dual-core xeon processor with 1GB memory for use in Java.

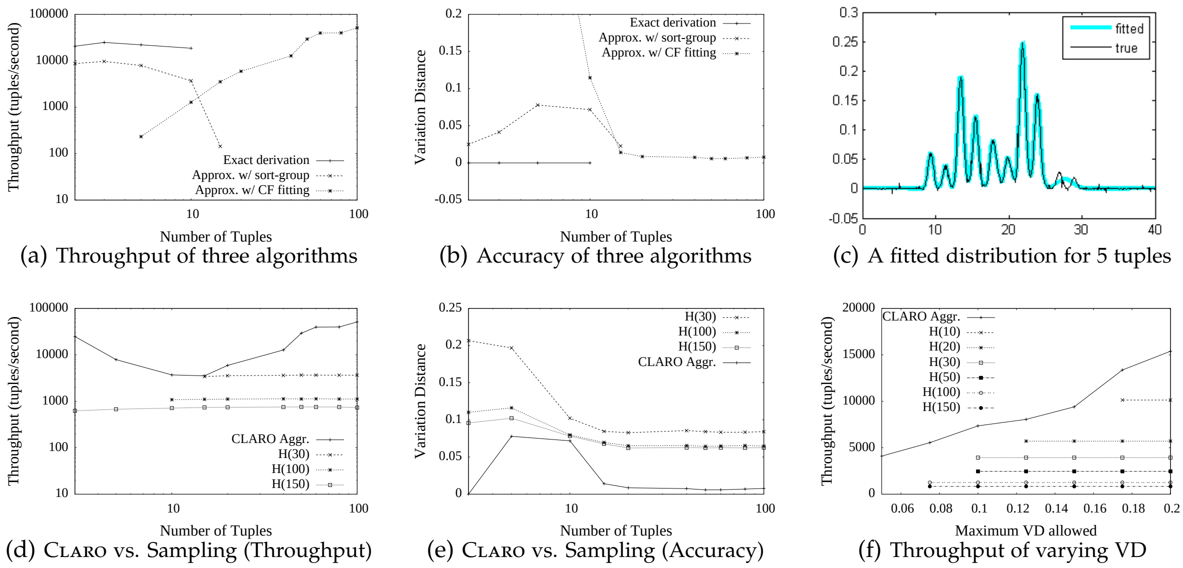
Expt 1: Compare our algorithms. We first compare the three algorithms that we proposed: exact derivation, approximation using sort-group, and approximation using characteristic function (CF) fitting. We varied the number of tuples in each window, N, since it directly affects the result distribution and the computation needed. The throughput and accuracy results are shown in the Fig. 5(a) and Fig. 5(b).

We observe that the throughout using exact derivation is high due to being a closed-form solution. However, we can only apply it to the small values of N because its formulas grow exponentially in N, hence becoming less and less useful. Sort-group is an approximation of the exact derivation, yielding simplified formulas. It achieves high throughput when N is small, e.g., up to 10, and deteriorates quickly after that. This is again because the size of the formula grows exponentially in N. In contrast, CF-fitting works well for large numbers of N, e.g., after 10. This is due to the smoother result distributions in this range, hence easier to fit, and the one-time fitting cost being amortized over more tuples.

For accuracy, most algorithms satisfy the requirement of VD ≤ 0.1. This is because both approximation algorithms compare with the true distributions through either direct VD comparison or function fitting with a small residual. We observe that the hardest range for approximation is 5 to 10 tuples. Result distributions in this range are complex and require a mixture of many components to fit. An example of the true and fitted distributions for 5 tuples is shown in Fig. 5(c). From 15 tuples onwards, the result distribution becomes smoother with fewer peaks. The only case of poor accuracy is CF fitting for less than 10 tuples. This is because we limited the maximum number of components it can search for to gain some performance.

We further did a cost breakdown for the two approximation algorithms. Table 1 presents different cost components for both 5 and 15 tuples. For sort-group, there are four cost factors: (1) enumerate all mixture components to generate the formula of the result distribution, (2) sort the components by the mean, (3) group them into K components, and (4) evaluate points from the true distribution for VD comparison. As N is increased from 5 to 15, all cost factors except grouping increase fast. This is because the number of mixture components to enumerate in the result formula is exponential in N, hence the dramatically increased costs.

The CF fitting algorithm has three cost factors: (1) point evaluation of the characteristic function of the true distribution, (2) get the initial guesses, and (3) fit by least squares. As expected, fitting is the dominating cost. More interestingly, this cost reduces significantly when the number of tuples is increased from 5 to 15. This is because more iterations of fitting are needed for 5 tuples due to the complex result distribution. Further, the one-time fitting cost occurs less frequently when we apply it once every 15 tuples rather than 5. This cost breakdown confirms that sort-group works well for a small number of tuples while CF fitting is the opposite.



(a) Throughput of three algorithms (b) Accuracy of three algorithms (c) A fitted distribution for 5 tuples

(d) Claro vs. Sampling (Throughput) (e) Claro vs. Sampling (Accuracy) (f) Throughput of varying VD

Figure 5: Experimental results for aggregation using our algorithms and histogram-based sampling

Table 1: Cost components of algorithms (msec/100tuples)

Sort-group			CF fitting		
Num of tuples	5	15	Num of tuples	5	15
Gen. Formula	0.93	258.34	Eval. Points	2.82	2.03
Sort	0.69	52.23	Get Guesses	1.66	0.53
Group	2.84	2.11	Fit	615.03	34.39
Eval. Points	15.35	1042.83			

The results of this experiment allow us to obtain a *hybrid solution* to combine the advantages of the three algorithms: When the number of tuples is smaller than 5, we use exact derivation since it is fast and its formula is not too complex. For the range of [5, 10], we use the sort-group algorithm. After that, we switch to CF fitting. This way, we gain the best throughput and accuracy among the three algorithms. In addition, when the number of tuples is large enough (e.g. ≥ 20), the result distributions are a smooth Gaussian most of the time. These distributions can be computed directly using the Central Limit Theorem (CLT). Hence, we can use CLT as an optimization and apply it when the number of tuples is greater than 30 (e.g., in Expt 3 below).

Expt 2: Compare to histogram-based sampling. Next, we compare our hybrid solution with the histogram based sampling algorithm [14]. Given N tuples, this algorithm (1) generates $k * \mu$ samples for each tuple, (2) performs aggregation over them to get $k * \mu$ result samples, and (3) sorts the result samples and builds a histogram with k buckets and μ samples for each bucket. k and μ are the parameters of this algorithm. Since we found the accuracy of this algorithm to be more sensitive to k, we used three settings in this experiment: k= 30, 100, or 150 while μ is fixed to 50.

Fig. 5(d) and Fig. 5(e) show the results. As observed, our hybrid algorithm outperforms the three settings of the histogram algorithm in both throughput and accuracy. In terms of accuracy, only the histogram with $k = 150$ ensures VD $\leq$ 0.1. The other two violate this in the "hard" range between 5 and 15 tuples (hence these throughput numbers are removed from Fig. 5(d)). These results show the advantages of our algorithm over the sampling based approach. Our algorithm can adapt to the accuracy requirement while maximizing the throughput. Using sampling, one has to manually choose the parameters to meet the requirement, and the optimal parameter setting varies with workloads.

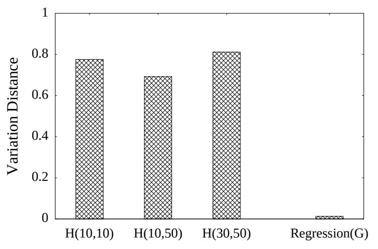
Expt 3: Vary the VD requirement. We further study how our algorithm behaves given different accuracy requirements. We varied VD from 0.05 to 0.2 and randomly chose N from [2,50] for each stream segment. We also included the histogram algorithm for comparison. Since its accuracy varies with the choice of k, we varied k in a wide range of [10, 150]. Fig. 5(f) shows the throughput results. For the histogram algorithm, throughput is reported only for the points where VD requirement is satisfied. As observed, our algorithm ourperforms the histogram algorithm for all values of VD used. It also confirms that our algorithm adapts well to the accuracy requirement. That is, under a relaxed condition, we can achieve better throughput.

6.2 Evaluation of Joins

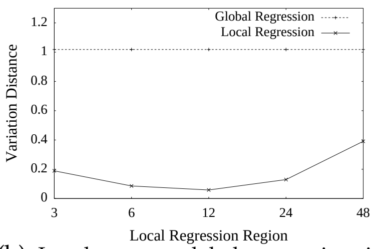
In this section, we evaluate joins between R=$\{a^p\}$, where a^p is a continuous random attribute, and S=$\{a, b\}$, where a is a deterministic attribute. We call R and S the *probe* and the *update* streams below. We design a sampling-based method as a baseline to compare with our regression-based methods.

Histogram-based sampling method. For each R tuple, this method takes samples from the distribution of $R.a^p$. It then extends each sample for a with a sample for b to construct the result distribution $f(a^p, b^p)$. Given the a sample, this method searches all S tuples in the update window for the closest two points based on the a value. It then applies linear interpolation to these points, with some added randomness, to obtain a b sample (such randomness is needed for the following step). Then it employs an equi-depth multidimensional histogram to approximately represent the joint distribution $f(a^p, b^p)$. Two parameters of the histogram affect performance: the number of buckets for each dimension, k, and the number of samples per bucket, μ.

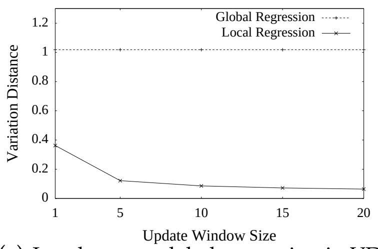
In this study, the R stream is an object location stream



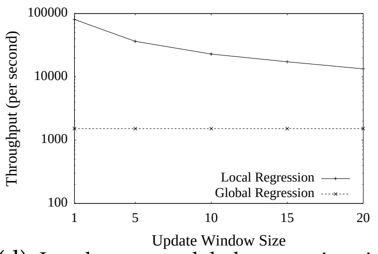
(a) Sampling versus global regression in VD (linear function).



(b) Local versus global regression in VD, with varying local regression region (quadratic function)

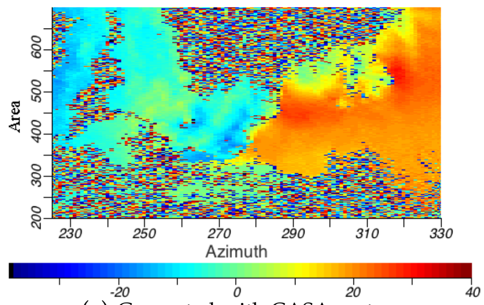


(c) Local versus global regression in VD, with varying update window (quadratic function)

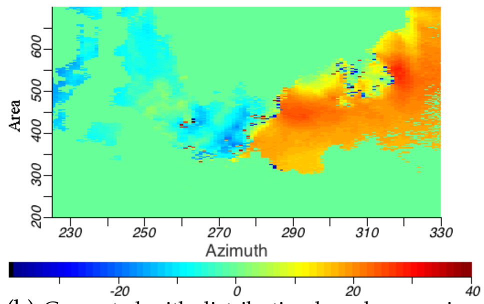


(d) Local versus global regression in Throughput, with varying update window (quadratic function)

Figure 6: Experimental results of joins using sampling, global regression, and local regression.



(a) Generated with CASA system.



(b) Generated with distribution-based processing.

Figure 7: Velocity maps of a true tornadic region.

generated from an RFID simulator [30]. Each location tuple has a Gaussian distribution. The S stream is produced by our temperature simulator that generates tuples, each as a (*location*,*temperature*) pair, by adding random noise to the underlying function between temperature and location. The underlying function can be either linear or quadratic in our experiments. The R and S tuples arrive at the same rate. We compute throughput based on the number of R tuples that can be pipelined through the left outer join.

Expt 4: Global regression versus sampling. We first use the linear underlying function to generate the temperature stream. We compare the histogram-based sampling with global regression. We use $\mathrm{H}(k, \mu)$ to represent histograms with different settings. As seen in Fig. 6(a), the sampling-based method is highly inaccurate (VD > 0.7) while global regression is much more accurate. The VD of the sampling method decreases as μ increases, e.g., from H(10,10) to H(10,50), because it uses a larger number of samples to construct the histogram. However, the VD worsens when k increases, e.g., from H(10,50) to H(30,50). This is because when k is large, the area of each bucket is very small, and hence the samples in each bucket mostly fit the noise added during interpolation. While it is possible to keep increasing μ, we note that sampling is already very slow: the throughput of H(10,10) is 34.8 and that of H(30,50) is down to 0.5. Our global regression approach gains a throughput of 1536.1, outperforming the sampling-based method by 2-3 orders of magnitude.

Expt 5: Global versus local regression. We next use the quadratic underlying function to generate the temperature stream and compare global versus local regression. Since local regression is sensitive to the number of data points available, which according to Section 5.1, is affected by both local regression (LR) region and update window size, we first set the update window size of local regression as 10, and vary its the LR region (which has no effect on global regression). As Fig. 6(b) shows, global regression yields poor accuracy as expected. The VD of local regression first decreases because it gets more points to do regression, hence more accurate. Then the VD increases because the region is too large to meet the assumption of local linearity; that is, the local regression is becoming more like the global regression.

We then fix the LR region as 6 and use the method of enlarging the update window, from 1 to 20, to feed more data points to local regression (while global regression still uses the window of size 1). This method is based on our belief that using a few old tuples will not add stale information because the underlying function changes slowly. Fig. 6(c) and Fig. 6(d) show that local regression outperforms global regression in both accuracy and speed. The increase of the update window improves the VD of local regression significantly due to the use of more points. Its throughput decreases for the same reason. However, even when local regression uses a window 20 times larger, its throughput is still much better than global regression due to the benefit of local computation.

6.3 Case Study: Tornado Detection

In this study, we evaluate the effectiveness of capturing uncertainty using distributions in the CASA system [3]. We focus on the tornado detection algorithm, and modify the velocity analysis module (see Figure 1) to generate velocity distributions using Gaussian Mixture Models. We keep the reflectivity analysis the same as in the current system.

We make a number of changes to the velocity analysis component of CASA. In the FFT module, a discrete distribution of velocity is obtained from Fourier Transform. Instead of

Table 2: Comparison between distribution-based data analysis and current CASA system (per-scan average).

	Analysis Time	Detection Time	False Positives
CASA	3.6 s	19.25 s	20.5
Distribution	7.17 s	5.75 s	1.75

taking a weighted average from the FFT distribution as in the current system, we apply three data analysis steps to generate a velocity distribution. (1) Strength Filter: If the strength of the FFT distribution is below a threshold, zero is output as the velocity value. The filter reduces unnecessary computation for distributions, as we can rarely get correct velocity from weak echoes. (2) Gaussian Estimation: The sample mean and sample variance of the normalized FFT distribution are used to estimate a Gaussian distribution $N(\mu, \sigma)$. If μ is close to the velocity with the highest strength in the FFT distribution and σ is small enough, the estimated distribution is considered to be accurate. Otherwise, do detailed analysis as follows. (3) GMM Fitting: Fit a mixture of two Gaussians from the FFT distribution. This model can capture useful velocity information using one Gaussian with a narrow peak, and noise using another Gaussian with a large variance; the latter is simply dropped. If a stream segment is on the boundary of two velocity regions, a GMM can also capture a bi-modal distribution of velocity.

For the average operation over the distributions of two stream segments, we apply the aggregation technique in Section 4.2 to capture the result distribution. Since the current tornado detection algorithm does not accept distributions as input, we feed the mean of distributions to the algorithm.

After applying all three steps of data analysis, we obtain a much smoother velocity map (Figure 7(b)) compared with what the current CASA system generates (Figure 7(a)). The Strength Filter removes many colorful dots on the map, i.e., noise, and the GMM Fitting smoothes velocity over the map by removing more noise. A detailed example of the noise removal effect of GMM fitting is shown in Figure 2(b) from Section 3. The velocity calculated in CASA is 5.6m/s. After dropping the Gaussian component of noise, the mean of the remaining Gaussian component is 6.4m/s, which is closer to the true value visually observed. This smoothing effect is favored as the true velocity changes gradually in space and the tornado detection algorithm expects smooth input data.

We further measure the speed of data analysis and the speed and quality of tornado detection. The evaluation is performed over a trace of 4 sector scans of a tornadic region. Each scan contains data of 7.25 seconds. As shown in Table 2, the distribution-based method reduces the detection time from 19.25s to 5.75s, and results in much fewer false positives. Although our method costs about twice data analysis time as the CASA system, it is still faster than the radar sensing speed. As such, our distribution-based method improves the detection algorithm to be stream-speed, and significantly improves the detection quality. We believe that the design of a distribution-based detection algorithm would give even better quality of detection results by fully utilizing our data analysis output.

7. EXTENSION TO CLARO

We now outline an extension of our system to support selections, projections, joins, and aggregates. First, we extend our data model by allowing each Gaussian Mixture Model to carry a selection condition. This condition bounds the GMM distribution to a finite region. Then, the tuple existence probability is the integration of the GMM over the selection region. The support of individual operators is sketched below:

Selections. Simply attach a selection condition to a GMM, incurring no computation. Equality predicates are rewritten to inequality predicates using a small bound δ.

Projections. Marginalize the GMM for the tuple distribution, eliminating the attributes not in the projection list. This operation is very efficient for GMMs. In the presence of a selection condition, directly marginalizing the GMM is equivalent to pushing the projection before the selection. Duplicate elimination is not needed for continuous attributes.

Joins. Since joins based on the cross-product semantics commute with selections, if we pull a selection over a join, the join can be evaluated the same as in §5.2. If probabilistic views are used instead, selections applied to the left join input can still commute with the left outer join. Selections applied to the right input (which is used to build the view) can filter tuples directly if they are applied to deterministic attributes, but require an extension of our current techniques if applied to probabilistic attributes involved in the view.

Aggregates: Since selections do not commute with aggregates, selection conditions attached to GMMs must be dealt with in aggregation. Consider sum(A) where A's pdf has a selection condition, denoted by f_θ. The effect of θ is to truncate the pdf and only keep the part within the region defined by θ. One method is to approximate f_θ using f itself if the edges of truncation have small probabilities. Another method is to approximate the truncated pdf using a histogram, compute using samples, and fit the result samples back to a GMM. Further consider avg(A). The selection applied before avg makes the existence probability of each tuple less than 1. Now count does not produce a deterministic value. Its distribution is further correlated to that of sum due to using the same set of tuples. To handle this, we will consider advanced techniques such as Monte Carlo simulation and the Delta method [12] when the expected number of count is large .

Finally, our discussion above raises interesting questions regarding query optimization under the Claro data model. One issue is where to place selections in a query plan. Evaluation of selections computes tuple existence probabilities, which incurs cost, but those probabilities can be used to filter tuples. Further, in scenarios when GMMs with selection regions cause difficulty in computation, we can temporarily switch to a histogram-based representation and use sampling in computation. This leads to a hybrid system using both representations. How to apply them to different parts of a query plan is another interesting issue to address. We will address these issues in our immediate future work.

8. RELATED WORK

Previous sections have discussed closely related work. Below, we survey several broader areas.

Probabilistic stream processing has gained research attention very recently. Existing work [5, 17, 20] adopts the finite and discrete tuple model used in probabilistic databases. As stated above, many of the techniques for discrete variables cannot be applied for problems involving continuous variables. Furthermore, the existing work such as [5] produces mean, variance, and higher moments to describe the result distributions, which cannot be easily used to compute the distribution of subsequent operators.

Models and views over uncertain data. Recent work on sensor data management [8, 7] builds statistical models to capture correlations among attributes and their changes over time. Given a query, such models enable reduced costs of data acquisition and communication. FunctionDB [29] transforms discrete sensor observations into continuous functions and supports querying over these functions. More relevant to us is supporting views over uncertain data [9, 19] where views are constructed using regression or probabilistic inference. However, the throughput of stream queries can be limited, e.g., 50 tuples/sec, due to high inference cost.

9. CONCLUSIONS

In this paper, we presented the Claro system for uncertain data stream processing. Claro employs a unique data model and advanced techniques for evaluating aggregates and joins over data streams. Our results show that Claro significantly outperforms sampling-based methods in accuracy and speed. Initial results of a case study further show that Claro can improve a tornado detection system with better quality of results and lower execution time.

We plan to extend our research in several directions. An immediate task is to extend Claro to a broader set of relational operators and address query optimization. We will also address inter-tuple correlations and explore advanced techniques such as lineage in stream processing. A third direction is to explore the combination of histograms and Gaussian Mixture Models to support both discrete and continuous random variables and to improve overall performance.

10. REFERENCES

[1] L. Antova, T. Jansen et al. Fast and simple relational processing of uncertain data. In *ICDE*, 983–992, 2008.

[2] O. Benjelloun, A. D. Sarma et al. Uldbs: Databases with uncertainty and lineage. In *VLDB*, 953–964, 2006.

[3] CASA. `http://www.casa.umass.edu/`.

[4] R. Cheng, D. V. Kalashnikov et al. Evaluating probabilistic queries over imprecise data. In *SIGMOD*, 551–562, 2003.

[5] G. Cormode and M. Garofalakis. Sketching probabilistic data streams. In *SIGMOD*, 281–292, 2007.

[6] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *VLDB J.*, 16(4):523–544, 2007.

[7] A. Deshpande, C. Guestrin et al. Exploiting correlated attributes in acquisitional query processing. In *ICDE*, 2005.

[8] A. Deshpande, C. Guestrin et al. Model-driven data acquisition in sensor networks. In *VLDB*, 588–599, 2004.

[9] A. Deshpande et al. MauveDB: supporting model-based user views in database systems. In *SIGMOD*, 2006.

[10] Y. Diao, B. Li et al. Capturing data uncertainty in high-volume stream processing. In *CIDR*, 2009.

[11] S. Frühwirth-Schnatter. *Finite Mixture and Markov Switching Models*. Springer, 2006.

[12] G. Casella and R. L. Berger. *Statistical Inference*. Duxbury, 2001.

[13] G. McLachlan et al. *Finite Mixture Models*. Wiley-Interscience.

[14] T. Ge and S. B. Zdonik. Handling uncertain data in array database systems. In *ICDE*, 1140–1149, 2008.

[15] L. Girod, Y. Mei et al. Xstream: a signal-oriented data stream management system. In *ICDE*, 1180–1189, 2008.

[16] C. Guestrin, P. Bodi et al. Distributed regression: an efficient framework for modeling sensor network data. In *IPSN*, 2004.

[17] T. S. Jayram, A. McGregor et al. Estimating statistical aggregates on probabilistic data streams. In *PODS*, 2007.

[18] S. R. Jeffery, M. J. Franklin et al. An adaptive RFID middleware for supporting metaphysical data independence. *VLDB J.*, 17(2):256–289, 2008.

[19] B. Kanagal and A. Deshpande. Online filtering, smoothing and probabilistic modeling of streaming data. In *ICDE*, 2008.

[20] B. Kanagal and A. Deshpande. Efficient query evaluation over temporally correlated probabilistic streams. In *ICDE*, 2009.

[21] C. Koch and D. Olteanu. Conditioning probabilistic databases. In *VLDB*, 2008.

[22] J. F. Kurose, E. Lyons et al. An end-user-responsive sensor network architecture for hazardous weather detection, prediction and response. In *AINTEC*, 1–15, 2006.

[23] B. G. Lindsay. *Mixture Models: Theory, Geometry and Applications*. Inst of Mathematical Statistic, 1996.

[24] C. Ré, J. Letchner et al. Event queries on correlated probabilistic streams. In *SIGMOD*, 715–728, 2008.

[25] C. Ré and D. Suciu. Approximate lineage for probabilistic databases. In *VLDB*, 2008.

[26] T. Sauer. *Numerical Analysis*. Addison Wesley, 2005.

[27] P. Sen, A. Deshpande et al. Exploiting shared correlations in probabilistic databases. In *VLDB*, 2008.

[28] S. Singh, C. Mayfield et al. Database support for probabilistic attributes and tuples. In *ICDE*, 1053–1061, 2008.

[29] A. Thiagarajan and S. Madden. Querying continuous functions in a database system. In *SIGMOD*, 791–804, 2008.

[30] T. Tran, C. Sutton et al. Probabilistic inference over rfid streams in mobile environments. In *ICDE*, 2009.

[31] D. Z. Wang, E. Michelakis et al. Bayesstore: Managing large, uncertain data repositories with probabilistic graphical models. In *VLDB*, 2008.

[32] A. Genz. Numerical computation of multivariate normal probabilities. In *Journal of Computational and Graphical Statistics*, 1992.

Appendix

Proof of Theorem 4.1.

We first consider the sum of two variables, $S = X_1 + X_2$, X_1 and X_2 be mixtures of m_1 and m_2 components. That is:

$$\begin{aligned} f_1(x) &= p_{11}N(\mu_{11}, \sigma_{11}) + ... + p_{1m_1}N(\mu_{1m_1}, \sigma_{1m_1}) \\ f_2(x) &= p_{21}N(\mu_{21}, \sigma_{21}) + ... + p_{2m_2}N(\mu_{2m_2}, \sigma_{2m_2}) \end{aligned}$$

The pdf of the sum S can be written as:

$$\begin{aligned} f_S(s) &= \int_{x_1} \int_{x_2:(x_1+x_2=s)} f_1(x_1) f_2(x_2) dx_2 dx_1 \\ &= \int_{-\infty}^{+\infty} f_1(x) f_2(s-x) dx \\ f_1(x)f_2(s-x) &= [p_{11}N(\mu_{11}, \sigma_{11}) + ... + p_{1m_1}N(\mu_{1m_1}, \sigma_{1m_1})] \\ &\quad [p_{21}N(\mu_{21}, \sigma_{21}) + ... + p_{2m_2}N(\mu_{2m_2}, \sigma_{2m_2})] \\ &= \sum_{i=1,j=1}^{m_1,m_2} p_{1i} p_{2j} \frac{1}{2\pi\sigma_{1i}\sigma_{2j}} e^{-\frac{1}{2}\left(\frac{(x-\mu_{1i})^2}{\sigma_{11}^2} + \frac{(x-\mu_{2j})^2}{\sigma_{2j}^2}\right)} \end{aligned}$$

Now consider the integral of one term of the sum:

$$A = \int_{-\infty}^{+\infty} p_{1i} p_{2j} \frac{1}{2\pi\sigma_{1i}\sigma_{2j}} e^{-\frac{1}{2}\left(\frac{(x-\mu_{1i})^2}{\sigma_{1i}^2} + \frac{(x-\mu_{2j})^2}{\sigma_{2j}^2}\right)}$$

Let B be the term in the exponent:

$$\begin{aligned} B &= \frac{(x-\mu_{1i})^2}{\sigma_{1i}^2} + \frac{(x-\mu_{2j})^2}{\sigma_{2j}^2} \\ &= \frac{1}{\sigma_{11}^2\sigma_{21}^2}\left[(\sigma_{1i}^2 + \sigma_{2j}^2)x^2 - 2\sigma_{2j}^2 x\mu_{1i} + 2\sigma_{1i}^2 x(\mu_{2j} - s) + \sigma_{2j}^2\mu_{1i}^2 + \sigma_{1i}^2(\mu_{2j} - s)^2\right] \\ &= \frac{\sigma_{1i}^2 + \sigma_{2j}^2}{\sigma_{11}^2\sigma_{21}^2}\left[\left(x - \frac{\sigma_{2j}^2\mu_{1i} + \sigma_{1i}^2(s - \mu_2 j)}{\sigma_{1i}^2 + \sigma_{2j}^2}\right)^2 - \left(\frac{\sigma_{2j}^2\mu_{1i} + \sigma_{1i}^2(s - \mu_{2j})}{\sigma_{1i}^2 + \sigma_{2j}^2}\right)^2 + \frac{\sigma_{2j}^2\mu_{1i}^2 + \sigma_{1i}^2(s - \mu_{2j})^2}{\sigma_{1i}^2 + \sigma_{2j}^2}\right] \\ &= \frac{\sigma_{1i}^2 + \sigma_{2j}^2}{\sigma_{1i}^2\sigma_{2j}^2}\left[\left(x - \frac{\sigma_{2j}^2\mu_{1i} + \sigma_{1i}^2(s - \mu_2 j)}{\sigma_{1i}^2 + \sigma_{2j}^2}\right)^2 + \frac{\sigma_{1i}^2 + \sigma_{2j}^2(s - \mu_{1i} - \mu_{2j})^2}{(\sigma_{1i}^2\sigma_{2j}^2)^2}\right] \end{aligned}$$

Substitute into A, we have:

$$\begin{aligned} A &= \int_{-\infty}^{+\infty} p_{1i} p_{2j} \frac{1}{2\pi\sigma_{1i}\sigma_{2j}} e^{\frac{1}{2}\frac{\sigma_{1i}^2+\sigma_{2j}^2}{\sigma_{1i}^2\sigma_{2j}^2}\left(x - \frac{\sigma_{2j}^2\mu_{1i}+\sigma_{1i}^2(s-\mu_2 j)}{\sigma_{1i}^2+\sigma_{2j}^2}\right)^2} e^{\frac{(s-\mu_{1i}-\mu_{2j})^2}{\sigma_{1i}^2+\sigma_{2j}^2}} dx \\ &= \frac{p_{1i}p_{2j}}{\sqrt{2\pi}\sqrt{\sigma_{1i}^2 + \sigma_{2j}^2}} e^{\frac{(s-\mu_{1i}-\mu_{2j})^2}{\sigma_{1i}^2+\sigma_{2j}^2}} \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}\frac{\sigma_{1i}\sigma_{2j}}{\sqrt{\sigma_{1i}^2+\sigma_{2j}^2}}} e^{\frac{1}{2}\frac{\sigma_{1i}^2+\sigma_{2j}^2}{\sigma_{1i}^2\sigma_{2j}^2}\left(x - \frac{\sigma_{2j}^2\mu_{1i}+\sigma_{1i}^2(s-\mu_{2j})}{\sigma_{1i}^2+\sigma_{2j}^2}\right)^2} dx \end{aligned}$$

The integral is equal to 1 since it is the integral of the pdf of a Gaussian distribution. Hence:

$$A = \frac{p_{1i}p_{2j}}{\sqrt{2\pi}\sqrt{\sigma_{1i}^2 + \sigma_{2j}^2}} e^{\frac{(s-\mu_{1i}-\mu_{2j})^2}{\sigma_{1i}^2+\sigma_{2j}^2}}$$

This is one component of a Gaussian mixture with mean $(\mu_{1i} + \mu_{2j})$, variance $(\sigma_{1i}^2 + \sigma_{2j}^2)$ and coefficient $p_{1i}p_{2j}$. Therefore, the theorem is proved for the case of $N = 2$.
The generalization to an arbitrary N is straightforward since we can do the sum for two distributions at a time by getting the result of previous sum and summing it with the next distribution.